W3C

Web Content Accessibility Guidelines 1.0

W3C Proposed Recommendation 24-Mar-1999

This version:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324
Latest version:
http://www.w3.org/TR/WAI-WEBCONTENT
Previous version:
http://www.w3.org/TR/1999/WD-WAI-PAGEAUTH-19990226
Related Documents:
Techniques for Web Content Accessibility Guidelines
List of Checkpoints for the Web Content Accessibility Guidelines 1.0
Editors:
Wendy Chisholm <chisholm@trace.wisc.edu>
Gregg Vanderheiden <gv@trace.wisc.edu>
Ian Jacobs <ij@w3.org>

Abstract

These guidelines explain how to make Web content [p. 24] accessible to people with disabilities. The guidelines are intended for all Web content developers [p. 25] (page authors and site designers) and for developers of authoring tools [p. 24] . The primary goal of these guidelines is to promote accessibility. However, following them will also make Web content more available to *all* users, whatever user agent [p. 27] they are using (e.g., desktop browser, voice browser, mobile phone, automobile-based PC, etc.) or constraints they may be operating under (e.g., noisy or noiseless surroundings, under- or over-illuminated rooms, in a hands-free environment, etc.). Following these guidelines will also help people find information on the Web more quickly. These guidelines do not discourage content developers from using images, video, etc., but rather explain how to make multimedia content more accessible to a wide audience.

This document is part of a series of accessibility guidelines published by the Web Accessibility Initiative. The series also includes User Agent Accessibility Guidelines ([WAI-USERAGENT] [p. 30]) and Authoring Tool Accessibility Guidelines ([WAI-AUTOOLS] [p. 30]).

This is a reference document for accessibility principles and design ideas. It is meant to be stable and therefore does not provide specific information about browser support for different technologies as that information changes rapidly. Instead, the Web Accessibility Initiative Web site provides such information.

Status of this document

This document is a Proposed Recommendation of the World Wide Web Consortium (W3C) and is currently undergoing review by the Members of the W3C. Review comments on this specification should be sent by 21 April 1999 to w3c-wai-gl@w3.org. An archive of public comments is available. W3C Members may send their formal comments, visible only to the W3C Team, to w3c-wai-wcag-review@w3.org.

This specification is a revision of the last call public Working Draft dated 26 February 1999. It incorporates suggestions from reviewers and further deliberations of the Web Content Guidelines Working Group. A detailed list of changes to the document is available from the Web Content Guidelines Working Group site.

The Working Group anticipates no further substantial changes to this specification. We encourage active implementation to test this specification during the Proposed Recommendation review period.

Publication as a Proposed Recommendation does not imply endorsement by the W3C membership. This is still a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite W3C Drafts 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.

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.

Related documents

The current document includes an appendix entitled "List of Checkpoints for the Web Content Accessibility Guidelines 1.0." This appendix organizes all of the checkpoints [p. 7] defined in the current document by topic and priority [p. 7] . The checkpoints in the appendix link to their definitions in the current document. The topics identified in the appendix include images, multimedia, tables, frames, forms, and scripts.

A separate document, entitled "Techniques for Web Content Accessibility Guidelines," explains how to implement these checkpoints. It discusses each checkpoint in more detail and provides examples using HTML, Cascading Style Sheets (CSS), Synchronized Multimedia Integration Language (SMIL), and MathML. **Note.** Not all browsers or multimedia tools may support the features described in the guidelines. In particular, new features of HTML 4.0 or CSS 1 or CSS 2 may not be supported. To help readers find out about support, the Techniques Document

includes:

- pointers on where to find information on browser support for the techniques.
- techniques for document validation and testing
- a techniques index of HTML elements and attributes that indicates which version of HTML defines them, which are deprecated [p. 25] in HTML 4.0, etc.
- a checkpoint map indicating where checkpoints are discussed in the Techniques Document.

The Techniques Document will be updated more regularly than the current document in order to track changes in technology.

Available formats

This document is available in the following formats:

HTML:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth.html
A plain text file:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth.txt,
HTML (Guidelines, List of Checkpoints, Techniques) as a gzip'ed tar file :
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth.tgz,
HTML (Guidelines, List of Checkpoints, Techniques) as a zip archive:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth.zip,
A PostScript file:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth.ps,
A PDF file:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth.pdf.

The definitive version of this document is the HTML version.

Table of Contents

Abstract 1
Status of this document 2
1. Introduction 5
2. Themes of Accessible Design 6
1. Ensuring Graceful Transformation 6
2. Making Content Understandable and Navigable 6
3. How the Guidelines are Organized 7
1. Document conventions 7
4. Priorities 7
5. Conformance 8
6. Web Content Accessibility Guidelines 9
1. Provide equivalent alternatives to auditory and visual content. . . .9
2. Don’t rely on color alone. 10
3. Use markup and style sheets properly. 11
4. Clarify natural language usage 12
5. Create tables that transform gracefully. 13
6. Ensure that pages featuring new technologies transform gracefully. 14
7. Ensure user control of time-sensitive content changes. . . . 15
8. Ensure direct accessibility of embedded user interfaces. . . . 16
9. Design for device-independence. 16
10. Use interim solutions. 17
11. Use W3C technologies and guidelines. 18
12. Provide context and orientation information. 19
13. Provide clear navigation mechanisms. 20
14. Ensure that documents are clear and simple. 21
Appendix A. - Validation 22
Appendix B. - Glossary 23
Acknowledgments 28
References 29

Additional appendix: List of Checkpoints for the Web Content Accessibility Guidelines 1.0.

1. Introduction

For those unfamiliar with accessibility issues pertaining to Web page design, consider that many users may be operating in contexts very different from your own:

- They may not be able to see, hear, move, or may not be able to process some types of information easily or at all.
- They may have difficulty reading or comprehending text.
- They may not have or be able to use a keyboard or mouse.
- They may have a text-only screen, a small screen, or a slow Internet connection.
- They may not speak or understand fluently the language in which the document is written.
- They may be in a situation where their eyes, ears, or hands are busy or interfered with (e.g., driving to work, working in a loud environment, etc.).
- They may have an early version of a browser, a different browser entirely, a voice browser, or a different operating system.

Content developers must consider these different situations during page design. While there are several situations to consider, each accessible design choice generally benefits several disability groups at once and the Web community as a whole. For example, by using style sheets [p. 26] to control font styles and eliminating the FONT element, HTML authors will have more control over their pages, make those pages more accessible to people with low vision, and by sharing the style sheets, will often shorten page download times for all users.

The guidelines discuss accessibility issues and provide accessible design solutions. They enumerate typical scenarios (similar to the font style example) that may pose problems for users with certain disabilities. For example, the first guideline [p. 9] explains how content developers can make images accessible. Some users may not be able to see images, others may use text-based browsers that do not support images, while others may have turned off support for images (e.g., due to a slow Internet connection). The guidelines do not suggest avoiding images as a way to improve accessibility. Instead, they explain that providing a text equivalent [p. 25] that states the purpose of the image will make it accessible.

How does a text equivalent make the image accessible? Users with blindness or low vision can understand the function of the image when the text is read aloud by a speech synthesizer. This is particularly important when the image is part of a hyperlink since, without an explanation of the link’s destination, a user with blindness wouldn’t know whether to follow it. In addition to benefitting users with disabilities, text equivalents can be used by search robots when indexing your pages.

2. Themes of Accessible Design

The guidelines address two general themes: ensuring graceful transformation, and making content understandable and navigable.

1. Ensuring Graceful Transformation

By following these guidelines, content developers can create pages that transform gracefully. Pages that transform gracefully remain accessible despite any of the constraints described in the introduction [p. 5] , including physical, sensory, and cognitive disabilities, work constraints, and technological barriers. Here are some keys to designing pages that transform gracefully:

- Separate content from structure from presentation. The appendix explains the difference between content, structure, and presentation [p. 24]).
- Provide text (including text equivalents [p. 25]). Text can be rendered in ways that are available to almost all browsing devices and accessible to almost all users.
- Create documents that work even if the user cannot see and/or hear. Some content will be sensory-specific (e.g., audio, video, applets that present visual information), so provide equivalent information in forms suited to other senses as well. **Note.** This does not mean creating an entire auditory version of a site. Screen readers [p. 26] will be able to speak all information on a page as long as it is available in text.
- Create documents that do not rely on one type of hardware. Pages should be usable by people without mice, with small screens, low resolution screens, black and white screens, no screens, with only voice or text output, etc.

The theme of graceful transformation is addressed primarily by guidelines 1 to 11.

2. Making Content Understandable and Navigable

Content developers should make content understandable and navigable. This includes not only making the language clear and simple, but also providing understandable mechanisms for navigating within and between pages. Providing navigation tools and orientation information in pages will maximize accessibility and usability. Not all users can make use of visual clues such as image maps, proportional scroll bars, side-by-side frames, or graphics that guide sighted users of graphical desktop browsers. Users also lose contextual information when they can only view a portion of a page, either because they are accessing the page one word at a time (speech synthesis or braille display [p. 24]), or one section at a time (small display, or a magnified display). Very large tables, lists, menus, etc. without orientation information may be very disorienting to users.

The theme of making content understandable and navigable is addressed primarily in guidelines 12 to 14.

3. How the Guidelines are Organized

This document includes fourteen guidelines, or general principles of accessible design. Each guideline includes:

- The guideline number.
- The statement of the guideline.
- Guideline navigation links. Three links allow navigation to the next guideline (right arrow icon), the previous guideline (left arrow icon), or the current guideline's position in the table of contents (up arrow icon).
- The rationale behind the guideline and some groups of users who benefit from it.
- A list of checkpoint definitions.

The list of checkpoint definitions in each guideline explain how the guideline applies in typical content development scenarios. Each checkpoint definition includes:

- The checkpoint number.
- The statement of the checkpoint.
- The priority of the checkpoint. Priority 1 labels are highlighted through style sheets.
- Optional informative notes, clarifying examples, and cross references to related guidelines or checkpoints.
- A link to a section of the Techniques Document where implementations and examples of the checkpoint are discussed.

Each checkpoint is specific enough so that someone reviewing a page or site may verify that the checkpoint has been satisfied.

1. Document conventions

The following editorial conventions are used throughout this document:

- Element names are in uppercase letters.
- Attribute names are quoted in lowercase letters.
- Links to definitions are rendered specially through style sheets.

4. Priorities

Each checkpoint has 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.

5. Conformance

This section defines three levels of conformance to this document:

- **Conformance Level "A"**: all Priority 1 checkpoints are satisfied;
- **Conformance Level "Double-A"**: all Priority 1 and 2 checkpoints are satisfied;
- **Conformance Level "Triple-A"**: all Priority 1, 2, and 3 checkpoints are satisfied;

Note. Conformance levels are spelled out in text so they may be understood when rendered to speech.

Claims of conformance to this document must use one of the following two forms.

Form 1: Specify:

- The guidelines title: "Web Content Accessibility Guidelines 1.0"
- The guidelines URI: http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324
- The conformance level satisfied: "A", "Double-A", or "Triple-A".
- The scope covered by the claim (e.g., page, or a defined scope of site).

Example of Form 1:

"This page conforms to W3C's "Web Content Accessibility Guidelines 1.0", available at http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324, level Double-A."

Form 2: Include, on each page claiming conformance, one of three icons provided by W3C and link the icon to the appropriate W3C explanation of the claim.

Note. *In the event this document becomes a Recommendation, by that date WAI will provide a set of three icons, for "A", "Double-A", or "Triple-A" conformance levels of "Web Content Accessibility Guidelines 1.0", together with a stable URI to the W3C Web site for linking the icons to the W3C explanation of conformance claims.*

6. Web Content Accessibility Guidelines

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

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

Although some people cannot use images, movies, sounds, applets, etc. directly, they may still use pages that include equivalent [p. 25] alternatives to the visual or auditory content. The equivalent information must serve the same purpose as the visual or auditory content. Thus, a text equivalent for an image of an upward arrow that links to a table of contents could be "Go to table of contents". In some cases, an equivalent should describe the appearance of visual content or the sound of auditory content as well (e.g., for complex charts, billboards, or diagrams).

This guideline emphasizes the importance of providing text equivalents [p. 25] of non-text media (images, pre-recorded audio, video) because text equivalents can be rendered in so many ways (e.g., braille, synthesized speech, visually-displayed text). If text equivalents are not provided for visual and auditory information, people who are blind, have low vision, who are deaf or hard of hearing, or any user who cannot or has chosen not to use visual or auditory information will not know the purpose of those elements of the page.

Providing non-text equivalents (pictures, videos, audio) of text is also beneficial to some users, especially nonreaders or people who have difficulty reading. In movies or visual presentations, visual action such as body language or other visual cues may not be accompanied by enough audio information to convey the same information. Unless verbal descriptions of this visual information are provided, people who cannot see (or look at) the visual content will not be able to perceive it.

For an audio description of a visual presentation (e.g., movie) that already has an auditory track:

- synchronize the audio description with the auditory track,
- collate [p. 24] a text version of the audio description with a text version of the auditory track (caption). The collated text will make the presentation available to people who are deaf-blind and to people who cannot play or choose not to play movies, animations, etc.

Note. The Techniques Document explains how to write text equivalents specifically for images used as image maps, spacers, bullets in lists, graphical buttons, in links, or to present mathematical equations.

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, image map regions, short animations (e.g., animated GIFs), applets, ascii art, frames, scripts, inserted list bullets, sounds (played with or without user interaction), stand-alone audio files, synthesized speech, audio tracks of video, and video. [Priority 1]

For example, in HTML:

- Use "alt" for the IMG, INPUT, and APPLET elements, or provide equivalent information in the content of the OBJECT and APPLET elements.
- For complex elements where the "alt" text does not suffice, provide an additional description using, for example, "longdesc" with IMG or FRAME, a link inside an OBJECT element, or a description link [p. 25] in the document.
- For image maps, use the "alt" attribute with AREA, or the MAP element with A elements and rich description as content.

