# Techniques for Web Content Accessibility Guidelines 1.0

## W3C Working Draft 23 June 2000

This version:
http://www.w3.org/WAI/GL/WD-WCAG10-TECHS-20000623
(plain text, postscript, pdf, gzip tar file of HTML, zip archive of HTML)
Latest version:
http://www.w3.org/WAI/GL/WCAG10-TECHS
Previous version:
http://www.w3.org/WAI/GL/WD-WCAG10-TECHS-20000615/
Latest version of "Web Content Accessibility Guidelines 1.0"
http://www.w3.org/WAI/GL/WCAG10
Editors:
Wendy Chisholm, W3C,
Gregg Vanderheiden, Trace R & D Center, University of Wisconsin -- Madison
Ian Jacobs, W3C

Copyright ©1999 - 2000 W3C® (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

---

## Abstract

This document provides techniques for implementing the checkpoints defined in "Web Content Accessibility Guidelines 1.0".

This document is part of a series of accessibility documents published by the Web Accessibility Initiative.

## Status of this document

This is a W3C Working Draft for review by the Web Content Accessibility Guidelines Working Group and other invited parties. It should eventually supersede the current W3C Note Techniques for Web Content Accessibility Guidelines 1.0. This is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

While Web Content Accessibility Guidelines 1.0 strives to be a stable document (as a W3C Recommendation), the current document is expected to evolve as technologies change and content developers discover more effective techniques for designing accessible Web sites and pages.

This document has been produced as part of the W3C Web Accessibility Initiative. The goal of the Web Content Guidelines Working Group is discussed in the Working Group charter.

The list of known errors in this document is available at http://www.w3.org/WAI/GL/WAI-WEBCONTENT-ERRATA. Please send detailed comments on this document to the working group at w3c-wai-gl@w3.org.

# Table of Contents

Abstract . . . . . .1
Status of this document . . . . . .1
1 Introduction . . . . . .5
2 Priorities . . . . . .5
3 How the Techniques are Organized . . . . . .5
3.1 Technology specific external resources . . . . . .5
3.2 Examples and Deprecated Examples . . . . . .6
4 Techniques for Web Content Accessibility Guidelines . . . . . .7
1. Provide equivalent alternatives to auditory and visual content. . . .7
2. Don't rely on color alone. . . . . .8
3. Use markup and style sheets and do so properly. . . . . .8
4. Clarify natural language usage . . . . . .9
5. Create tables that transform gracefully. . . . . .9
6. Ensure that pages featuring new technologies transform gracefully. 10
7. Ensure user control of time-sensitive content changes. . . . 10
8. Ensure direct accessibility of embedded user interfaces. . . . 11
9. Design for device-independence. . . . . 12
10. Use interim solutions. . . . . 12
11. Use W3C technologies and guidelines. . . . . 13
12. Provide context and orientation information. . . . . 14
13. Provide clear navigation mechanisms. . . . . 14
14. Ensure that documents are clear and simple. . . . . 15
5 Accessibility Themes . . . . . 15
5.1 Structure vs. Presentation . . . . . 16
5.2 Text equivalents . . . . . 17
5.3 Alternative pages . . . . . 18
5.4 Keyboard access . . . . . 19
5.5 Navigation . . . . . 20
5.6 Comprehension . . . . . 20
5.7 Content negotiation . . . . . 22
5.8 Automatic page refresh . . . . . 22
5.9 Screen flicker . . . . . 22
5.10 Bundled documents . . . . . 23
5.11 Validation . . . . . 23
5.12 Browser Support . . . . . 24
5.13 Accessibility Reviewed Technologies . . . . . 25
5.14 Audio and Video . . . . . 25
5.15 Audio information . . . . . 25
5.16 Visual information and motion . . . . . 26
5.17 Collated text transcripts . . . . . 27
6 Glossary . . . . . 27

7 Acknowledgments . . . . . 34
8 References . . . . . 35
9 Resources . . . . . 36
9.1 Operating system and programming guidelines . . . . . 36
9.2 User agents and other tools . . . . . 37
9.3 Accessibility resources . . . . . 37

# 1 Introduction

# 2 Priorities

Each checkpoint has a priority level assigned by the Working Group based on the checkpoint's impact on accessibility.

[Priority 1]
: A Web content developer **must** satisfy this checkpoint. Otherwise, one or more groups will find it impossible to access information in the document. Satisfying this checkpoint is a basic requirement for some groups to be able to use Web documents.

[Priority 2]
: A Web content developer **should** satisfy this checkpoint. Otherwise, one or more groups will find it difficult to access information in the document. Satisfying this checkpoint will remove significant barriers to accessing Web documents.

[Priority 3]
: A Web content developer **may** address this checkpoint. Otherwise, one or more groups will find it somewhat difficult to access information in the document. Satisfying this checkpoint will improve access to Web documents.

Some checkpoints specify a priority level that may change under certain (indicated) conditions.

# 3 How the Techniques are Organized

Section 4 of this document reproduces the guidelines and checkpoints of the "Web Content Accessibility Guidelines 1.0" [WCAG10] [p. 36] . Each checkpoint definition includes a link to the checkpoint definition in Web Content Accessibility Guidelines 1.0 [WCAG10] [p. 36] .