Techniques for checkpoint 1.1

1.2 Provide redundant text links for each active region of an image map. [Priority 1 - if server-side image maps [p. 25] are used, Priority 2 - if client-side image maps [p. 25] are used. Redundant text links for client-side image maps are only required until user agents [p. 27] render text equivalents for the map links.]

Techniques for checkpoint 1.2

1.3 For each movie, provide an auditory description of the video track and synchronize it with the audio track. [Priority 1]

Techniques for checkpoint 1.3

1.4 For any time-based presentation (e.g., a movie, animation, or multimedia presentation), synchronize equivalent alternatives (e.g., captions or video descriptions) with the presentation. [Priority 1]

Techniques for checkpoint 1.4

1.5 Replace ASCII art with an image or explain it. [Priority 1 or Priority 2 depending on the importance of the information.]

Refer also to checkpoint 1.1, checkpoint 13.10, and the example of ascii art in the glossary [p. 23] .

Techniques for checkpoint 1.5

Guideline 2. Don’t rely on color alone.

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

If color alone is used to convey information, people who cannot differentiate between certain colors and users with devices that have non-color or non-visual displays will not receive the information. When foreground and background colors are too close to the same hue, they may not provide sufficient contrast when viewed using monochrome displays or by people with different types of color deficits.

Checkpoints:

2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup. [Priority 1]
Techniques for checkpoint 2.1
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].
Techniques for checkpoint 2.2

Guideline 3. Use markup and style sheets properly.

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

Using markup improperly - not according to specification - hinders accessibility. Misusing markup for a presentation effect (e.g., using a table for layout or a header to change the font size) makes it difficult for users with specialized software to understand the organization of the page or to navigate through it. Furthermore, presentation effects used alone to convey structure (e.g., constructing what looks like a table of data with an HTML PRE element) make it difficult to render a page intelligibly to other devices (refer to the description of difference between content, structure, and presentation [p. 24]).

Content developers may be tempted to use (or misuse) constructs that achieve a desired formatting effect on older browsers. They must be aware that these practices cause accessibility problems and must consider whether the formatting effect is so critical as to make the document inaccessible to some users.

At the other extreme, content developers must not sacrifice appropriate markup because a certain browser or assistive technology does not process it correctly. For example, it is appropriate to use the TABLE element in HTML to mark up tabular information. [p. 27] Doing so (and creating tables that transform gracefully (refer to guideline 5) make it possible for software to render tables other than as two-dimensional grids. Refer also to checkpoint 10.3.

Checkpoints:

3.1 When an appropriate markup language exists, use markup rather than images to convey information. [Priority 2]
For example, use MathML to mark up mathematical equations, and style sheets [p. 26] to format text and control layout. Also, avoid using images to represent text - use text and style sheets instead.
Note. If important information is conveyed in many images on the page, provide an alternative accessible page. [p. 19] Refer also to guideline 6 and guideline 11.
Techniques for checkpoint 3.1

3.2 Use header elements to convey logical structure and use them according to specification. [Priority 2]

For example, in HTML, use H2 to indicate a subsection of H1. Do not use headers for font effects.

Techniques for checkpoint 3.2

3.3 Mark up lists and list items properly. [Priority 2]

For example, in HTML, nest OL, UL, and DL lists properly.

Techniques for checkpoint 3.3

3.4 Mark up quotations. Do not use quotation markup for formatting effects such as indentation. [Priority 2]

For example, in HTML, use the Q and BLOCKQUOTE elements to markup short and longer quotations, respectively.

Techniques for checkpoint 3.4

3.5 Create documents that validate to published formal grammars. [Priority 2]

For example, include a document type declaration at the beginning of a document that refers to a published DTD.

Techniques for checkpoint 3.5

3.6 Use style sheets to control layout and presentation. [Priority 2]

For example, use the CSS 'font' property instead of the HTML FONT element to control font styles.

Techniques for checkpoint 3.6

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

For example, in CSS, use 'em' or percentage lengths rather than 'pt' or 'cm', which are absolute units. If absolute units are used, validate that the rendered content is usable (refer to the section on validation [p. 22]).

Techniques for checkpoint 3.7

3.8 Provide individual button controls in a form rather than simulating a set of buttons with an image map. [Priority 2]

Note. When a button has an image, specify text equivalent [p. 25] for the image.

Techniques for checkpoint 3.8

Guideline 4. Clarify natural language usage

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

Changes between multiple languages on the same page and abbreviations can both be indecipherable when machine-spoken or brailled unless they are identified. Content developers should identify the predominant natural language [p. 26] of a document's content and indicate when language changes occur. They should also provide expansions of abbreviations and acronyms. This information also helps search engines find key words and identify documents in a desired language. The information also improves readability of the Web for all people, including those with learning disabilities, cognitive disabilities, or people who are deaf or hard of hearing.

Checkpoints:

4.1 Clearly identify changes in the natural language of a document's text and any text equivalents [p. 25] (e.g., captions) of non-text content. [Priority 1]
For example, in HTML use the "lang" attribute. In XML, use "xml:lang". Server operators should configure their server to take advantage of the content negotiation mechanisms of the HTTP protocol so that clients can automatically retrieve documents of the preferred language.
Techniques for checkpoint 4.1
4.2 Specify the expansion of abbreviations and acronyms. [Priority 2 for the first occurrence of the acronym or abbreviation in a given document, Priority 3 thereafter.]
For example, in HTML, use the "title" attribute of the ABBR and ACRONYM elements.
Techniques for checkpoint 4.2
4.3 Identify the primary natural language of a document. [Priority 3]
For example, in HTML, set the "lang" attribute on the HTML element. In XML, use "xml:lang".
Techniques for checkpoint 4.3

Guideline 5. Create tables that transform gracefully.

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

Tables should be used to mark up truly tabular information [p. 27] ("data tables"). Content developers should avoid using them to lay out pages ("layout tables"). Tables for any use also present special problems to users of screen readers [p. 26] ([-linear-tables]).

Many user agents [p. 27] transform tables to present them and if not marked up properly, the tables will not make sense when rendered. Refer also to guideline 3.

The following checkpoints will directly benefit people who access a table through auditory means (e.g., a screen reader or an Automobile PC that operates by speech input and output) or who view only a portion of the page at a time (e.g., users with blindness or low vision using speech output or a braille display, [p. 24] or other users of devices with small displays, etc.).

Checkpoints:

5.1 For data tables, identify row and column headers. [Priority 1]
For example, in HTML, use TD to identify data cells and TH to identify headers.
Techniques for checkpoint 5.1
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]
For example, in HTML, use THEAD, TFOOT, and TBODY to group rows, COL and COLGROUP to group columns, and the "axis", "scope", and "headers"

attributes, to describe more complex relationships among data.
Techniques for checkpoint 5.2

5.3 Avoid using tables for layout. [Priority 2]
Techniques for checkpoint 5.3

5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting. [Priority 2]
For example, in HTML do not use the TH element to cause the content of a (non-table header) cell to be displayed centered and in bold.
Techniques for checkpoint 5.4

5.5 Provide summaries for tables. [Priority 3]
For example, in HTML, use the "summary" attribute of the TABLE element.
Techniques for checkpoint 5.5

5.6 Provide abbreviations for header labels. [Priority 3]
For example, in HTML, use the "abbr" attribute on the TH element.
Techniques for checkpoint 5.6

Refer also to checkpoint 10.3 and checkpoint 3.1.

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

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

Although content developers are encouraged to use new technologies that solve problems raised by existing technologies, they should know how to make their pages still work with older browsers and people who choose to turn off features.

Checkpoints:

6.1 Organize content logically using appropriate structural markup so the organization remains clear even when associated style sheets are turned off or are not supported. [Priority 1]
This makes it more likely that the document will be understood even when styles are turned off or overridden by the user.
Techniques for checkpoint 6.1

6.2 Ensure that descriptions and text alternatives for dynamic content are updated when the dynamic content changes. [Priority 1]
Techniques for checkpoint 6.2

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 mechanisms on an alternative accessible page. [p. 19] [Priority 1]
For example, in HTML provide a text equivalent [p. 25] with the NOSCRIPT element or via a server-side script. Or provide a non-text equivalent (e.g., a snapshot in place of an animation, a video equivalent of an applet, etc.). For applets and programmatic objects, provide text equivalents [p. 25] . Refer also to guideline 1.

Techniques for checkpoint 6.3

6.4 For scripts and applets, until user agents [p. 27] provide device-independent means to activate event handlers, ensure that event handlers are keyboard operable. [Priority 2]

Techniques for checkpoint 6.4

6.5 Provide an alternative presentation or page when the primary content is dynamic (e.g., when frame contents change, when scripts cause changes, etc.). [Priority 2]

For example: In HTML, use NOFRAMES at the end of each frameset, NOSCRIPT for every script, and server-side instead of client-side scripts.

Techniques for checkpoint 6.5

Refer also to checkpoint 11.4.

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

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

Some people with cognitive or visual disabilities are unable to read moving text quickly enough or at all. Movement can also cause such a distraction that the rest of the page becomes unreadable for people with cognitive disabilities. Screen readers [p. 26] are unable to read moving text. People with physical disabilities might not be able to move quickly or accurately enough to interact with moving objects.

Note. All of the following checkpoints involve some content developer responsibility until user agents [p. 27] provide adequate feature control mechanisms.

Checkpoints:

7.1 Until user agents [p. 27] allow users to control it, avoid causing the screen to flicker. [Priority 1]

Note. People with photosensitive epilepsy can have seizures 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).

Techniques for checkpoint 7.1

7.2 Until user agents [p. 27] allow users to control it, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off). [Priority 2]

Techniques for checkpoint 7.2

7.3 Until user agents [p. 27] allow users to freeze moving content, avoid movement in pages. [Priority 2]

When a page includes moving content, provide a mechanism within a script or applet to allow users to freeze motion or updates. Using style sheets with scripting to create movement allows users to turn off or override the effect more easily. Refer also to guideline 8.

Techniques for checkpoint 7.3

7.4 Until user agents [p. 27] provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2]

For example, in HTML, don't cause pages to auto-refresh with "HTTP-EQUIV=refresh" until user agents allow users to turn off the feature.

Techniques for checkpoint 7.4

7.5 Until user agents [p. 27] provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects. [Priority 2]

Techniques for checkpoint 7.5

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.

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

When an embedded object has its "own interface", the interface - like the interface to the browser itself - must be accessible. If the interface of the embedded object cannot be made accessible, an alternative accessible solution must be provided.

Note. For information about accessible interfaces, please consult the User Agent Accessibility Guidelines ([WAI-USERAGENT] [p. 30]), the Authoring Tool Accessibility Guidelines ([WAI-AUTOOL] [p. 30]), and the discussion of applets and other objects in the Techniques Document.

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. 26] and not presented elsewhere, otherwise Priority 2.]

Refer also to guideline 6.

Techniques for checkpoint 8.1

Guideline 9. Design for device-independence.

Use features that enable activation of page elements via input devices other than a pointing device (e.g., keyboard, voice, etc.).

Interaction with a document must not depend on a particular input device such as a mouse. If, for example, a form control can only be activated with a mouse or other pointing device, someone who is using the page without sight, with voice input, or with a keyboard or who is using an input device other than a mouse will not be able to use the form.

Note. Providing text equivalents for image maps or images used as links makes it possible for users to interact with them without a pointing device. Refer also to guideline 1.

Generally, pages that allow keyboard interaction are also accessible through speech input or a command line interface.

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]
Refer also to checkpoint 1.1 and checkpoint 1.2.
Techniques for checkpoint 9.1
9.2 Ensure that all elements that have their own interface are keyboard operable. [Priority 2]
Refer also to guideline 8.
Techniques for checkpoint 9.2
9.3 For scripts, specify logical event handlers rather than device-dependent event handlers. [Priority 2]
For example, in HTML use "onfocus", "onblur", and "onselect".
Techniques for checkpoint 9.3
9.4 Create a logical tab order through links, form controls, and objects. [Priority 3]
For example, in HTML, specify tab order via the "tabindex" attribute or ensure a logical page design.
Techniques for checkpoint 9.4
9.5 Provide keyboard shortcuts to important links (including those in client-side image maps [p. 25]), form controls, and groups of form controls. [Priority 3]
For example, in HTML, specify shortcuts via the "accesskey" attribute.
Techniques for checkpoint 9.5

Guideline 10. Use interim solutions.

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

For example, older browsers do not allow users to navigate to empty edit boxes. Older screen readers read lists of consecutive links as one link. These active elements are therefore difficult or impossible to access. Also, changing the current window or popping up new windows can be very disorienting to users who have available, but aren’t using, the graphical features of the desktop environment.

Note. The following checkpoints apply until user agents [p. 27] and assistive technologies [p. 23] address these issues.

Checkpoints:

10.1 Until user agents [p. 27] 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]

For example, in HTML, avoid using a frame whose target is a new window.

Techniques for checkpoint 10.1

10.2 For all form controls with implicitly associated labels, ensure that the label is properly positioned. [Priority 2]

The label must immediately precede its control on the same line (allowing more than one control/label per line) or be in the line preceding the control (with only one label and one control per line). Refer also to checkpoint 12.4.

Techniques for checkpoint 10.2

10.3 Until user agents [p. 27] or 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 2]

Note. This checkpoint benefits people with user agents [p. 27] (such as some screen readers [p. 26]) that are unable to handle blocks of text presented side-by-side; the checkpoint should not discourage content developers from using tables to represent tabular information [p. 27] .

Techniques for checkpoint 10.3

10.4 Until user agents [p. 27] handle empty controls correctly, include default, place-holding characters in edit boxes and text areas. [Priority 3]

For example, in HTML, do this for TEXTAREA and INPUT.

Techniques for checkpoint 10.4

10.5 Until user agents [p. 27] or assistive technologies render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links. [Priority 3]

Techniques for checkpoint 10.5

Guideline 11. Use W3C technologies and guidelines.

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

Many non-W3C formats (e.g., PDF, Shockwave, etc.) require viewing with either plug-ins or stand-alone applications. Often, these formats cannot be viewed or navigated with standard Web access or screen reading tools. Avoiding non-W3C and non-standard features (proprietary elements, attributes, properties, and extensions) will tend to make pages more accessible to more people using a wider variety of hardware and software.

Even when W3C technologies are used, they must be used in accordance with accessibility guidelines.

Note. Converting documents (from PDF, PostScript, RTF, etc.) to W3C markup languages (HTML, XML) does not always create an accessible document. Therefore, validate each page for accessibility and usability after the conversion process (refer to the section on validation [p. 22]). If a page does not readily convert, either revise the page until its original representation converts appropriately or provide an HTML or plain text equivalent.

Checkpoints:

11.1 Use W3C technologies and use the latest versions when they are supported. [Priority 2]
Refer to the list of references [p. 29] for information about where to find the latest W3C specifications.
Techniques for checkpoint 11.1
11.2 Avoid deprecated features of W3C technologies. [Priority 2]
For example, in HTML, don't use the deprecated [p. 25] FONT element; use style sheets instead (e.g., the 'font' property in CSS).
Techniques for checkpoint 11.2
11.3 Provide information so that users may receive documents according to their preferences (e.g., language, content type, etc.) [Priority 3]
Note. Use content negotiation where possible.
Techniques for checkpoint 11.3
11.4 If, after best efforts [p. 19] , you cannot create an accessible page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent [p. 25] information, and is updated as often as the inaccessible (original) page. [Priority 1]
Techniques for checkpoint 11.4

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; simplifying it is likely to make it more effective for all users.

Guideline 12. Provide context and orientation information.

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

Grouping and providing contextual information about the relationships between elements can be useful for all users. Complex relationships between elements on a page may be difficult for people with cognitive disabilities and people with visual disabilities to interpret.

Checkpoints:

12.1 Title each frame so that users can keep track of frames by title. [Priority 1]
For example, in HTML use the "title" attribute on FRAME elements.
Techniques for checkpoint 12.1
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]
For example, in HTML, use "longdesc," or a description link. [p. 25]
Techniques for checkpoint 12.2
12.3 Divide large blocks of information into more manageable groups where natural and appropriate. [Priority 2]
For example, in HTML, use OPTGROUP to group OPTION elements inside a SELECT; group form controls with FIELDSET and LEGEND; use nested lists where appropriate; use headings to structure documents, etc. Refer also to guideline 3.
Techniques for checkpoint 12.3
12.4 Associate labels explicitly with their controls. [Priority 2]
For example, in HTML use LABEL and its "for" attribute.
Techniques for checkpoint 12.4

Guideline 13. Provide clear navigation mechanisms.

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

Clear and consistent navigation mechanisms [p. 26] are important to people with cognitive disabilities or blindness, and benefit all users.

Checkpoints:

13.1 Clearly identify the target of each link. [Priority 2]
Link text [p. 26] should be meaningful enough to make sense when read out of context - either on its own or as part of a sequence of links. Link text should also be terse.
For example, in HTML, write "Information about version 4.3" instead of "click here". In addition to clear link text, content developers may further clarify the target of a link with an informative link title (e.g., in HTML, the "title" attribute).
Techniques for checkpoint 13.1

13.2 Provide metadata to add semantic information to pages and sites. [Priority 2]
For example, use RDF ([RDF] [p. 29]) to indicate the document's author, the type of content, and to describe the navigation mechanism, etc.
Note. Some HTML user agents [p. 27] can build navigation tools from document relations described by the HTML LINK element and "rel" or "rev" attributes (e.g., rel="next", rel="previous", rel="index", etc.). Refer also to checkpoint 13.5.
Techniques for checkpoint 13.2
13.3 Provide information about the general layout of a site (e.g., a site map, or table of contents). [Priority 2]
Also indicate what accessibility features are available and how to use them.
Techniques for checkpoint 13.3
13.4 Use navigation mechanisms in a consistent manner. [Priority 2]
Techniques for checkpoint 13.4
13.5 Provide navigation bars to highlight and give access to the navigation mechanism. [Priority 3]
Techniques for checkpoint 13.5
13.6 Group related links, identify the group (for user agents), and, until user agents [p. 27] do so, provide a way to bypass the group. [Priority 3]
Techniques for checkpoint 13.6
13.7 Enable different types of searches for different skill levels and preferences. [Priority 3]
Techniques for checkpoint 13.7
13.8 Place distinguishing information at the beginning of headings, paragraphs, lists, etc. [Priority 3]
Note. This is commonly referred to as "front-loading" and is especially helpful for people accessing information with serial devices such as speech synthesizers.
Techniques for checkpoint 13.8
13.9 Provide information about document collections (i.e., documents comprising multiple pages.). [Priority 3]
For example, in HTML describe 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, gzip, stuffit, etc.) of the multiple pages.
Note. The performance improvement gained by offline processing can make browsing much less expensive for people with disabilities who may be browsing slowly.
Techniques for checkpoint 13.9
13.10 Provide a means to skip over multi-line ASCII art. [Priority 3]
Refer also to checkpoint 1.5.
Techniques for checkpoint 13.10

Guideline 14. Ensure that documents are clear and simple.

Ensure that documents are clear and simple to promote comprehension.

Consistent page layout, recognizable graphics, and easy to understand language benefit all users. In particular, they help people with cognitive disabilities or who have difficulty reading. (However, ensure that images have text equivalents for people who are blind, have low vision, or for any user who cannot or has chosen not to view graphics. Refer also to guideline 1.)

Using clear and simple language promotes effective communication. Access to written information can be difficult to impossible for people who have cognitive disabilities, learning disabilities, or who are deaf or hard of hearing. This consideration also applies to the many people whose first language differs from your own.

Checkpoints:

14.1 Use the clearest and simplest language appropriate for a site's content. [Priority 1]
Techniques for checkpoint 14.1
14.2 Provide visual or auditory equivalents to text where they facilitate comprehension of the page. [Priority 3]
Note. For example, provide a video representation of an applet, an audio description of video, recorded spoken text, etc. These equivalent presentations are particularly important for non-readers. This checkpoint does not mean that text content should be removed. Refer also to guideline 1.
Techniques for checkpoint 14.2
14.3 Create a style of presentation that is consistent across pages. [Priority 3]
Techniques for checkpoint 14.3

Appendix A. - Validation

Validate accessibility with automatic tools and human review. Automated methods are generally rapid and convenient but cannot identify all accessibility issues. Human review can help ensure clarity of language and ease of navigation.

Begin using validation methods at the earliest stages of development. Accessibility issues identified early are easier to correct and avoid.

Following are some important validation methods, discussed in more detail in the section on validation in the Techniques Document.

1. Use an automated accessibility tool and browser validation tool. Please not that software tools do not address all accessibility issues, such as the meaningfulness of link text, the applicability of a text equivalent [p. 25] , etc.
2. Validate the HTML.
3. Validate the style sheets.

4. Use a text-only browser or emulator.
5. Use multiple graphic browsers, with:
   - sounds and graphics loaded,
   - graphics not loaded,
   - sounds not loaded,
   - no mouse,
   - frames, scripts, style sheets, and applets not loaded
6. Use several browsers, old and new.
7. Use a self-voicing browser, a screen reader, magnification software, a small display, etc.
8. Use spell and grammar checkers. 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. Eliminating grammar problems increases comprehension.
9. Review content and structure [p. 24] for clarity and simplicity. Readability statistics, such as those generated by some word processors may be useful indicators of clarity and simplicity. Better still, ask an experienced (human) editor to review written content for clarity. Editors can also improve the usability of documents by identifying intercultural problems that might arise due to language or icon usage.
10. Invite people with disabilities to review your documents. Expert and novice users with disabilities will provide valuable feedback about accessibility or usability problems and their severity.

Appendix B. - 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 that provides a specialized service and generally operates in conjunction with an independent user agent [p. 27] . Assistive technologies include screen readers, screen magnifiers, and voice input software.

ASCII art
ASCII art refers to text characters and symbols that are combined to create an image. For example ";-)" is the smiley emoticon and the following drawing represents a cow [skip ascii cow [p. 24]]:

```
          (__)
          (oo)
    /-------\/
   / |     ||
  *  ||----||
```

Authoring tool

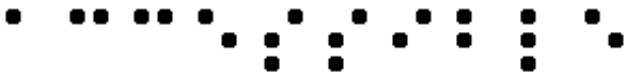
HTML editors, document conversion tools, tools that generate Web content from databases. Refer to the "Authoring Tool Accessibility Guidelines" ([WAI-AUTOOLS] [p. 30]) for information about developing accessible tools.

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. Dynamic braille displays range in size from one cell (six or eight dots) to an eighty cell line. Displays with twelve to twenty cells per line are the most common.

Caption

When a text equivalent [p. 25] for video is synchronized with the video presentation it is called a *caption*. Captions are used by people who cannot hear the audio track of the video material. Without captions, people who are deaf, or hard of hearing, or any user who cannot or has chosen not to hear sound cannot perceive the information presented through speech, sound effects, music, etc.

Collate

When referring to text transcripts [p. 25] , collating means combining the text version of the descriptions and the text transcript (captions) of the primary audio track into a single document to read like a script of the movie. In other words, the two documents are not combined but flow as a single document.

Content/Structure/Presentation

The content of a document refers to both what the document says and the bytes (text and markup) that make it up. The structure of a document is how it is organized logically (e.g., by chapter, with an introduction and table of contents, etc.) 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.

Consider a header, for example. The content of the header is what the header says (e.g., "Sailboats") and how it is marked up (e.g., with an H2 element in HTML). In terms of structure, the header may be part of a chapter of the document. 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.

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 techniques 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.

Dynamic HTML (DHTML)

DHTML is the marketing term applied to a mixture of standards including HTML, style sheets [p. 26] , the Document Object Model [DOM1] [p. 29] 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 6, and guideline 7.

Equivalent

An "equivalent" is information that, when presented to a user, provides essentially the same function (or fulfills the same purpose) as another piece of information. Due to the importance of text for a variety of output devices, we distinguish ***text equivalents*** from ***non-text equivalents***. For example, for an audio clip of a conversation between two people, a ***text-transcript*** is a text equivalent of the audio clip. Examples of non-text equivalents include spoken audio recordings of text, icons, and videos of sign language renderings of content. In the context of this document, text and non-text equivalents are generally intended to serve as alternatives for portions of the primary content that are inaccessible to people with disabilities.

Part of providing equivalent information may involve describing the appearance (visual or audio) of an image, table, chart, or other complex object. Descriptive information may be part of the equivalent or may be separated from it (often the case when the description is lengthy). Descriptive information may even be external to the document, available through a link (e.g., via the "longdesc" attribute in HTML or via a *description links*, or d-links).

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 is important if understanding it in detail is necessary for the overall understanding of a document.

Link text

The rendered text content of a link.

Natural Language

National spoken or written languages such as French, English, and Japanese. The natural language of content may be indicated with the "lang" attribute in HTML ([HTML40] [p. 29] , section 8.1) and the "xml:lang" attribute in XML ([XML] [p. 30] , 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. 29]), 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 a user agent [p. 27] or assistive technology [p. 23] . Unfortunately, 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). Therefore, to address cases where the responsibility for providing accessible control lies with the user agent or assistive technology but that control is not yet easily or widely available, certain checkpoints contain the phrase "until user agents ...". When content developers see this phrase, they should recognize that they are being asked to provide additional support for accessibility until most user agents are capable of providing the necessary control.

For each checkpoint containing this phrase, content developers must consider:

1. What is their anticipated audience? For example, designing for a company intranet where everyone uses the same browser is different than designing for the World Wide Web.
2. Do those users have tools available that satisfy the demands of the checkpoint? If the answer to this question is yes, content developers are not required to satisfy the checkpoint.

How will content developers know when most user agents or assistive technologies meet specific needs? The W3C WAI will make information about support for accessibility features available from its Web site (either directly or by providing links to the information). Content developers are encouraged to consult this information regularly for pertinent updates.

User agent

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

Acknowledgments

Web Content Guidelines Working Group Co-Chairs:

Chuck Letourneau, Starling Access Services

Gregg Vanderheiden, Trace Research and Development

W3C Team contacts:

Judy Brewer and Daniel Dardailler

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, 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, Jaap van Lelieveld, and Jason White

The original draft of this document is based on "The Unified Web Site Accessibility Guidelines" ([UWSAG] [p. 30]) compiled by the Trace R & D Center at the University of Wisconsin. That document includes a list of additional contributors.

References

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

[CSS1]
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. The CSS1 Recommendation is available at: 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. The CSS2 Recommendation is available at: 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. The DOM Level 1 Recommendation is available at: 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

[HTML40]
"HTML 4.0 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds., 17 December 1997, revised 24 April 1998. The HTML 4.0 Recommendation is available at: http://www.w3.org/TR/1998/REC-html40-19980424.
The latest version of HTML 4.0 is available at: http://www.w3.org/TR/REC-html40.

[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.

[MATHML]
"Mathematical Markup Language", P. Ion and R. Miner, eds., 7 April 1998. The MathML 1.0 Recommendation is available at: http://www.w3.org/TR/1998/REC-MathML-19980407.
The latest version of MathML 1.0 is available at: http://www.w3.org/TRREC-MathML.

[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.