## 3.1 Technology specific external resources

This document introduces some general techniques to promote accessibility that are independent of any specific markup language. It also refers to the following resources:

HTML Techniques [WCAG10-HTML-TECHNIQUES] [p. 36]
: This document explains how to implement applicable checkpoints in HTML (refer to [HTML4] [p. 35] , [HTML32] [p. 35] ) and includes numerous practical examples.

CSS Techniques [WCAG10-CSS-TECHNIQUES] [p. 36]
: This section explains how to implement applicable checkpoints in CSS1 and CSS2 (refer to [CSS1] [p. 35] , [CSS2] [p. 35] ).

Non-W3C Techniques [WCAG10-NONW3C-TECHNIQUES] [p. 36]
This section explains how to implement applicable checkpoints for formats that are developed outside of W3C.

## 3.2 Examples and Deprecated Examples

This document contains a number of examples that illustrate accessible solutions in HTML, CSS, etc. but also deprecated examples that illustrate what content developers should not do. The deprecated examples are highlighted and readers should approach them with caution -- they are meant for illustrative purposes only.

# 4 Techniques for Web Content Accessibility Guidelines

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

Checkpoints:

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] (Checkpoint 1.1)

- In this document: Text equivalents [p. 17]
- HTML Techniques: Images used as bullets
- HTML Techniques: Link text
- HTML Techniques: Short text equivalents for images ("alt-text")
- HTML Techniques: Long descriptions of images
- HTML Techniques: Text and non-text equivalents for applets and programmatic objects
- HTML Techniques: Text equivalents for multimedia
- HTML Techniques: Describing frame relationships
- HTML Techniques: Writing for browsers that do not support FRAME
- HTML Techniques: Graphical buttons
- HTML Techniques: Alternative presentation of scripts
- Non-W3C Technology Techniques: Static images

1.2 Provide redundant text links for each active region of a server-side image map. [Priority 1] (Checkpoint 1.2)

Refer also to checkpoint 1.5 and checkpoint 9.1.

- In this document: Text equivalents [p. 17]
- HTML Techniques: Server-side image maps

1.3 Until user agents [p. 32] 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] (Checkpoint 1.3)

- In this document: Visual information and motion [p. 26]

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] (Checkpoint 1.4)

- In this document: Audio information [p. 25]
- HTML Techniques: Audio and Video produced by dynamic objects

1.5 Until user agents [p. 32] 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] (Checkpoint 1.5)

Refer also to checkpoint 1.2 and checkpoint 9.1.

- In this document: Text equivalents [p. 17]

- HTML Techniques: Client-side image maps

## Guideline 2. Don't rely on color alone.

Checkpoints:

2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup. [Priority 1] (Checkpoint 2.1)

- In this document: Structure vs. Presentation [p. 16]
- CSS Techniques: Colors

2.2 Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen. [Priority 2 for images, Priority 3 for text]. (Checkpoint 2.2)

- HTML Techniques: Color in images
- CSS Techniques: Colors

## Guideline 3. Use markup and style sheets and do so properly.

Checkpoints:

3.1 When an appropriate markup language exists, use markup rather than images to convey information. [Priority 2] (Checkpoint 3.1)

- In this document: Structure vs. Presentation [p. 16]
- HTML Techniques: Markup and style sheets rather than images: The example of math

3.2 Create documents that validate to published formal grammars. [Priority 2] (Checkpoint 3.2)

- HTML Techniques: Metadata
- HTML Techniques: The !DOCTYPE statement

3.3 Use style sheets to control layout and presentation. [Priority 2] (Checkpoint 3.3)

- In this document: Structure vs. Presentation [p. 16]
- HTML Techniques: Emphasis
- CSS Techniques: Text instead of images
- CSS Techniques: Text formatting
- CSS Techniques: Layout, positioning, layering, and alignment

3.4 Use relative rather than absolute units in markup language attribute values and style sheet property values. [Priority 2] (Checkpoint 3.4)

- HTML Techniques: Directly accessible applets
- HTML Techniques: Sizing frames with relative units
- CSS Techniques: Guidelines for creating style sheets

3.5 Use header elements to convey document structure and use them according to specification. [Priority 2] (Checkpoint 3.5)

- In this document: Structure vs. Presentation [p. 16]
- HTML Techniques: Section headers

3.6 Mark up lists and list items properly. [Priority 2] (Checkpoint 3.6)

- In this document: Structure vs. Presentation [p. 16]
- HTML Techniques: Lists

3.7 Mark up quotations. Do not use quotation markup for formatting effects such as indentation. [Priority 2] (Checkpoint 3.7)

- HTML Techniques: Quotations

## Guideline 4. Clarify natural language usage

Checkpoints:

4.1 Clearly identify changes in the natural language of a document's text and any text equivalents [p. 30] (e.g., captions). [Priority 1] (Checkpoint 4.1)

- HTML Techniques: Language information

4.2 Specify the expansion of each abbreviation or acronym in a document where it first occurs. [Priority 3] (Checkpoint 4.2)

- HTML Techniques: Acronyms and abbreviations

4.3 Identify the primary natural language of a document. [Priority 3] (Checkpoint 4.3)

- HTML Techniques: Language information

## Guideline 5. Create tables that transform gracefully.

Checkpoints:

5.1 For data tables, identify row and column headers. [Priority 1] (Checkpoint 5.1)

- HTML Techniques: Tables of data

5.2 For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells. [Priority 1] (Checkpoint 5.2)

- HTML Techniques: Tables of data

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 [p. 31] ). [Priority 2] (Checkpoint 5.3)

- In this document: Structure vs. Presentation [p. 16]
- HTML Techniques: Tables for layout
- CSS Techniques: Layout, positioning, layering, and alignment

5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting. [Priority 2] (Checkpoint 5.4)

- In this document: Structure vs. Presentation [p. 16]
- HTML Techniques: Tables for layout

5.5 Provide summaries for tables. [Priority 3] (Checkpoint 5.5)

- HTML Techniques: Tables of data

5.6 Provide abbreviations for header labels. [Priority 3] (Checkpoint 5.6)

- HTML Techniques: Tables of data

Refer also to checkpoint 10.3.

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

Checkpoints:

6.1 Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document. [Priority 1] (Checkpoint 6.1)

- CSS Techniques: Creating style sheets that transform gracefully

6.2 Ensure that equivalents for dynamic content are updated when the dynamic content changes. [Priority 1] (Checkpoint 6.2)

- HTML Techniques: Text and non-text equivalents for applets and programmatic objects
- HTML Techniques: Frame sources
- HTML Techniques: Alternative presentation of scripts

6.3 Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page. [Priority 1] (Checkpoint 6.3)

- HTML Techniques: Text and non-text equivalents for applets and programmatic objects
- HTML Techniques: Graceful transformation of scripts

6.4 For scripts and applets, ensure that event handlers are input device-independent. [Priority 2] (Checkpoint 6.4)

- In this document: Structure vs. Presentation [p. 16]
- HTML Techniques: Directly accessible applets
- HTML Techniques: Device-independent event handlers

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

- In this document: Alternative pages [p. 18]
- In this document: Audio information [p. 25]
- HTML Techniques: Directly accessible applets
- HTML Techniques: Writing for browsers that do not support FRAME
- HTML Techniques: Graceful transformation of scripts
- Non-W3C Technology Techniques: Providing alternative images

Refer also to checkpoint 11.4.

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

Checkpoints:

7.1 Until user agents [p. 32] allow users to control flickering, avoid causing the screen to flicker. [Priority 1] (Checkpoint 7.1)

- In this document: Screen flicker [p. 22]
- In this document: Visual information and motion [p. 26]
- HTML Techniques: Directly accessible applets
- HTML Techniques: Graceful transformation of scripts

7.2 Until user agents [p. 32] allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off). [Priority 2] (Checkpoint 7.2)

- HTML Techniques: Directly accessible applets
- HTML Techniques: Graceful transformation of scripts
- CSS Techniques: Text style

7.3 Until user agents [p. 32] allow users to freeze moving content, avoid movement in pages. [Priority 2] (Checkpoint 7.3)

- In this document: Visual information and motion [p. 26]
- HTML Techniques: Animated images
- HTML Techniques: Directly accessible applets
- HTML Techniques: Graceful transformation of scripts
- CSS Techniques: Creating style sheets that transform gracefully

7.4 Until user agents [p. 32] provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2] (Checkpoint 7.4)

- In this document: Automatic page refresh [p. 22]
- HTML Techniques: Metadata
- HTML Techniques: Directly accessible applets
- HTML Techniques: Page updates and new windows

7.5 Until user agents [p. 32] provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects. [Priority 2] (Checkpoint 7.5)

- In this document: Automatic page refresh [p. 22]
- HTML Techniques: Metadata
- HTML Techniques: Page updates and new windows

**Note.** The BLINK and MARQUEE elements are not defined in any W3C HTML specification and should not be used. Refer also to guideline 11.

## Guideline 8. Ensure direct accessibility of embedded user interfaces.

Checkpoint:

8.1 Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important [p. 31] and not presented elsewhere, otherwise Priority 2.] (Checkpoint 8.1)

Refer also to guideline 6.

- HTML Techniques: Directly accessible applets

- HTML Techniques: Audio and Video produced by dynamic objects
- HTML Techniques: Graceful transformation of scripts

## Guideline 9. Design for device-independence.

Checkpoints:

9.1 Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape. [Priority 1] (Checkpoint 9.1)

Refer also to checkpoint 1.1, checkpoint 1.2, and checkpoint 1.5.

- HTML Techniques: Client-side image maps

9.2 Ensure that any element that has its own interface can be operated in a device-independent manner. [Priority 2] (Checkpoint 9.2)

Refer to the definition of device independence [p. 29] .

Refer also to guideline 8.