[RDF]
"Resource Description Framework (RDF) Model and Syntax Specification", O. Lassila, R. Swick, eds., 22 February 1999. The RDF Recommendation is available at: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

The latest version of RDF 1.0 is available at:
http://www.w3.org/TR/REC-rdf-syntax

[SMIL]

"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", P. Hoschka, ed., 15 June 1998. The SMIL 1.0 Recommendation is available at: 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-AUTOOLS]

"Authoring Tool Accessibility Guidelines", J. Treviranus, J. Richards, I. Jacobs, C. McCathieNevile, eds. The latest Working Draft of these guidelines for designing accessible authoring tools is available at:
http://www.w3.org/TR/WD-WAI-AUTOOLS/

[WAI-USERAGENT]

"User Agent Accessibility Guidelines", J. Gunderson and I. Jacobs, eds. The latest Working Draft of these guidelines for designing accessible user agents is available at: http://www.w3.org/TR/WD-WAI-USERAGENT/

[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. This document is available at:
http://www.tracecenter.org/docs/html_guidelines/version8.htm

[XML]

"Extensible Markup Language (XML) 1.0.", T. Bray, J. Paoli, C.M. Sperberg-McQueen, eds., 10 February 1998. The XML 1.0 Recommendation is available at: 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

List of Checkpoints for the Web Content Accessibility Guidelines 1.0

W3C Proposed Recommendation 24-Mar-1999

This version:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/full-checklist
Latest version:
http://www.w3.org/TR/WAI-WEBCONTENT/full-checklist
Previous version:
http://www.w3.org/TR/1999/WD-WAI-PAGEAUTH-19990226/full-checklist
Related Documents:
Web Content Accessibility Guidelines 1.0
Techniques for Web Content Accessibility Guidelines
Editors:
Wendy Chisholm <chisholm@trace.wisc.edu>
Gregg Vanderheiden <gv@trace.wisc.edu>
Ian Jacobs <ij@w3.org>

Abstract

This document is an appendix to the W3C "Web Content Accessibility Guidelines 1.0". It provides a list of all checkpoints from the Web Content Accessibility Guidelines 1.0, organized by concept, as a checklist for Web content developers. Please refer to the Guidelines document for introductory information, information about related documents, a glossary of terms, and more.

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

Status of this document

This document is a Proposed Recommendation of the World Wide Web

Consortium (W3C) and is currently undergoing review by the Members of the W3C. Review comments on this specification should be sent by 21 April 1999 to w3c-wai-gl@w3.org. An archive of public comments is available. W3C Members may send their formal comments, visible only to the W3C Team, to w3c-wai-wcag-review@w3.org.

This specification is a revision of the last call public Working Draft dated 26 February 1999. It incorporates suggestions from reviewers and further deliberations of the Web Content Guidelines Working Group. A detailed list of changes to the document is available from the Web Content Guidelines Working Group site.

Publication as a Proposed Recommendation does not imply endorsement by the W3C membership. This is still a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite W3C Drafts 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.

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

Priorities

Each checkpoint has 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.

Priority 1 checkpoints

In General (Priority 1)	Yes	No	N/A
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, image map regions, short animations (e.g., animated GIFs), applets, ascii art, frames, scripts, inserted list bullets, sounds (played with or without user interaction), stand-alone audio files, synthesized speech, audio tracks of video, and video.			
2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup.			
4.1 Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions) of non-text content.			
6.1 Organize content logically using appropriate structural markup so the organization remains clear even when associated style sheets are turned off or are not supported.			
7.1 Until user agents allow users to control it, avoid causing the screen to flicker.			
14.1 Use the clearest and simplest language appropriate for a site's content.			
And if you use images and image maps (Priority 1)	**Yes**	**No**	**N/A**
1.2 Provide redundant text links for each active region of an image map. [Priority 1 - if server-side image maps are used, Priority 2 - if client-side image maps are used. Redundant text links for client-side image maps are only required until user agents render text equivalents for the map links.]			
1.5 Replace ASCII art with an image or explain it. [Priority 1 or Priority 2 depending on the importance of the information.]			
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.			
And if you use tables (Priority 1)	**Yes**	**No**	**N/A**
5.1 For data tables, identify row and column headers.			
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.			
And if you use frames (Priority 1)	**Yes**	**No**	**N/A**
6.2 Ensure that descriptions and text alternatives for dynamic content are updated when the dynamic content changes.			

12.1 Title each frame so that users can keep track of frames by title.			
And if you use applets and scripts (Priority 1)	**Yes**	**No**	**N/A**
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 mechanisms on an alternative accessible page.			
And if you use multimedia (Priority 1)	**Yes**	**No**	**N/A**
1.3 For each movie, provide an auditory description of the video track and synchronize it with the audio track.			
1.4 For any time-based presentation (e.g., a movie, animation, or multimedia presentation), synchronize equivalent alternatives (e.g., captions or video descriptions) with the presentation.			
And if all else fails (Priority 1)	**Yes**	**No**	**N/A**
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, and is updated as often as the inaccessible (original) page.			

Priority 2 checkpoints

In General (Priority 2)	**Yes**	**No**	**N/A**
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].			
3.1 When an appropriate markup language exists, use markup rather than images to convey information.			
3.2 Use header elements to convey logical structure and use them according to specification.			
3.3 Mark up lists and list items properly.			
3.4 Mark up quotations. Do not use quotation markup for formatting effects such as indentation.			
3.5 Create documents that validate to published formal grammars.			
3.6 Use style sheets to control layout and presentation.			
3.7 Use relative rather than absolute units in markup language attribute values and style sheet property values.			
4.2 Specify the expansion of abbreviations and acronyms. [Priority 2 for the first occurrence of the acronym or abbreviation in a given document, Priority 3 thereafter.]			
7.2 Until user agents allow users to control it, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off).			

7.4 Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages.			
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.			
11.1 Use W3C technologies and use the latest versions when they are supported.			
11.2 Avoid deprecated features of W3C technologies.			
12.3 Divide large blocks of information into more manageable groups where natural and appropriate.			
13.1 Clearly identify the target of each link.			
13.2 Provide metadata to add semantic information to pages and sites.			
13.3 Provide information about the general layout of a site (e.g., a site map, or table of contents).			
13.4 Use navigation mechanisms in a consistent manner.			
And if you use images and image maps (Priority 2)	**Yes**	**No**	**N/A**
3.8 Provide individual button controls in a form rather than simulating a set of buttons with an image map.			
And if you use tables (Priority 2)	**Yes**	**No**	**N/A**
5.3 Avoid using tables for layout.			
5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting.			
10.3 Until user agents or 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.			
And if you use frames (Priority 2)	**Yes**	**No**	**N/A**
6.5 Provide an alternative presentation or page when the primary content is dynamic (e.g., when frame contents change, when scripts cause changes, etc.).			
12.2 Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone.			
And if you use forms (Priority 2)	**Yes**	**No**	**N/A**
10.2 For all form controls with implicitly associated labels, ensure that the label is properly positioned.			
12.4 Associate labels explicitly with their controls.			
And if you use applets and scripts (Priority 2)	**Yes**	**No**	**N/A**
6.4 For scripts and applets, until user agents provide device-independent means to activate event handlers, ensure that event handlers are keyboard operable.			
7.3 Until user agents allow users to freeze moving content, avoid movement in pages.			
8.1 Make programmatic elements such as scripts and			

applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important and not presented elsewhere, otherwise Priority 2.]			
9.2 Ensure that all elements that have their own interface are keyboard operable.			
9.3 For scripts, specify logical event handlers rather than device-dependent event handlers.			
10.1 Until user agents 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 3 checkpoints

In General (Priority 3)	Yes	No	N/A
4.3 Identify the primary natural language of a document.			
9.4 Create a logical tab order through links, form controls, and objects.			
9.5 Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls.			
11.3 Provide information so that users may receive documents according to their preferences (e.g., language, content type, etc.)			
13.5 Provide navigation bars to highlight and give access to the navigation mechanism.			
13.6 Group related links, identify the group (for user agents), and, until user agents do so, provide a way to bypass the group.			
13.7 Enable different types of searches for different skill levels and preferences.			
13.8 Place distinguishing information at the beginning of headings, paragraphs, lists, etc.			
13.9 Provide information about document collections (i.e., documents comprising multiple pages.).			
13.10 Provide a means to skip over multi-line ASCII art.			
14.2 Provide visual or auditory equivalents to text where they facilitate comprehension of the page.			
14.3 Create a style of presentation that is consistent across pages.			
And if you use tables (Priority 3)	**Yes**	**No**	**N/A**
5.5 Provide summaries for tables.			
5.6 Provide abbreviations for header labels.			
And if you use forms (Priority 3)	**Yes**	**No**	**N/A**

10.4 Until user agents handle empty controls correctly, include default, place-holding characters in edit boxes and text areas.			
10.5 Until user agents or assistive technologies render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links.			

[guidelines] [techniques]

W3C

Techniques for Web Content Accessibility Guidelines

W3C Working Draft 24-Mar-1999

This version:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth-tech
Latest version:
http://www.w3.org/TR/WAI-WEBCONTENT/wai-pageauth-tech
Previous version:
http://www.w3.org/TR/1999/WD-WAI-PAGEAUTH-19990226/wai-pageauth-tech
Related Documents:
Web Content Accessibility Guidelines 1.0
List of Checkpoints for the Web Content Accessibility Guidelines 1.0
Editors:
Wendy Chisholm <chisholm@trace.wisc.edu>
Gregg Vanderheiden <gv@trace.wisc.edu>
Ian Jacobs <ij@w3.org>

Abstract

This document is a list of techniques that implement the guidelines described in "Web Content Accessibility Guidelines 1.0". This document includes primarily techniques that Web content developers may use to implement the guidelines, but also refers to some CSS techniques as well.

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

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 W3C members and other interested parties. It 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". This is work in progress

and does not imply endorsement by, or the consensus of, either W3C or members of the Web Content Guidelines Working Group.

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.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Please send detailed comments on this document to w3c-wai-gl@w3.org.

Available formats

This document is available in the following formats:

HTML:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth-tech.html
A plain text file:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth-tech.txt,
HTML as a gzip'ed tar file:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth.tgz,
HTML as a zip file (this is a '.zip' file not an '.exe'):
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth.zip,
A PostScript file:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth-tech.ps,
A PDF file:
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth-tech.pdf.

In case of a discrepancy between the various formats of the specification, http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324/wai-pageauth-tech.html is considered the definitive version.

Table of Contents

1 Accessibility Themes for HTML 4
1.1 Structure vs. Presentation 4
1.2 Text equivalents and descriptions 7
1.3 Alternative pages 9
1.4 Keyboard access 10
1.5 Navigation 11
1.6 Comprehension 12
1.7 Content negotiation 14
1.8 Automatic page refresh 14
1.9 Other topics 15
1.10 Validation 15
1.11 Browser Support 16
2 HTML topics 16
2.1 Document structure and metadata 16
2.2 Language information 18
2.3 Text markup 18
2.4 Lists 19
2.5 Tables 22
2.6 Links 27
2.7 Images and image maps 28
2.8 Applets and other objects 34
2.9 Audio and video 36
2.10 Style and style sheets 38
2.11 Frames 44
2.12 Forms 48
2.13 Scripts 50
3 Acknowledgments 51
4 Reference specifications 53
5 Services 53
6 Checkpoint Map 54
7 Index of HTML elements and attributes 58
7.1 Elements 58
7.2 Attributes 62

Priorities

Each checkpoint has 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.

The checkpoints in this document are numbered to match their numbering in Web Content Accessibility Guidelines 1.0.

---

1 Accessibility Themes for HTML

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

1.1 Structure vs. Presentation

When designing a document or series of documents, content developers should strive to identify the desired structure for their documents without 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 scalability.

Identifying what is structure and what is presentation may be a challenging task at times that content developers must develop a mindset for recognizing. 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 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.) 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, even though the BLOCKQUOTE element may cause indented text in some browsers, that is not its meaning, only 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 next two sections identify (by theme) precisely those elements and attributes considered structural (and some of their uses) and those that are considered to control presentation. The section on style and style sheets [p. 38] discusses how to use CSS to accomplish the same tasks as the HTML presentation elements and attributes.

Elements and attributes that are deprecated in HTML 4.0 ([HTML40] [p. 52]) appear in red and followed by an asterisk (*) in this document. Most presentation elements have been deprecated in HTML 4.0.

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

1.1.1 Structural elements and attributes

Document head and body	HTML [p. 60] , HEAD [p. 60] , BODY [p. 59]
Data about the document	TITLE [p. 62] , META [p. 61] , ADDRESS [p. 59]
Attributes	alt, title, longdesc
Chapters, sections, etc.	H1 [p. 60] -H6
Author-defined structures	DIV [p. 60] , SPAN [p. 61]
Attributes	class, id
Language, writing direction	BDO [p. 59]
Attributes	lang, hreflang, dir
Creating paragraphs	EM [p. 60] , STRONG [p. 61]
Acronyms and abbreviations	ACRONYM [p. 59] , ABBR [p. 59]
Subscripts and superscripts	SUB [p. 61] , SUP [p. 61]

Inserted and deleted text	INS [p. 60] , DEL [p. 59]
Attributes	cite, datetime
Quotations	BLOCKQUOTE [p. 59] , Q [p. 61]
Attributes	cite
Identifying chunks of text	DFN [p. 59] , CODE [p. 59] , SAMP [p. 61] , KBD [p. 60] , VAR [p. 62] , CITE [p. 59]
Lists	UL [p. 62] , OL [p. 61] , LI [p. 60] , DL [p. 60] , DT [p. 60] , DD [p. 59] , DIR* [p. 60] ,
Attributes	start*, value*
Tables	TABLE [p. 61] , CAPTION [p. 59] , THEAD [p. 62] , TFOOT [p. 62] , TBODY [p. 61] , COLGROUP [p. 59] , COL [p. 59] , TR [p. 62] , TH [p. 62] , TD [p. 62]
Attributes	abbr, axis, headers, scope, rowspan, span, summary
Links	LINK [p. 60] , IMG [p. 60] , OBJECT [p. 61] , PARAM [p. 61] , APPLET* [p. 59] , MAP [p. 60] , AREA [p. 59]
Attributes	shape, ismap, coords
Forms and keyboard control	FORM [p. 60] , INPUT [p. 60] , BUTTON [p. 59] , SELECT [p. 61] , OPTGROUP [p. 61] , OPTION [p. 61] , TEXTAREA [p. 62] , LABEL [p. 60] , FIELDSET [p. 60] , LEGEND [p. 60] , ISINDEX* [p. 60]
Attributes	tabindex, accesskey, label, for
Scripts	SCRIPT [p. 61] , NOSCRIPT [p. 61]
Attributes	onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onload, onload, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onreset, onselect, onsubmit, onunload

1.1.2 Presentation elements and attributes

Note 1. Not all attributes are listed in this section. Only those deemed pertinent to accessibility have been listed.

Note 2. While the STYLE element is presentational, it may be used to *enhance* the presentation of structural elements. The other presentation elements are often used *instead of* structural elements.

Style sheets	PRE [p. 61] , CENTER* [p. 59] , BR [p. 59] , STYLE [p. 61]
Attributes	align*, valign*, clear*, nowrap*, char, charoff , style
Spacing	
Attributes	hspace*, vspace*, cellpadding, cellspacing, compact*, type*
Text style and fonts	TT [p. 62] , I [p. 60] , B [p. 59] , BIG [p. 59] , SMALL [p. 61] , STRIKE* [p. 61] , S* [p. 61] , U* [p. 62] , FONT* [p. 60] , BASEFONT* [p. 59]
Attributes	face*, size*
Colors	
Attributes	background*, bgcolor*, color*, text*, link*, alink*, vlink*
Rules and borders	HR [p. 60]
Attributes	border, noshade*, rules, size (deprecated according to element).
Frames	FRAMESET [p. 60] , FRAME [p. 60] , IFRAME [p. 60] , NOFRAMES [p. 61]
Attributes	target, marginheight, marginwidth, frame, frameborder, noresize, rows, cols, scrolling

1.2 Text equivalents and descriptions

Text is considered accessible to almost all users since it may be handled by screen readers, non-visual browsers, braille readers, etc. It is good practice, as you design a document containing non-textual information (images, graphics, applets, sounds, etc.) to think about supplementing that information with textual equivalents wherever possible.

There are several types of textual supplements to consider:

Text equivalents

A text equivalent (or alternative text) describes the function or purpose of content. A text equivalent should not describe visual appearance or how something sounds. For example, if an image of a magnifying glass is used for a search button, the alt-text would be "Search" rather than "Magnifying glass".

Text equivalents should 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 layout 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?

Related checkpoints:

1. Refer also to checkpoint 1.1.
2. For all image map links:
   - Refer also to checkpoint 1.2.
3. Refer also to checkpoint 3.8.
4. Refer also to checkpoint 1.5.
5. Refer also to checkpoint 13.10.
6. Refer also to checkpoint 3.6.
7. [Checkpoint 12.1] Title each frame so that users can keep track of frames by title. [Priority 1]

Long descriptions

In general, a long description should describe visual appearance or how something sounds. Long descriptions are generally stored in external documents (but may be inline if necessary).

1. [Checkpoint 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]

HTML allows content developers to specify these substitutes in several ways. Here is a summary of attributes used for alt-text and long descriptions and the elements they apply to (by version of HTML):

To implement alt-text, use the "alt" attribute.

In HTML 4.0, applies to: IMG, AREA, INPUT, APPLET. The attribute is mandatory for the IMG and AREA elements.
In HTML 3.2, applies to: IMG, AREA, APPLET.
In HTML 2.0, applies to: IMG.

To implement long descriptions, use the "longdesc" attribute

In HTML 4.0, applies to: FRAME, IFRAME, IMG
In HTML 3.2, applies to: N/A.
In HTML 2.0, applies to: N/A.

Not every user agent supports these attributes. When required to design documents for a version of HTML known not to support one of these attributes, content developers should implement alt-text or descriptions in other ways. Methods include:

- Inline text equivalents or descriptions. For example, include a description of the image immediately after the image.
- Links to descriptions (called description links or "d-links"). Description links should be ordinary links (on the same page or another designed for such links) that designate a document containing a long description. The link text (content of the A element) should be "[D]".

Each of the HTML topics [p. 16] below describes prioritized scenarios for alt-text and descriptions.

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. 52]) content developers may also supply information there as well.

1.3 Alternative pages

[Checkpoint 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, and is updated as often as the inaccessible (original) page. [Priority 1]

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

- Allow users to navigate to a separate page that is accessible and maintained with the same frequency as the inaccessible original page. There are several 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.
  2. Use the LINK element to designate alternative documents. Browsers should load the alternative page automatically based on the user's browser type and preferences.
- Instead of static alternative pages, set up server-side scripts that generate accessible versions of a page on demand.
- [Checkpoint 6.5] Provide an alternative presentation or page when the primary content is dynamic (e.g., when frame contents change, when scripts cause changes, etc.). [Priority 2] . See the examples for Frames [p. 45] and Scripts [p. 50] .
- Otherwise provide a phone number, fax number, e-mail, or postal address where information is available and accessible, preferably 24 hours a day

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.html"
      media="aural, braille, tty">
</HEAD>
<BODY><P>...</BODY>
```

End example.

1.4 Keyboard access

Not every user has a graphic environment with a mouse or other pointing device. Some users rely on keyboard or voice input to navigate links, activate form controls, etc. Content developers should always 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 two ways:

Keyboard shortcuts
: Keyboard shortcuts allow users to combine keystrokes to navigate links or form controls on a page. **Note.** Keyboard shortcuts may be handled differently by different operating systems.

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).

Here is a summary of attributes used for keyboard access and tabbing order and the elements they apply to (by version of HTML):

To implement shortcuts, use the "accesskey" attribute.
- In HTML, 4.0 applies to: A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA
- In HTML, 3.2 applies to: N/A.
- In HTML, 2.0 applies to: N/A.

To implement tabbing order, use the "tabindex" attribute
- In HTML, 4.0 applies to: A, AREA, BUTTON, INPUT, OBJECT, SELECT, TEXTAREA
- In HTML, 3.2 applies to: N/A
- In HTML, 2.0 applies to: N/A

In the following example, if the accesskey "C" is activated,"doc.html" is retrieved by the browser:

Example.

```
<A accesskey="C" href="doc.html" hreflang="en"
   title="XYZ company home page">
      XYZ company home page</A>
```

End example.

The next example assigns "U" as the access key. Typing "U" gives focus to the label, which in turn gives focus to the input control, so that the user can input text.

Example.

```
<FORM action="submit" method="post">
<P>
      <LABEL for="user" accesskey="U">name</LABEL>
      <INPUT type="text" id="user">
</FORM>
```

End example.

In the next example, we specify a tabbing order among elements (in order, "field2", "field1", "submit"):

Example.

```
<FORM action="submit" method="post">
<P>
<INPUT tabindex="2" type="text" name="field1">
<INPUT tabindex="1" type="text" name="field2">
<INPUT tabindex="3" type="submit" name="submit">
</FORM>
```

End example.

[Checkpoint 9.2] Ensure that all elements that have their own interface are keyboard operable. [Priority 2] Some elements import objects whose interfaces cannot be controlled through HTML (applets, images, etc.) In such cases, users should ensure that the imported objects themselves provide accessible interfaces.

1.5 Navigation

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

A consistent style of presentation on each page allows users to easily find navigation buttons between pages, as well as find the primary content for each page. While this helps make it easier for everyone, it especially benefits people with learning and reading disabilities. Making it easy to predict where the needed information is found on each page will increase the likelihood that it will be found.

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

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

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

A navigation structure is the set of possible paths available for a user to take through the content on your site. Providing navigation bars, site maps, and search features all increase the likelihood that a user will navigate easily to the information that they seek on your site. If your site is highly visual in nature, the structure might be harder to navigate through if the user can’t form a mental map of where they are

going or where they have been. Therefore, provide a description that will help users discover the mechanisms you have provided.

- [Checkpoint 13.5] Provide navigation bars to highlight and give access to the navigation mechanism. [Priority 3]
- [Checkpoint 13.3] Provide information about the general layout of a site (e.g., a site map, or table of contents). [Priority 2]
- [Checkpoint 13.7] Enable different types of searches for different skill levels and preferences. [Priority 3]
- [Checkpoint 13.2] Provide metadata to add semantic information to pages and sites. [Priority 2] . See [RDF] [p. 52] .

  Content developers should use link types (reference to HTML 4.0, 6.12) with LINK so that user agents may use build navigation structures. (Provide example with "next", "prev", "content", and "start").

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

See also the section on links [p. 27] .

1.6 Comprehension

1. [Checkpoint 14.1] Use the clearest and simplest language appropriate for a site's content. [Priority 1] Provide a simplified text equivalent as an alternative to the primary text. This may be helpful to non-readers and people who have difficulty reading. Non-readers can hear the simplified text using speech output technology. Individuals with reading difficulties may be able to read the simplified text. Because of the challenges of maintaining separate texts, it is preferable to simplify only the main text and not provide a simplified text equivalent.
2. [Checkpoint 14.2] Provide visual or auditory equivalents to text where they facilitate comprehension of the page. [Priority 3] Visual, non-text equivalents may include, for example graphics, animations, or videos. These are especially helpful for non-readers who can perceive visual presentations. For example, sighted deaf non-readers may benefit from video equivalents in manual communication (sign language). Non-readers, whether they have disability or not, may also benefit from highly graphical equivalents.

   Non-visual, non-text equivalents are very diverse. Among the most common are pre-recorded audio of music, spoken language, or sound effects. Such equivalents would be especially important for non-readers who can perceive audio presentations. Presentations in the audio medium of synthesized speech and the tactile medium of braille are usually derived from text or text equivalents so usually require no additional work from the developer.

Follow these writing suggestions:

-
- Strive for clear and accurate headings and link descriptions. Scrutinize every heading, outline, and menu to see if the crucial words mean exactly what is intended, and if there are more common words that would convey the same meaning.
- State the topic of the sentence or paragraph at the beginning of the sentence or paragraph.
- Limit each paragraph to one main idea.
- Avoid idiomatic language, technical jargon, and other unfamiliar vocabulary and expressions.
- Avoid specialized meanings of familiar vocabulary, unless explanations are provided.
- Avoid the passive voice.
- Avoid complex sentence structures.
- Make link phrases terse yet meaningful enough so they make sense when read out of context, alone or as part of a series of links.

Because people tend to scan rather than read Web pages, the quality of headings is particularly important. Good headings will at least get people to a section that has the information they need. From there they can go to a dictionary or even print out a section and ask for help.

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

Grouping mechanisms in HTML 4.0 include:

- Use FIELDSET to group form controls into semantic units [p. 48] and describe the group with the LEGEND element.
- Use OPTGROUP to organize long lists of menu options into smaller groups. [p. 49] .
- Use tables for tabular data [p. 22] and describe the table with CAPTION.
- Group table rows and columns [p. 22] with THEAD, TBODY, TFOOT, and COLGROUP.
- Nest lists [p. 19] with UL, OL, and DL.
- Use section headers [p. 17] (H1 - H6) to create structured documents and break up long stretches of text.
- Break up lines of text into paragraphs (with the P element).

All of these grouping mechanisms should be used when appropriate and natural, i.e., when the information lends itself to logical groups. Content developers should not create groups randomly, as this will confuse all users.

1.7 Content negotiation

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

1. Instead of including links such as "Here is the French version of this document", use content negotiation so that the French version is served to clients preferring
2. If not possible to use content negotiation, in HTML use "type" and "hreflang".

1.8 Automatic page refresh

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. There are two types of refresh mechanisms commonly used.

[Checkpoint 7.4] Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2] This type of refresh changes the user's page at regular intervals. This might be accomplished by the following HTML markup:

Deprecated example.

```
<META http-equiv="refresh" content="60">
<BODY>
<P>...Information...
</BODY>
```

Content developers should **not** use this technique to simulate "push" technology. Developers cannot predict how much time a user will require to read a page; premature refresh can disorient users. Content developers should avoid periodic refresh and allow users to choose when they want the latest information.

[Checkpoint 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] This type of forward redirects the user from one page to another generally after a timeout. The following HTML markup is frequently used to achieve this effect:

Deprecated example.