- In this document: Keyboard access [p. 19]
- HTML Techniques: Directly accessible applets

9.3 For scripts, specify logical event handlers rather than device-dependent event handlers. [Priority 2] (Checkpoint 9.3)

- In this document: Keyboard access [p. 19]
- HTML Techniques: Device-independent event handlers

9.4 Create a logical tab order through links, form controls, and objects. [Priority 3] (Checkpoint 9.4)

- In this document: Keyboard access [p. 19]
- HTML Techniques: Keyboard access
- HTML Techniques: Keyboard access to forms

9.5 Provide keyboard shortcuts to important links (including those in client-side image maps [p. 31] ), form controls, and groups of form controls. [Priority 3] (Checkpoint 9.5)

- In this document: Keyboard access [p. 19]
- HTML Techniques: Keyboard access
- HTML Techniques: Client-side image maps
- HTML Techniques: Keyboard access to forms

## Guideline 10. Use interim solutions.

Checkpoints:

10.1 Until user agents [p. 32] allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user. [Priority 2] (Checkpoint 10.1)

- HTML Techniques: Anchors and targets
- HTML Techniques: Directly accessible applets
- HTML Techniques: Using FRAME targets
- HTML Techniques: Page updates and new windows

10.2 Until user agents [p. 32] support explicit associations between labels and form controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned. [Priority 2] (Checkpoint 10.2)

- HTML Techniques: Labeling form controls

10.3 Until user agents [p. 32] (including assistive technologies) render side-by-side text correctly, provide a linear text alternative (on the current page or some other) for *all* tables that lay out text in parallel, word-wrapped columns. [Priority 3] (Checkpoint 10.3)

- HTML Techniques: Linearizing tables

10.4 Until user agents [p. 32] handle empty controls correctly, include default, place-holding characters in edit boxes and text areas. [Priority 3] (Checkpoint 10.4)

- HTML Techniques: Techniques for specific controls

10.5 Until user agents [p. 32] (including assistive technologies) render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links. [Priority 3] (Checkpoint 10.5)

- HTML Techniques: Grouping and bypassing links
- HTML Techniques: Client-side image maps

## Guideline 11. Use W3C technologies and guidelines.

Checkpoints:

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

- In this document: Accessibility Reviewed Technologies [p. 25]

11.2 Avoid deprecated features of W3C technologies. [Priority 2] (Checkpoint 11.2)

- HTML Techniques: Index of HTML elements and attributes
- CSS Techniques: Fonts

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

**Note.** Use content negotiation where possible.

- In this document: Content negotiation [p. 22]
- HTML Techniques: Metadata
- CSS Techniques: Creating style sheets that transform gracefully

11.4 If, after best efforts [p. 13] , you cannot create an accessible [p. 27] page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent [p. 30] information (or functionality), and is updated as often as the inaccessible (original) page. [Priority 1] (Checkpoint 11.4)

- In this document: Alternative pages [p. 18]

**Note.** Content developers should only resort to alternative pages when other solutions fail because alternative pages are generally updated less often than "primary" pages. An out-of-date page may be as frustrating as one that is inaccessible since, in both cases, the information presented on the original page is unavailable. Automatically generating alternative pages may lead to more frequent updates, but content developers must still be careful to ensure that generated pages

always make sense, and that users are able to navigate a site by following links on primary pages, alternative pages, or both. Before resorting to an alternative page, reconsider the design of the original page; making it accessible is likely to improve it for all users.

## Guideline 12. Provide context and orientation information.

Checkpoints:

12.1 Title each frame to facilitate frame identification and navigation. [Priority 1] (Checkpoint 12.1)

- HTML Techniques: Providing a frame title

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] (Checkpoint 12.2)

- In this document: Text equivalents [p. 17]
- HTML Techniques: Describing frame relationships

12.3 Divide large blocks of information into more manageable groups where natural and appropriate. [Priority 2] (Checkpoint 12.3)

- HTML Techniques: Structural grouping
- HTML Techniques: Grouping form controls

12.4 Associate labels explicitly with their controls. [Priority 2] (Checkpoint 12.4)

- HTML Techniques: Labeling form controls

## Guideline 13. Provide clear navigation mechanisms.

Checkpoints:

13.1 Clearly identify the target of each link. [Priority 2] (Checkpoint 13.1)

- HTML Techniques: Link text
- HTML Techniques: Client-side image maps

13.2 Provide metadata to add semantic information to pages and sites. [Priority 2] (Checkpoint 13.2)

- In this document: Navigation [p. 20]
- HTML Techniques: Metadata
- HTML Techniques: The LINK element and navigation tools

13.3 Provide information about the general layout of a site (e.g., a site map or table of contents). [Priority 2] (Checkpoint 13.3)

- In this document: Navigation [p. 20]

13.4 Use navigation mechanisms in a consistent manner. [Priority 2] (Checkpoint 13.4)

- In this document: Navigation [p. 20]

13.5 Provide navigation bars to highlight and give access to the navigation mechanism. [Priority 3] (Checkpoint 13.5)

- In this document: Navigation [p. 20]

13.6 Group related links, identify the group (for user agents), and, until user agents [p. 32] do so, provide a way to bypass the group. [Priority 3] (Checkpoint 13.6)

- HTML Techniques: Structural grouping
- HTML Techniques: Grouping and bypassing links

13.7 If search functions are provided, enable different types of searches for different skill levels and preferences. [Priority 3] (Checkpoint 13.7)

- In this document: Navigation [p. 20]

13.8 Place distinguishing information at the beginning of headings, paragraphs, lists, etc. [Priority 3] (Checkpoint 13.8)

- In this document: Comprehension [p. 20]

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

For example, in HTML specify document collections with the LINK element and the "rel" and "rev" attributes. Another way to create a collection is by building an archive (e.g., with zip, tar and gzip, stuffit, etc.) of the multiple pages.