```
<HEAD>
<TITLE>Don't use this!</TITLE>
<META http-equiv="refresh" content="5;
         http://www.acme.com/newpage">
</HEAD>
<BODY>
<P>If your browser supports Refresh,
you'll be transported to our
<A href="http://www.acme.com/newpage">new site</A>
in 5 seconds, otherwise, select the link manually.
</BODY>
```

However, users should **not** redirect users with this markup since is non-standard, it disorients users, and it can disrupt a browser's history of visited pages. 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.

1.9 Other topics

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

For example, Indicate which is the first page of the document and which page follows the current one. (e.g., by using the LINK element).

1.10 Validation

1. Use an automated accessibility validation tool such as Bobby (refer to [BOBBY] [p. 53]).
2. Use an HTML validation service such as the W3C HTML Validation Service (refer to [HTMLVAL] [p. 53]).
3. Use a style sheets validation service such as the W3C CSS Validation Service (refer to [CSSVAL] [p. 53]).
4. Test your pages with a text-only browser such as Lynx ([LYNX] [p. 54]) or a Lynx emulator such as Lynx Viewer ([LYNXVIEW] [p. 54]) or Lynx-me ([LYNXME] [p. 54]).
5. Use multiple graphic browsers, with:
   - sounds and graphics loaded,
   - graphics not loaded,
   - sounds not loaded,
   - no mouse,
   - frames, scripts, style sheets, and applets not loaded
6. Use several browsers, old and new.
7. Use a self-voicing browser, a screen reader, magnification software, a small display, etc. Self-voicing browsers include "pwwebspeak".
8. Use spell and grammar checkers. 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. Eliminating grammar problems increases comprehension.

If, after completing these tests and adjusting your design accordingly, you find that your page is still not accessible, you must create an alternative page [p. 9] that is accessible.

1.11 Browser Support

1. [Checkpoint 11.1] Use W3C technologies and use the latest versions when they are supported. [Priority 2] .
2. In general, user agents ignore HTML attributes they don't support and they render the content of unsupported elements.

---

2 HTML topics

The following sections list some techniques for using HTML and CSS to design accessible documents and some techniques for avoiding HTML accessibility traps. The sections are organized by topic (and mirror the organization of the HTML 4.0 specification, [HTML40] [p. 52]).

2.1 Document structure and metadata

[Checkpoint 3.5] Create documents that validate to published formal grammars. [Priority 2]

As discussed above, content developers should use structural markup wherever possible (and use it as intended by the authors of W3C specifications). Structural elements promote consistency in documents and supply information to other tools (e.g., indexing tools, search engines, programs that extract tables to databases, navigation tools that use header elements, and automatic translation software that translates text from one language into another.

2.1.1 Metadata

Some structural elements provide information about the document itself. This is called "metadata" about the document (Metadata is information about data). Well-crafted metadata can provide important orientation information to users. HTML elements that provide useful information about a document include:

- TITLE: The document title. Note that the TITLE element (one time only in a document) is different from the "title" attribute, which applies to almost every HTML 4.0 element. This document makes use of the "title" attribute for many occasions that require advisory titles. Content developers should use the "title" attribute in accordance with the HTML 4.0 specification. For example, "title" should be used with links to provide information about the target of the link.
- ADDRESS: Can be used to provide information about the creator of the page.

- LINK: Can be used to indicate alternative documents (different structure, different language, different target device, etc.).
- The META element can be used to describe metadata about a document. Please refer to the section on automatic page refresh [p. 14] for information on why META should not be used to redirect pages.

2.1.2 Section headers

Sections should be introduced with the HTML header elements (H1-H6). Other markup may complement these elements to improve presentation (e.g., the HR element to create a horizontal dividing line), but visual presentation is not sufficient to identify document sections.

[Checkpoint 3.2] Use header elements to convey logical structure and use them according to specification. [Priority 2] Since some users skim through a document by navigating its headings, it is important to use them appropriately to convey document structure. Users should order heading elements properly. For example, in HTML, H2 elements should follow H1 elements, H3 elements should follow H2 elements, etc. Content developers should not "skip" levels (e.g., H1 directly to H3). Do not use headings to create font effects; use style sheets [p. 38] .

Note that in HTML, heading elements (H1 - H6) do not contain entire logical sections in their content. They should be used to begin a section of a document. The following HTML markup shows how to create a true document "section" and control its appearance with style sheets:

Example.

```
<HEAD>
<TITLE>Cooking techniques</TITLE>
<STYLE type="text/css">
   /* Indent the entire section */
   DIV.section2 { margin-left: 5% }
</STYLE>
</HEAD>
<BODY>
<H1>Cooking techniques</H1>
... some text here ...
<DIV class="section2">
<H2>Cooking with oil</H2>
... text of the section ...
</DIV>

<DIV class="section2">
<H2>Cooking with butter</H2>
... text of the next section ...
</DIV>
```

End example.

2.2 Language information

[Checkpoint 4.1] Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions) of non-text content. [Priority 1] If you use a number of different languages on a page, make sure that any changes in language are clearly identified by using the "lang" attribute:

Example.

```
<P>And with a certain <SPAN lang="fr">je ne sais quoi</SPAN>,
she entered both the room, and his life, forever. <Q>My name
is Natasha,</Q> she said. <Q lang="it">Piacere,</Q>
he replied in impeccable Italian, locking the door.
```

End example.

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

2.3 Text markup

As mentioned above, structural elements add information to a page that may be used by browsers, search engines, and other software. Content developers are encouraged to use structural elements and attributes [p. 5] whenever possible. Below we discuss how to further improve accessibility by careful use of attributes with these elements.

2.3.1 Emphasis

The proper HTML elements should be used to mark up emphasis: EM and STRONG. The B and I elements should not be used; they are used to create a visual presentation effect. The EM and STRONG elements were designed to indicate structural emphasis that may be rendered in a variety of ways (font style changes, speech inflection changes, etc.)

2.3.2 Acronyms and abbreviations

[Checkpoint 4.2] Specify the expansion of abbreviations and acronyms. [Priority 2 for the first occurrence of the acronym or abbreviation in a given document, Priority 3 thereafter.] Mark up abbreviations and acronyms with ABBR and ACRONYM and use "title" to indicate the expansion:

Example.

```
<P>Welcome to the <ACRONYM title="World Wide Web">WWW</ACRONYM>!
```

End example.

2.3.3 Quotations

[Checkpoint 3.4] Mark up quotations. Do not use quotation markup for formatting effects such as indentation. [Priority 2] See the language example [p. 18] above for an illustration of the Q element.

Example.

```
<BLOCKQUOTE cite="http://www.shakespeare.com/loveslabourlost">
  <P>Remuneration! O! that’s the Latin word for three farthings.
     --- William Shakespeare (Love’s Labor Lost).
  </P>
</BLOCKQUOTE>
```

End example.

2.3.4 Text markup rather than images

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

2.4 Lists

[Checkpoint 3.3] Mark up lists and list items properly. [Priority 2] The HTML list elements DL, UL, and OL (available in HTML 3.2 and HTML 4.0) should only be used to create lists, not for formatting effects such as indentation.

Ordered lists help non-visual users navigate. Non-visual users often "get lost" in lists, especially those with several layers of embedding and those that do not indicate the specific level of indentation for each item. Content developers are encouraged to use UL for unordered lists and OL for ordered lists (i.e., use markup appropriately). However, until user agents provide a means to identify list context clearly (e.g., by supporting the ’:before’ pseudo-element in CSS2), content developers should consider including contextual clues in their lists.

Please note that even for numbered lists, compound numbers are more informative than simple numbers. Thus, a list numbered like this:

```
1.
  1.1
  2.1
  3.1
2.
  2.1
```

is more informative than a list numbered like this:

```
1.
  1.
  2.
  3.
2.
  2.
```

[CSS1] [p. 52] and [CSS2] [p. 52] even more so provide ways to control list numbering styles. Users may apply list numbering styles even to unordered lists through user style sheets.

Non-visual users may have difficulties knowing where a list itself begins and ends and where each list item starts. Furthermore, if a list entry wraps to the next line on the screen, it may appear to be two separate items in the list. This may pose a problem for legacy screen readers.

Example.

The following CSS2 style sheet shows how to provide compound numbers for nested lists created with either UL or OL elements. Items are numbered as "1", "1.1", "1.1.1", etc.

```
<STYLE type="text/css">
   UL, OL { counter-reset: item }
   LI { display: block }
   LI:before { content: counters(item, "."); counter-increment: item }
</STYLE>
```

Until either CSS2 is widely supported by users agents or user agents allow users to control rendering of lists through other means, authors should consider providing contextual clues in nested lists. The following CSS1 mechanism shows how to hide the end of a list when style sheets are turned on and to reveal it when style sheets are turned off, when user style sheets override the hiding mechanism, or when style sheets are not supported:

```
<HEAD>
   <TITLE>Contextual clues in nested lists</TITLE>
   <STYLE type="text/css">
      .endoflist { display: none }
   </STYLE>
<BODY>
   <UL>
     <LI>Paper:
       <UL>
       <LI>Envelopes
       <LI>Notepaper
       <LI>Letterhead
       <LI>Poster paper
          <SPAN class="endoflist">(End of Paper)</SPAN>
       </UL>
     <LI>Pens:
       <UL>
       <LI>Blue writing pens
       <LI>whiteboard pens
          <SPAN class="endoflist">(End of Pens)</SPAN>
       </UL>
    <LI>Fasteners:
       <UL>
       <LI>paper clips
       <LI>staples
```

```
      <LI>Big lengths of rope.
         <SPAN class="endoflist">(End of Fasteners)</SPAN>
      </UL>
   </UL>
```

End example.

2.4.1 Use style sheets to change list bullets

To change the "bullet" style of unordered list items, use style sheets. This way, if images are not loaded, the browser will draw a default bullet.

Example.

```
<HEAD>
<TITLE>Using style sheets to change bullets</TITLE>
<STYLE type="text/css">
   UL { list-style: url(star.gif) }
</STYLE>
</HEAD>
<BODY>
<UL>
   <LI>Audrey
   <LI>Laurie
   <LI>Alice
</UL>
```

End example.

Avoid using images as bullets in definition lists. However, if this method is used, be sure to provide alt-text [p. 7] for the images.

Deprecated example.

```
<DL>
   <DD><IMG src="star.gif" alt="Item">Audrey
   <DD><IMG src="star.gif" alt="Item">Laurie
   <DD><IMG src="star.gif" alt="Item">Alice
</DL>
```

Content developers should avoid list styles where bullets provide additional (visual) information. However, if this is done, be sure to provide alt-text describing meaning of the bullet:

Deprecated example.

```
<DL>
<DD><IMG src="red.gif" alt="New:">Roth IRA</DD>
<DD><IMG src="yellow.gif" alt="Old:">401(k)</DD>
</DL>
```

Here is a better way to change list bullet styles (using style sheets). To further ensure that users understand differences between list items indicated visually, content developers should provide a label before or after the list item phrase:

Example.

```
<HEAD>
<TITLE>Bullet styles example</TITLE>
<STYLE type="text/css">
   .newtxt { font-weight: bold;
             color: red;
             background-color: yellow }
   .newbullet { list-style : url(yellow.gif) }
</STYLE>
</HEAD>
<BODY>
<UL>
   <LI class="newbullet">Roth IRA <SPAN class="newtext">New</SPAN></LI>
   <LI> 401(k)</LI>
</UL>
</BODY>
```

End example.

2.5 Tables

Content developers may make tables more accessible in a number of ways:

- Provide a caption via the CAPTION element.
- [Checkpoint 5.5] Provide summaries for tables. [Priority 3] Summaries are especially useful for non-visual readers.
- [Checkpoint 5.1] For data tables, identify row and column headers. [Priority 1] Future browsers and assistive technologies will be able to automatically translate tables into linear sequences if data is labeled appropriately.
- [Checkpoint 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] Identify structural groups of rows (THEAD for repeated table headers, TFOOT for repeated table footers, and TBODY for other groups of rows) and groups of columns (COLGROUP and COL). Label table elements with the "scope", "headers", and "axis" attributes so that future browsers and assistive technologies will be able to select data from a table by filtering on categories. This markup will also help browsers translate tables into linear sequences automatically (also called table "serialization"). A linear sequence is usually generated by reading a row left to right and proceeding each cell with the label of its column.
- [Checkpoint 5.6] Provide abbreviations for header labels. [Priority 3] Provide terse substitutes for header labels with the "abbr" attribute on TH. These will be particularly useful for future speaking technologies that can read row and column labels for each cell. Abbreviations cut down on repetition and reading time.
- [Checkpoint 5.4] If a table is used for layout, do not use any structural markup for the purpose of visual formatting. [Priority 2]

Most of the above elements and attributes are only available in HTML 4.0.

This markup will allow accessible browsers and other user agents to restructure tables for non-visual media.

For information about table headers, see the table header algorithm and discussion in the HTML 4.0 Recommendation ([HTML40] [p. 52] , section 11.4.3).

The following example shows how to associate data cells with their corresponding headers by means of the "headers" attribute. The "headers" attribute specifies a list of header cells (row and column labels) associated with the current data cell. This requires each header cell to have an "id" attribute.

Example.

```
<TABLE border="1"
       summary="This table charts the number of
                cups of coffee consumed by each senator,
                the type of coffee (decaf or regular),
                and whether taken with sugar.">
  <CAPTION>Cups of coffee consumed by each senator</CAPTION>
  <TR>
      <TH id="header1">Name</TH>
      <TH id="header2">Cups</TH>
      <TH id="header3" abbr="Type">Type of  Coffee</TH>
      <TH id="header4">Sugar?</TH>
  <TR>
      <TD headers="header1">T. Sexton</TD>
      <TD headers="header2">10</TD>
      <TD headers="header3">Espresso</TD>
      <TD headers="header4">No</TD>
  <TR>
      <TD headers="header1">J. Dinnen</TD>
      <TD headers="header2">5</TD>
      <TD headers="header3">Decaf</TD>
     <TD headers="header4">Yes</TD>
</TABLE>
```

End example.

A speech synthesizer might render this tables as follows:

```
Caption: Cups of coffee consumed by each senator
Summary: This table charts the number of cups of coffee
         consumed by each senator, the type of coffee
        (decaf or regular), and whether taken with sugar.
Name: T. Sexton, Cups: 10, Type: Espresso, Sugar: No
Name: J. Dinnen, Cups: 5, Type: Decaf, Sugar: Yes
```

A visual user agent might render this table as follows:

Cups of coffee consumed by each senator

Name	Cups	Type of Coffee	Sugar?
T. Sexton	10	Espresso	No
J. Dinnen	5	Decaf	Yes

[D]

The next example associates the same header and data cells as before, but this time uses the "scope" attribute rather than"headers." "Scope" must have one of the following values: row, col, rowgroup or colgroup. Scope specifies the set of data cells to be associated with the current header cell. This method is particularly useful for simple tables. It should be noted that the spoken rendering of this table would be identical to that of the previous example. A choice between the "headers" and "scope" attributes is dependent on the complexity of the table. It does not affect the output so long as the relationships between header and data cells are made clear in the markup.

Example.

```
<TABLE border="1"
       summary="This table charts ...">
   <CAPTION>Cups of coffee consumed by each senator</CAPTION>
   <TR>
        <TH scope="col">Name</TH>
        <TH scope="col">Cups</TH>
        <TH scope="col" abbr="Type">Type of Coffee</TH>
        <TH scope="col">Sugar?</TH>
  <TR>
        <TD>T. Sexton</TD>  <TD>10</TD>
        <TD>Espresso</TD>   <TD>No</TD>
  <TR>
        <TD>J. Dinnen</TD>  <TD>5</TD>
        <TD>Decaf</TD>       <TD>Yes</TD>
</TABLE>
```

End example.

The following example shows how to create categories within a table using the "axis" attribute.

Example.