- In this document: Bundled documents [p. 23]
- HTML Techniques: The LINK element and navigation tools

13.10 Provide a means to skip over multi-line ASCII art. [Priority 3] (Checkpoint 13.10)

- HTML Techniques: Ascii art

## Guideline 14. Ensure that documents are clear and simple.

Checkpoints:

14.1 Use the clearest and simplest language appropriate for a site's content. [Priority 1] (Checkpoint 14.1)

- In this document: Comprehension [p. 20]

14.2 Supplement text with graphic or auditory presentations where they will facilitate comprehension of the page. [Priority 3] (Checkpoint 14.2)

- In this document: Comprehension [p. 20]

14.3 Create a style of presentation that is consistent across pages. [Priority 3] (Checkpoint 14.3)

- In this document: Navigation [p. 20]
- CSS Techniques: Guidelines for creating style sheets

# 5 Accessibility Themes

The following sections discuss some accessibility themes that Web content developers should keep in mind as they design documents and sites.

## 5.1 Structure vs. Presentation

Checkpoints in this section: 2.1, 3.1, 3.3, 3.5, 3.6, 5.3, 5.4, and 6.4.

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 rule (the HR element) communicates a structural division. This may be true for sighted users, but to unsighted users or users without graphical browsers, a horizontal rule has next to no meaning (One might "guess" that an HR element implies a structural division, but without other information, there is no guarantee.) 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. 37] ,

> 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.

## 5.2 Text equivalents

Checkpoints in this section: 1.1, 1.2, 1.5, and 12.2.

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?

### *5.2.1 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. 35] ) 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. 35] ) content developers may also supply information there as well.

### *5.2.2 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. 37] ). 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).

## 5.3 Alternative pages

Checkpoints in this section: 11.4, and 6.5.

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 and maintained with the same frequency as the inaccessible original 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. For example, in HTML, use the LINK element as follows:

   **Example.**

   User agents that support LINK will load the alternative page for those users whose browsers may be identified as supporting "aural","braille", or "tty" rendering.

```
<HEAD>
<TITLE>Welcome to the Virtual Mall!</TITLE>
<LINK title="Text-only version"
      rel="alternate"
      href="text_only"
      media="aural, braille, tty">
</HEAD>
<BODY><P>...</BODY>
```

End example.

## 5.4 Keyboard access

Checkpoints in this section: 9.2, 9.3, 9.4, and 9.5.

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.

### *5.4.1 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.

## 5.5 Navigation

Checkpoints in this section: 14.3, 13.4, 13.5, 13.3, 13.7, and 13.2.

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.

## 5.6 Comprehension

Checkpoints in this section: 14.1, 13.8, and 14.2.

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

### *5.6.1 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. 36] ) 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. 36] with examples and the algorithm online at [TECHHEAD] [p. 37] ). 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.

### *5.6.2 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.

## 5.7 Content negotiation

Checkpoints in this section: 11.3.

1. Use content negotiation to serve content per the client request. For example, serve the French version of a document to clients requesting French.
2. If not possible to use content negotiation, indicate content type or language through markup (e.g., in HTML use "type" and "hreflang").
3. 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.

## 5.8 Automatic page refresh

Checkpoints in this section: 7.4, and 7.5.

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 by 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. @@link

## 5.9 Screen flicker

Checkpoints in this section: 7.1.

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).

## 5.10 Bundled documents

Checkpoints in this section: 13.9.

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.

## 5.11 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. 13] that is accessible.

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

### *5.11.1 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. 37] ).
2. An HTML validation service such as the W3C HTML Validation Service (refer to [HTMLVAL] [p. 37] ).
3. A style sheets validation service such as the W3C CSS Validation Service (refer to [CSSVAL] [p. 37] ).

### *5.11.2 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. 37] ) is working to develop a suite of tools that will help authors not only identify issues but solve them interactively.

### *5.11.3 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. 37] ) or a Lynx emulator such as Lynx Viewer ([LYNXVIEW] [p. 37] ) or Lynx-me ([LYNXME] [p. 37] ).
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. 37] and [HOMEPAGEREADER] [p. 37] ), a screen reader (e.g., [JAWS] [p. 37] and [WINVISION] [p. 37] ), magnification software, a small display, an onscreen keyboard, an alternative keyboard, etc.

### *5.11.4 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.

## 5.12 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. 35] ) 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.

## 5.13 Accessibility Reviewed Technologies

Checkpoints in this section: 11.1.

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@@)
- @@others? WebCGM?

@@ Do we want to recommend HTML over XHTML? what about CSS1 and CSS2? What about non-W3C specifications like PDF, Flash, etc.

## 5.14 Audio and Video

## 5.15 Audio information

Checkpoints in this section: 1.4 and 6.5.

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. 35] ) 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. 37] .

## 5.16 Visual information and motion

Checkpoints in this section: 1.3, 7.1 and 7.3.

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. 27] of a clip from "The Lion King" (available at [DVS] [p. 37] ). 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. 25] for more information about multimedia formats.