```
<TABLE border="1">
  <CAPTION>Travel Expense Report</CAPTION>
  <TR>
       <TH></TH>
       <TH id="header2" axis="expenses">Meals
       <TH id="header3" axis="expenses">Hotels
       <TH id="header4" axis="expenses">Transport
       <TD>subtotals</TD>
  <TR>
       <TH id="header6" axis="location">San Jose
       <TH> <TH> <TH> <TD>
  <TR>
      <TD id="header7" axis="date">25-Aug-97
      <TD headers="header6 header7 header2">37.74
      <TD headers="header6 header7 header3">112.00
      <TD headers="header6 header7 header4">45.00
      <TD>
  <TR>
      <TD id="header8" axis="date">26-Aug-97
      <TD headers="header6 header8 header2">27.28
      <TD headers="header6 header8 header3">112.00
```

```
        <TD headers="header6 header8 header4">45.00
        <TD>
    <TR>
        <TD>subtotals
        <TD>65.02
        <TD>224.00
        <TD>90.00
        <TD>379.02
    <TR>
        <TH id="header10" axis="location">Seattle
        <TH> <TH> <TH> <TD>
    <TR>
        <TD id="header11" axis="date">27-Aug-97
        <TD headers="header10 header11 header2">96.25
        <TD headers="header10 header11 header3">109.00
        <TD headers="header10 header11 header4">36.00
        <TD>
    <TR>
        <TD id="header12" axis="date">28-Aug-97
        <TD headers="header10 header12 header2">35.00
        <TD headers="header10 header12 header3">109.00
        <TD headers="header10 header12 header4">36.00
        <TD>
    <TR>
        <TD>subtotals
        <TD>131.25
        <TD>218.00
        <TD>72.00
        <TD>421.25
    <TR>
        <TH>Totals
        <TD>196.27
        <TD>442.00
        <TD>162.00
        <TD>800.27
</TABLE>
```

End example.

This table lists travel expenses at two locations: San Jose and Seattle, by date, and category (meals, hotels, and transport). The following image shows how a visual user agent might render it.

Travel Expense Report

	Meals	Hotels	Transport	subtotals
San Jose				
25-Aug-97	37.74	112.00	45.00	
26-Aug-97	27.28	112.00	45.00	
subtotals	65.02	224.00	90.00	379.02
Seattle				
27-Aug-97	96.25	109.00	36.00	
28-Aug-97	35.00	109.00	36.00	
subtotals	131.25	218.00	72.00	421.25
Totals	196.27	442.00	162.00	**800.27**

[D]

2.5.1 Wrapped text in tables

One source of problems for screen readers that do not interpret the source HTML is wrapped text in table cells. They read across the page, reading sentences on the same row from different columns as one sentence.

For example, if a table is rendered like this on the screen:

```
There is a 30% chance of              Classes at the University of Wisconsin
rain showers this morning, but they   will resume on September 3rd.
should stop before the weekend.
```

This might be read by a screen reader as:

```
There is a 30% chance of Classes at the University of Wisconsin
rain showers this morning, but they will resume on September 3rd.
should stop before the weekend.
```

Screen readers that read the source HTML will recognize the structure of each cell, but for older screen readers, content developers should minimize the risk of word wrapping by limiting the amount of text in each cell. Also, the longest chunks of text should all be in the last column (rightmost for left-to-right tables). This way, if they wrap, they will still be read coherently.

Content developers should test tables for wrapping with a browser window dimension of "640x480".

Quicktest! To get a better understanding of how a screen reader would read a table, run a piece of paper down the page and read your table line by line.

2.5.2 Backwards compatibility issues for tables

Rows of a TFOOT element will appear before the BODY of the document in an HTML3.2 browser.

2.5.3 Avoid tables for layout

[Checkpoint 5.3] Avoid using tables for layout. [Priority 2]

2.6 Links

Users who are blind often jump from link to link when skimming a page or looking for information. When they do this, only the text of the link (the "link text") is read.

"Auditory users," people who are **blind**, **have difficulty seeing**, or who are **using devices with small or no displays** are unable to scan the page quickly with their eyes and often use a list of links to get an overview of a page or to quickly find a link. When links are not descriptive enough, do not make sense when read out of context, or are not unique, the auditory user must stop to read the text surrounding each link to identify it.

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

Avoid, for example, general link text, such as "click here" (which is device-dependent in addition to saying nothing about what is to be found at the end of the link). Instead of "click here", link text should indicate the nature of the link target, as in "more information about sea lions" or "text-only version of this page". Note that for the latter case (and other format- or language-specific documents), content developers are encouraged to use content negotiation [p. 14] instead, so that users who prefer text versions will have them served automatically.

Non-visual users often browse by reading only the links of a document, tabbing from one to the next as the links appear in the document source (or in an author-specified tabbing order). Not every link must be understandable entirely out of context as long as the targets are clear when the text of a series of links read in succession is sufficiently clear.

In addition to clear link text, content developers may specify a value of the "title" attribute that clearly and accurately describes the target of the link.

When an image is used as the content of a link, specify alt-text [p. 7] for the image that makes sense in context.

Quicktest! To choose alt-text in this case, think of what you would say in words rather than an image in this context.

Example.

```
<A href="routes.html">
   <IMG src="topo.html"
        alt="Current routes at Boulders Climbing Gym">
</A>
```

End example.

If more than one link on a page shares the same link text, all those links should point to the same resource. Such consistency will help page design as well as accessibility. If two or more links refer to different targets but share the same link text, HTML authors should distinguish the links by specifying a different value for the

"title" attribute of each link.

"Navigation bars" (sets of links that appear on every page in a site) are usually the first thing someone encounters on a page. For speech users, this means taking the time to read through x number of links on every page before reaching the unique content of a page. Therefore, grouping links will allow a user with a user agent that can navigate by elements, to jump over the group. This is similar to how people with vision skip reading the links when they see the same set on each page. Since this type of mechanism is not available today, providing a link that skips over the links, or using a tabindex at a link just before the content begins are strategies that work today.

Example.

```
<HEAD>
<TITLE>How to use our site</TITLE>
</HEAD>
<BODY>
  <P class="nav">
    <A href="home.html">[Home]</A>
    <A href="search.html">[Search]</A>
    <A href="new.html">[New and highlighted]</A>
    <A href="sitemap.html">[Site map]</A>
  </P>
  <H1><A name="how" tabindex=1>How to use our site</A></H1>
<!-- content of page -->
</BODY>
```

End example.

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

For example, in HTML, creating a navigation bar composed of links with DIV, SPAN, FRAME, etc. Identify the group with the "id" attribute. Use "tabindex=1" on an anchor after the group so users may quickly skip the navigation bar.

2.7 Images and image maps

Images may be inserted by two elements in HTML:

IMG
: Available in HTML 4.0, 3.2, 2.0

OBJECT
: Available in HTML 4.0.

Images include those that carry out simple animations (e.g., a "gif" image).

2.7.1 Equivalent text for images

[Checkpoint 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, image map regions, short animations (e.g., animated GIFs), applets, ascii art, frames, scripts, inserted list bullets, sounds (played with or without user interaction), stand-alone audio files, synthesized speech, audio tracks of video, and video.
[Priority 1] When using IMG, specify alt-text with the "alt" attribute.

Example.

```
<IMG src="magnifyingglass.gif" alt="Search">
```

End example.

When using OBJECT, specify alt-text [p. 7]

in the body of the OBJECT element:

Example.

```
<OBJECT data="magnifyingglass.gif" type="image/gif">
   Search
</OBJECT>
```

End example.

Additionally, use the "title" attribute:

Example.

```
<OBJECT classid="Duke.class" title="Hello!"
        width="50" height="50">
     Duke waves hello!
</OBJECT>
```

End example.

2.7.2 Long descriptions for images

When using IMG, specify a long description [p. 8] of the image with the "longdesc" attribute:

Example.

```
<IMG src="97sales.gif" alt="Sales for 1997"
     title="Sales pie chart"
     longdesc="sales97.html">
```

In sales97.html:

A chart showing how sales in 1997 progressed. The chart
is a bar-chart showing percentage increases in sales
by month. Sales in January were up 10% from December 1996,
sales in February dropped 3%, ..

End example.

For browsers that don't support "longdesc", provide a description link as well next to the graphic:

Example.

```
<IMG src="97sales.gif" alt="Sales for 1997" longdesc="sales.html">
<A href="sales.html" title="Description of 1997 sales figures">[D]</A>
```

End example.

When using OBJECT, provide a long description in the body of the element:

Example.

```
<OBJECT data="97sales.gif" type="image/gif">
       Sales in 1997 were down subsequent to our
       anticipated purchase ...
</OBJECT>
```

End example.

Or, provide a link to a long description within the body of the element:

Example.

```
<OBJECT data="97sales.gif" type="image/gif">
       Chart of our Sales in 1997.
       A <A href="desc.html">textual description</A> is available.
</OBJECT>
```

End example.

2.7.3 Ascii art

[Checkpoint 1.5] Replace ASCII art with an image or explain it. [Priority 1 or Priority 2 depending on the importance of the information.]

Avoid ascii art (character illustrations) and use real images instead since it is easier to supply alt-text [p. 7] and long descriptions [p. 8] for images. The priority of this checkpoint depends on the importance of the information (e.g., an important chart).

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

However, if ascii art must be used provide a link to jump over the ASCII art, as follows.

Example.

```
<P>
<a href="#post-art">skip over ASCII art</a>
<!-- ASCII art goes here -->
<a name="post-art">caption for ASCII art</a>
```

End example.

ASCII art may also be marked up as follows:

Example.

```
<P>
<OBJECT data="cow.txt" type="text/x-ascii-art" title="drawing of a cow">
include cow ascii art from guidelines
</OBJECT>
```

End example.

Another option is to use a SPAN or an ABBR element with "title".

Example.

```
<P><SPAN class="smile" title="smiley in ascii art">:-)</SPAN>
```

End example.

If the description of (important) ASCII art is long, provide a description [p. 8] in addition to alt-text.

Another way to replace ascii art is to use human language substitutes. For example, <wink> might substitute for the emoticon <SPAN title="wink smiley">;-)</SPAN>, the word"therefore" could replace arrows consisting of dashes and greater than signs (e.g., -->), and the word "great" for the uncommon abbreviation "gr8".

2.7.4 Mathematical equations in images

2.7.5 Image maps

An image map is an image that has "active regions". When the user selects one of the regions, some action takes place -- a link may be followed, information sent to a server, etc. To make an image map accessible, content developers must ensure that each action associated with a visual region may be activated without a pointing device.

Image maps are created with the MAP element (available in HTML 4.0 and 3.2). HTML allows two types of image maps: client-side (the user’s browser processes a URI) and server-side (the server processes click coordinates). For all image maps, content developers must supply alt-text [p. 7] (described below), and long descriptions [p. 8] where needed (see the section on long descriptions for images [p. 29]).

Content developers should either avoid server-side image maps (because they require a specific input device - a mouse) or provide the same functionality or information in an alternative accessible format. One way to achieve this is to provide a textual link for each active region so that each link is navigable with the keyboard [p. 10] . If you must use a server-side image map, please consult the section on server-side image maps [p. 33]

2.7.6 Client-side image maps

The active regions of a client-side image map are defined within the MAP element and may be created with two elements:

AREA
: Available in HTML 4.0, 3.2

A
: Available in HTML 4.0.

[Checkpoint 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]

Provide text equivalents [p. 7] for image maps since they convey visual information.

If AREA is used, use the"alt" attribute:

Example.

```
<IMG src="welcome.gif" alt="Image map of areas in the library"
     usemap="#map1">
<MAP name="map1">
  <AREA shape="rect" coords="0,0,30,30"
        href="reference.html" alt="Reference">
  <AREA shape="rect" coords="34,34,100,100"
        href="media.html" alt="Audio visual lab">
</MAP>
```

End example.

The same idea, but use OBJECT instead of IMG to insert the image to provide more information about the image:

Example.

```
<OBJECT data="welcome.gif" type="image/gif" usemap="#map1">
    There are several areas in the library including
    the <A href="reference.html">Reference</A> section and the
    <A href="media.html">Audio Visual Lab</A>.
</OBJECT>
<MAP name="map1">
  <AREA shape="rect" coords="0,0,30,30"
        href="reference.html" alt="Reference">
  <AREA shape="rect" coords="34,34,100,100"
        href="media.html" alt="Audio visual lab">
</MAP>
```

End example.

[Checkpoint 1.2] Provide redundant text links for each active region of an image map. [Priority 1 - if server-side image maps are used, Priority 2 - if client-side image maps are used. Redundant text links for client-side image maps are only required until user agents render text equivalents for the map links.] In addition to providing

alt-text, provide redundant textual links. If the A element is used instead of AREA, the content developer may describe the active regions and provide redundant links at the same time:

Example.

```
<OBJECT data="navbar1.gif" type="image/gif" usemap="#map1">
<MAP name="map1">
  <P>Navigate the site.
  <A href="guide.html" shape="rect"
     coords="0,0,118,28">[Access Guide]</A>
  <A href="shortcut.html" shape="rect"
     coords="118,0,184,28">[Go]</A>
  <A href="search.html" shape="circle"
     coords="184.200,60">[Search]</A>
  <A href="top10.html" shape="poly"
     coords="276,0,373,28,50,50">[Top Ten]</A>
</MAP>
</OBJECT>
```

End example.

Note that in the previous example the MAP element is the content of the OBJECT element so that the alternative links will only be displayed if the image map (navbar1.gif) is not.

Note also that links have been separated by brackets ([]). This is to prevent screen readers from reading several adjacent links as a single link.

[Checkpoint 10.5] Until user agents or assistive technologies render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links. [Priority 3] Content developers should make sure they include printable characters (such as brackets or a vertical bar (|)) surrounded by spaces between adjacent links.

2.7.7 Server-side image maps

When a server-side image map must be used, content developers should provide an alternative list of image map choices. There are three techniques:

- If an alternative list of links follows the image map, content developers should indicate the existence and location of the alternative list. If IMG is used to insert the image, provide this information in the"alt" attribute. If OBJECT is used, provide it in the "title" attribute.

  Example.