## 5.17 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.

---

# 6 Glossary

***Accessible***
: Content is accessible when it may be used by someone with a disability.

***Applet***
: A program inserted into a Web page.

***Assistive technology***
: Software or hardware that has been specifically designed to assist people with disabilities in carrying out daily activities. Assistive technology includes wheelchairs, reading machines, devices for grasping, etc. In the area of Web Accessibility, common software-based assistive technologies include screen readers, screen magnifiers, speech synthesizers, and voice input software that operate in conjunction with graphical desktop browsers (among other user agents [p. 33] ). Hardware assistive technologies include alternative keyboards and pointing devices.

***ASCII art***
: ASCII art refers to text characters and symbols that are combined to create an image. For example ";-)" is the smiley emoticon. The following is an ascii figure

showing the relationship between flash frequency and photoconvulsive response in patients with eyes open and closed [skip over ascii figure [p. 28] or consult a description of chart]:

```
  %   __ __ __ __ __ __ __ __ __ __ __ __ __ __
100 |             *                            |
 90 |                *  *                      |
 80 |          *           *                   |
 70 |             @           *                |
 60 |          @                  *            |
 50 |       *        @              *          |
 40 |                   @              *       |
 30 |    *  @              @  @           *    |
 20 |                                          |
 10 |    @                        @  @  @  @   |
      0  5 10 15 20 25 30 35 40 45 50 55 60 65 70
      Flash frequency (Hertz)
```

***Authoring tool***

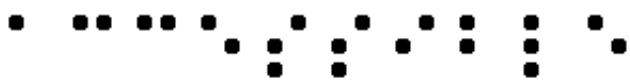
HTML editors, document conversion tools, tools that generate Web content from databases are all authoring tools. Refer to the "Authoring Tool Accessibility Guidelines" ([ATAG10] [p. 35] ) for information about developing accessible tools.

***Backward compatible***

Design that continues to work with earlier versions of a language, program, etc.

***Braille***

Braille uses six raised dots in different patterns to represent letters and numbers to be read by people who are blind with their fingertips. The word "Accessible" in braille follows:



A ***braille display,*** commonly referred to as a "dynamic braille display," raises or lowers dot patterns on command from an electronic device, usually a computer. The result is a line of braille that can change from moment to moment. Current dynamic braille displays range in size from one cell (six or eight dots) to an eighty-cell line, most having between twelve and twenty cells per line.

***Content developer***

Someone who authors Web pages or designs Web sites.

***Deprecated***

A deprecated element or attribute is one that has been outdated by newer constructs. Deprecated elements may become obsolete in future versions of HTML. The index of HTML elements and attributes in the Techniques Document indicates which elements and attributes are deprecated in HTML 4.0.
Authors should avoid using deprecated elements and attributes. User agents should continue to support for reasons of backward compatibility.

***Device independent***

Users must be able to interact with a user agent (and the document it renders) using the supported input and output devices of their choice and according to their needs. Input devices may include pointing devices, keyboards, braille devices, head wands, microphones, and others. Output devices may include monitors, speech synthesizers, and braille devices.

Please note that "device-independent support" does not mean that user agents must support every input or output device. User agents should offer redundant input and output mechanisms for those devices that are supported. For example, if a user agent supports keyboard and mouse input, users should be able to interact with all features using either the keyboard or the mouse.

***Document Content, Structure, and Presentation***

The content of a document refers to what it says to the user through natural language, images, sounds, movies, animations, etc. The structure of a document is how it is organized logically (e.g., by chapter, with an introduction and table of contents, etc.). An element [p. 29] (e.g., P, STRONG, BLOCKQUOTE in HTML) that specifies document structure is called a ***structural element***. The presentation of a document is how the document is rendered (e.g., as print, as a two-dimensional graphical presentation, as an text-only presentation, as synthesized speech, as braille, etc.) An element [p. 29] that specifies document presentation (e.g., B, FONT, CENTER) is called a ***presentation element***.

Consider a document header, for example. The content of the header is what the header says (e.g., "Sailboats"). In HTML, the header is a structural element marked up with, for example, an H2 element. Finally, the presentation of the header might be a bold block text in the margin, a centered line of text, a title spoken with a certain voice style (like an aural font), etc.

***Dynamic HTML (DHTML)***

DHTML is the marketing term applied to a mixture of standards including HTML, style sheets [p. 32] , the Document Object Model [DOM1] [p. 35] and scripting. However, there is no W3C specification that formally defines DHTML. Most guidelines may be applicable to applications using DHTML, however the following guidelines focus on issues related to scripting and style sheets: guideline 1, guideline 3, guideline 6, guideline 7, and guideline 9.

***Element***

This document uses the term "element" both in the strict SGML sense (an element is a syntactic construct) and more generally to mean a type of content (such as video or sound) or a logical construct (such as a header or list). The second sense emphasizes that a guideline inspired by HTML could easily apply to another markup language.

Note that some (SGML) elements have content that is rendered (e.g., the P, LI, or TABLE elements in HTML), some are replaced by external content (e.g., IMG), and some affect processing (e.g., STYLE and SCRIPT cause information to be processed by a style sheet or script engine). An element that causes text characters to be part of the document is called a ***text element***.

***Equivalent***

Content is "equivalent" to other content when both fulfill essentially the same function or purpose upon presentation to the user. In the context of this document, the equivalent must fulfill essentially the same function for the person with a disability (at least insofar as is feasible, given the nature of the disability and the state of technology), as the primary content does for the person without any disability. For example, the text "The Full Moon" might convey the same information as an image of a full moon when presented to users. Note that equivalent information focuses on **fulfilling the same function**. If the image is part of a link and understanding the image is crucial to guessing the link target, an equivalent must also give users an idea of the link target. Providing equivalent information for inaccessible content is one of the primary ways authors can make their documents accessible to people with disabilities.

As part of fulfilling the same function of content an equivalent may involve a description of that content (i.e., what the content looks like or sounds like). For example, in order for users to understand the information conveyed by a complex chart, authors should describe the visual information in the chart.

Since text content can be presented to the user as synthesized speech, braille, and visually-displayed text, these guidelines require ***text equivalents*** for graphic and audio information. Text equivalents must be written so that they convey all essential content. ***Non-text equivalents*** (e.g., an auditory description of a visual presentation, a video of a person telling a story using sign language as an equivalent for a written story, etc.) also improve accessibility for people who cannot access visual information or written text, including many individuals with blindness, cognitive disabilities, learning disabilities, and deafness.

Equivalent information may be provided in a number of ways, including through attributes (e.g., a text value for the "alt" attribute in HTML and SMIL), as part of element content (e.g., the OBJECT in HTML), as part of the document's prose, or via a linked document (e.g., designated by the "longdesc" attribute in HTML or a ***description link***). Depending on the complexity of the equivalent, it may be necessary to combine techniques (e.g., use "alt" for an abbreviated equivalent, useful to familiar readers, in addition to "longdesc" for a link to more complete information, useful to first-time readers). The details of how and when to provide equivalent information are part of the Techniques Document ([WCAG10-TECHS] [p. 36] ).

A ***text transcript*** is a text equivalent of audio information that includes spoken words and non-spoken sounds such as sound effects. A ***caption*** is a text transcript for the audio track of a video presentation that is synchronized with the video and audio tracks. Captions are generally rendered visually by being superimposed over the video, which benefits people who are deaf and hard-of-hearing, and anyone who cannot hear the audio (e.g., when in a crowded room). A ***collated text transcript*** combines (collates) captions with text descriptions of video information (descriptions of the actions, body language, graphics, and scene changes of the video track). These text equivalents make presentations accessible to people who are deaf-blind and to people who cannot play movies, animations, etc. It also makes the information

available to search engines.

One example of a non-text equivalent is an ***auditory description*** of the key visual elements of a presentation. The description is either a prerecorded human voice or a synthesized voice (recorded or generated on the fly). The auditory description is synchronized with the audio track of the presentation, usually during natural pauses in the audio track. Auditory descriptions include information about actions, body language, graphics, and scene changes.

***Image***

A graphical presentation.

***Image map***

An image that has been divided into regions with associated actions. Clicking on an active region causes an action to occur.

***When a user clicks on an active region of a client-side image map,*** the user agent calculates in which region the click occurred and follows the link associated with that region. ***Clicking on an active region of a server-side image map*** causes the coordinates of the click to be sent to a server, which then performs some action.

Content developers can make client-side image maps accessible by providing device-independent access to the same links associated with the image map's regions. Client-side image maps allow the user agent to provide immediate feedback as to whether or not the user's pointer is over an active region.

***Important***

Information in a document is important if understanding that information is crucial to understanding the document.

***Linearized table***

A table rendering process where the contents of the cells become a series of paragraphs (e.g., down the page) one after another. The paragraphs will occur in the same order as the cells are defined in the document source. Cells should make sense when read in order and should include structural elements [p. 29] (that create paragraphs, headers, lists, etc.) so the page makes sense after linearization.

***Link text***

The rendered text content of a link.

***Natural Language***

Spoken, written, or signed human languages such as French, Japanese, American Sign Language, and braille. The natural language of content may be indicated with the "lang" attribute in HTML ([HTML4] [p. 35] , section 8.1) and the "xml:lang" attribute in XML ([XML] [p. 36] , section 2.12).

***Navigation Mechanism***

A navigation mechanism is any means by which a user can navigate a page or site. Some typical mechanisms include:

**navigation bars**

A navigation bar is a collection of links to the most important parts of a document or site.

**site maps**

A site map provides a global view of the organization of a page or site.

**tables of contents**
A table of contents generally lists (and links to) the most important sections of a document.

***Personal Digital Assistant (PDA)***
A PDA is a small, portable computing device. Most PDAs are used to track personal data such as calendars, contacts, and electronic mail. A PDA is generally a handheld device with a small screen that allows input from various sources.

***Screen magnifier***
A software program that magnifies a portion of the screen, so that it can be more easily viewed. Screen magnifiers are used primarily by individuals with low vision.

***Screen reader***
A software program that reads the contents of the screen aloud to a user. Screen readers are used primarily by individuals who are blind. Screen readers can usually only read text that is printed, not painted, to the screen.

***Style sheets***
A style sheet is a set of statements that specify presentation of a document. Style sheets may have three different origins: they may be written by content providers, created by users, or built into user agents. In CSS ([CSS2] [p. 35] ), the interaction of content provider, user, and user agent style sheets is called the *cascade*.

***Presentation markup*** is markup that achieves a stylistic (rather than structuring) effect such as the B or I elements in HTML. Note that the STRONG and EM elements are not considered presentation markup since they convey information that is independent of a particular font style.

***Tabular information***
When tables are used to represent logical relationships among data -- text, numbers, images, etc., that information is called "tabular information" and the tables are called "data tables". The relationships expressed by a table may be rendered visually (usually on a two-dimensional grid), aurally (often preceding cells with header information), or in other formats.

***Until user agents ...***
In most of the checkpoints, content developers are asked to ensure the accessibility of their pages and sites. However, there are accessibility needs that would be more appropriately met by user agents [p. 33] (including assistive technologies [p. 27] ). As of the publication of this document, not all user agents or assistive technologies provide the accessibility control users require (e.g., some user agents may not allow users to turn off blinking content, or some screen readers may not handle tables well). Checkpoints that contain the phrase "until user agents ..." require content developers to provide additional support for accessibility until most user agents readily available to their audience include the necessary accessibility features.

**Note.** The W3C WAI Web site (refer to [WAI-UA-SUPPORT] [p. 35] ) provides information about user agent support for accessibility features. Content developers are encouraged to consult this page regularly for updated

information.

***User agent***

Software to access Web content, including desktop graphical browsers, text browsers, voice browsers, mobile phones, multimedia players, plug-ins, and some software assistive technologies used in conjunction with browsers such as screen readers, screen magnifiers, and voice recognition software.

---

# 7 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]]] compiled by the Trace R & D Center at the University of Wisconsin. That document includes a list of additional contributors.

# 8 References

For the latest version of any W3C specification please consult the list of W3C Technical Reports.

**[ATAG10]**
"Authoring Tool Accessibility Guidelines 1.0", J. Treviranus, C. McCathieNevile, I. Jacobs, and J. Richards, eds., 3 February 2000. This ATAG 1.0 Recommendation is http://www.w3.org/TR/2000/REC-ATAG10-20000203.

**[CSS1]**
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. This CSS1 Recommendation is http://www.w3.org/TR/1999/REC-CSS1-19990111. The latest version of CSS1 is available at http://www.w3.org/TR/REC-CSS1.

**[CSS2]**
"CSS, level 2 Recommendation", B. Bos, H. Wium Lie, C. Lilley, and I. Jacobs, eds., 12 May 1998. This CSS2 Recommendation is http://www.w3.org/TR/1998/REC-CSS2-19980512. The latest version of CSS2 is available at http://www.w3.org/TR/REC-CSS2.

**[DOM1]**
"Document Object Model (DOM) Level 1 Specification", V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood, eds., 1 October 1998. This DOM Level 1 Recommendation is http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001. The latest version of DOM Level 1 is available at http://www.w3.org/TR/REC-DOM-Level-1.

**[HTML32]**
"HTML 3.2 Recommendation", D. Raggett, ed., 14 January 1997. The latest version of HTML 3.2 is available at http://www.w3.org/TR/REC-html32.

**[HTML4]**
"HTML 4.01 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds., 24 December 1999. This HTML 4.01 Recommendation is http://www.w3.org/TR/1999/REC-html401-19991224.

**[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.

**[WAI-UA-SUPPORT]**
"User Agent Support for Accessibility. 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/Resources/WAI-UA-Support.

**[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.

**[WCAG10-CSS-TECHNIQUES]**
CSS Techniques for Web Content Accessibility Guidelines 1.0, W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. The latest version of this document is available at http://www.w3.org/WAI/GL/WCAG10-CSS-TECHS.

**[WCAG10-HTML-TECHNIQUES]**
HTML Techniques for Web Content Accessibility Guidelines 1.0, W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. The latest version of this document is available at http://www.w3.org/WAI/GL/WCAG10-HTML-TECHS.

**[WCAG10-NONW3C-TECHNIQUES]**
Non-W3C Technologies Techniques for Web Content Accessibility Guidelines 1.0, W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. The latest version of this document is available at http://www.w3.org/WAI/GL/WCAG10-NONW3C-TECHS.

**[WCAG10-TECHS]**
"Techniques for Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, I. Jacobs, eds. This document explains how to implement the checkpoints defined in "Web Content Accessibility Guidelines 1.0". The latest draft of the techniques is available at http://www.w3.org/WAI/GL/WCAG10-TECHS/.

**[XML]**
"Extensible Markup Language (XML) 1.0.", T. Bray, J. Paoli, C.M. Sperberg-McQueen, eds., 10 February 1998. This XML 1.0 Recommendation is: http://www.w3.org/TR/1998/REC-xml-19980210. The latest version of XML 1.0 is available at http://www.w3.org/TR/REC-xml.

# 9 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.*

## 9.1 Operating system and programming 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.

**[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.

## 9.2 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.

**[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.
**[WINVISION]**
Artic's WinVision.

## 9.3 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. 36] .
**[WAI-ER]**
The WAI Evaluation and Repair Working Group