```
<IMG src="welcome.gif" alt="Links to this image map follow immediately"
     usemap="#map1" title="Welcome insignia" >
<MAP name="map1">
  <AREA shape="rect" coords="0,0,30,30"
        href="reference.html" alt="Reference">
  <AREA shape="rect" coords="34,34,100,100"
        href="media.html" alt="Audio visual lab">
</MAP>

<P><A href="reference.html">[Reference]</A>
   <A href="media.html">[Audio Visual Lab]</A>
```

End example.

- A more straightforward solution, although newer and less backwards compatible, is to include the alternative links within the body of an OBJECT element (see the previous example illustrating links in the OBJECT element [p. 33]).
- One final possibility is to create an alternative page [p. 9] that is accessible.

[Checkpoint 3.8] Provide individual button controls in a form rather than simulating a set of buttons with an image map. [Priority 2]

2.8 Applets and other objects

Applets may be inserted by two elements in HTML:

APPLET
: Available in HTML 4.0 (deprecated), 3.2.

OBJECT
: Available in HTML 4.0.

Other objects, such as those requiring a plug-in, should also use the OBJECT element. However, for backwards compatibility with Netscape browsers, use the proprietary EMBED element within the OBJECT element as follows:

Example.

```
<OBJECT classid="clsid:A12BCD3F-GH4I-56JK-xyz"
codebase="http://site.com/content.cab" width=100 height=80>
<PARAM name="Movie" value="moviename.swf">
    <EMBED src="moviename.swf" width=100 height=80
    pluginspage="http://www.macromedia.com/shockwave/download/">
    </EMBED>

    <NOEMBED>
        <IMG alt="Still from Movie"
                src="moviename.gif" width=100 height=80>
    </NOEMBED>

</OBJECT>
```

End example.

For more information refer to [MACROMEDIA] [p. 54] .

2.8.1 Audio and Video produced by dynamic objects

There are several checkpoints that content developers should use to ensure that audio or visual information presented by a dynamic object is accessible:

1. Provide a long description if necessary. See the section on long descriptions for images [p. 29] .
2. Refer also to checkpoint 8.1.
3. [Checkpoint 1.4] For any time-based presentation (e.g., a movie, animation, or multimedia presentation), synchronize equivalent alternatives (e.g., captions or video descriptions) with the presentation. [Priority 1]

2.8.2 Equivalent text and descriptions for applets and programmatic objects

If OBJECT is used, provide alt-text [p. 7] as the content of the element:

Example.

```
<OBJECT classid="java:Press.class" width="500" height="500"
        title="Java applet: how temperature affects pressure">
      As temperature increases, the molecules in the balloon...
</OBJECT>
```

End example.

A more complex example takes advantage of the fact the OBJECT elements may be embedded to provide for alternative representations of information:

Example.

```
<OBJECT title="How temperature affects pressure"
        classid="java:Press.class" width="500" height="500">
   <OBJECT data="Pressure.mpeg" type="video/mpeg">
      <OBJECT data="Pressure.gif" type="image/gif">
         As temperature increases, the molecules in the balloon...
      </OBJECT>
   </OBJECT>
</OBJECT>
```

End example.

If APPLET is used, provide alt-text [p. 7] with the "alt" attribute and content in the APPLET element. This enables them to transform gracefully for those user agents that only support one of the two mechanisms ("alt" or content).

Deprecated example.

```
<APPLET code="Press.class" width="500" height="500"
        alt="Java applet: how temperature affects pressure">
      As temperature increases, the molecules in the balloon...
</APPLET>
```

2.8.3 Directly accessible applets

[Checkpoint 8.1] Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important and not presented elsewhere, otherwise Priority 2.] If an applet requires user interaction (e.g., the ability to manipulate a physics experiment) that cannot be duplicated in an alternative format, make the applet directly accessible.

For more information about accessible applets, please refer to [JAVAACCESS] [p. 54] and [IBMJAVA] [p. 53] .

2.9 Audio and video

Audio and video should be accompanied by *text transcripts*, textual descriptions or equivalents of auditory or visual 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. Full audio transcripts include spoken dialogue as well as any other significant sounds including on-screen and off-screen sounds, music, laughter, applause, etc. The following two examples show captions, a text transcript, and auditory descriptions.

Example.

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

[phone rings]

[ring]

[ring]

Hello?"

End example.

Example.

Here’s an example of a transcript of a clip from "The Lion King" (available at [DVS] [p. 53]).

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.

2.9.1 Audio information

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.

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. 53]) 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. 54] .

2.9.2 Visual information and motion

Video descriptions are used primarily by people who are blind to follow the action and other non-auditory information in video material. The description provides 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.

[Checkpoint 1.3] For each movie, provide an auditory description of the video track and synchronize it with the audio track. [Priority 1] For movies, provide auditory descriptions that are synchronized with the original audio. See the section on audio information [p. 37] for more information about multimedia formats.

Text transcripts, in conjunction with the full audio transcript described above, allow access by people with both visual and hearing disabilities. This also provides everyone with the ability to index and search for information contained in audio/visual materials.

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

However, if necessary to include an applets that involves motion or updates, content developers should provide a mechanism for freezing this motion (for an example, refer to [TRACE] [p. 54]). Content developers should use animated gifs to create motion that may be suspended by the browser for people that have trouble with it.

When necessary, a long description be provided for visual information (e.g., animations) to enable understanding of the page. For example for an ad that displays text like a marquee, the text should be provided in the text equivalent, unless there is a lot of text. For a looping image of cloud cover over the United States, if the image is in the context of a weather status report, where the information is presented in text, a less verbose description of the image is necessary. However, if the image appears on in a pedagogical setting, elaborating on cloud formations in relation to land mass, ought to be described.

See also the section on text style [p. 39] for controlling blinking.

2.10 Style and style sheets

[Checkpoint 3.6] Use style sheets to control layout and presentation. [Priority 2] CSS1 ([[CSS1] [p. 52]) and CSS2 ([[CSS2] [p. 52]) allow content developers to duplicate almost every HTML 4.0 presentation feature and offer more power with less cost. However, until most users have browsers that support style sheets, not every presentation idiom may be expressed satisfactorily with style sheets. In the following sections, we show how style sheets may be used to create accessible pages. We also provide examples of how to use HTML 4.0 features (e.g., tables, bitmap text) more accessibly when they must be used.

See also the section on text markup [p. 19] .

2.10.1 General style sheet techniques

Make sure to validate [p. 15] that your pages still work when style sheets are turned off!

- [Checkpoint 6.1] Organize content logically using appropriate structural markup so the organization remains clear even when associated style sheets are turned off or are not supported. [Priority 1]
- [Checkpoint 3.7] Use relative rather than absolute units in markup language attribute values and style sheet property values. [Priority 2]

2.10.2 Text formatting

Content developers should use style sheets for text formatting rather than converting text to images. For example, stylized text on a colored background can be created with style sheets instead of as an image. This provides flexibility for people to view the text in a form that is most readable to them including magnified, in a particular

color combination such as white on black, or in a particular font.

However, if you must use a bitmap to create a text effect (special font, transformation, shadows, etc.) it must be accessible.

When bitmapped text is used in a way that makes a page inaccessible, content developers must supply alternative pages [p. 9] .

To make a bitmap representing text accessible, it must have alt-text [p. 7] that is the same text represented by the image.

Example.

In this example, the inserted image shows the large red characters "Example", reflected by the alt-text.

```
<P>This is an
   <IMG src="BigRedExample.gif" alt="Example"> of what we mean.
</P>
```

End example.

This is true of Drop Caps (large first letter of a paragraph) as well. However, we recommend using style sheets to create the effect, as the following example illustrates.

Example.

```
<HEAD>
<TITLE>Drop caps</TITLE>
<STYLE type="text/css">
      .dropcap { font-size : 120%; font-family : Helvetica }
</STYLE>
</HEAD>
<BODY>
<P><SPAN class="dropcap">O</SPAN>nce upon a time...
</BODY>
```

Note. As of the writing of this document, the CSS pseudo-element ':first-letter', which allows content developers to refer to the first letter of a chunk of text, is not widely supported.

2.10.3 Text style

Content developers should use style sheets instead of deprecated presentation elements and attributes [p. 6] that control visual presentation.

- [Checkpoint 7.1] Until user agents allow users to control it, avoid causing the screen to flicker. [Priority 1] A flickering or flashing screen may cause seizures in users with photosensitive epilepsy. 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).
- [Checkpoint 7.2] Until user agents allow users to control it, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on

and off). [Priority 2] If blinking content (e.g., a headline that appears and disappears at regular intervals) is used, provide a mechanism for stopping the blinking. In CSS, 'text-decoration: blink' will cause content to blink and will allow users to stop the effect by turning off style sheets or overriding the rule in a user style sheet. **Note.** Do not use the BLINK and MARQUEE elements. These elements are not part of any W3C specification for HTML (i.e., they are non-standard elements).

- Indentation: Use the CSS 'text-indent' property to indent text. Do not use the BLOCKQUOTE or any other structural element to indent text.
- Spacing: Content developers should achieve spacing effects with the CSS 'word-spacing' and 'white-space' properties rather than by putting actual spaces between letters and using the PRE element.

2.10.4 Fonts

Instead of using deprecated presentation elements and attributes [p. 6] , use the many CSS properties to control font characteristics: 'font-family', 'font-size', 'font-size-adjust', 'font-stretch', 'font-style', 'font-variant', and 'font-weight'.

2.10.5 Colors

Use these CSS properties to specify colors:

- 'color', for foreground text color.
- 'background-color', for background colors.
- 'border-color', for border colors.
- For link colors, see the :link, :visited, and :active pseudo-classes.

[Checkpoint 2.1] Ensure that all information conveyed with color is also available without color, for example from context or markup. [Priority 1] Be careful when color is used in references, as in "Please select an item from those listed in green." This reference becomes useless to those who cannot process color.

For example, in this document, examples appear in a different color than the rest of the text. However, that is not enough to identify them as examples, so we precede each one with the word "Example." or "Deprecated example."

Quicktest! To test whether your page passes the text, examine it with a monochrome monitor or colors turned off.

[Checkpoint 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].

Quicktest! To test whether color contrast is sufficient to be read by people with color deficiencies or by those with low resolution monitors, print pages on a black and white printer (with backgrounds and colors appearing in grayscale).

Also try taking the printout and copying it for two or three generations to see how it degrades. This will show you where you need to add redundant cues (example: hyperlinks are usually underlined on Web pages), or whether the cues are two small or indistinct to hold up well.

Also, try setting up a color scheme in your browser that only uses black, white, and the four browser-safe greys and see how your page holds up.

For more information about colors and contrasts, refer to [LIGHTHOUSE] [p. 54] .

2.10.6 Layout, positioning, layering, and alignment

Layout, positioning, layering, and alignment should be done through style sheets (notably by using CSS floats and absolute positioning).

When laying out tabular information, content developers should use tables that are designed for accessibility (see the section on tables [p. 22] . Do not use PRE to create a tabular layout of text.

[Checkpoint 10.3] Until user agents or 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 2] However, until user agents and screen readers are able to handle text presented side-by-side, all tables that lay out text in parallel, word-wrapped columns require a linear text alternative (on the current page or some other). See also the section on tables [p. 22] for additional information on table accessibility.

Provide text equivalents for all images, including invisible or transparent images.

If content developers cannot use style sheets and must use invisible or transparent images to lay out images on the page, they should supply "null" alt-text (alt="") for them.

Deprecated example.

In this example, an image is used to create a carefully defined space between words or graphics. "White space" alt-text is used to prevent the words from running together when the image is not loaded:

```
my poem requires a big space<IMG src="10pttab.gif" alt="   ">here
```

In this next example, an image is used to force a graphic to appear in a certain position:

```
<IMG src="spacer.gif" alt="spacer">
<IMG src="colorfulwheel.gif" alt="The wheel of fortune">
```

2.10.7 Rules and borders

Rules and borders may convey the notion of "separation" to visually enabled users but that meaning cannot be inferred out of a visual context.

While content developers may use HR to create a horizontal rule, they should do so in a way that also conveys the structure in a non-visual way (e.g., by using DIV in conjunction with the "class" attribute).

Example.

```
<DIV class="navigation-bar">
   <HR title="navigation-bar">
   <A rel="Next" href="next.html">[Next page]</A>
   <A rel="Previous" href="previous.html">[Prevous page]</A>
   <A rel="First" href="first.html">[First page]</A>
</DIV>
```

End example.

When using graphics (e.g., horizontal rules) as section separators, content developers may provide an advisory title of what the graphic represents to the visually enabled user via the "title" attribute. Hence, in the previous example, we specified title="navigation-bar".

Example.

In this example, a red line is used to separate Chapter 7 from Chapter 8:

```
<IMG src="redline.gif" alt="redline graphic"
     title="End of Chapter 7 - Visual Displays">
<H1>Chapter 8 - Auditory and Tactile Displays</H1>
```

We recommend using style sheets to accomplish such styling of the line:

```
<HEAD>
<TITLE>Redline with style sheets</TITLE>
<STYLE type="text/css">
     HR.redline { color : red }
</STYLE>
</HEAD>
<BODY>
<HR class="redline" title="End of Chapter 7 - Visual Displays">
<H1>Chapter 8 - Auditory and Tactile Displays</H1>
</BODY>
```

End example.

2.10.8 Guidelines for good CSS style sheets

- Use a minimal number of style sheet for your site
- If you have more than one, use the same "class" name for the same concept in all of the style sheets.
- Use linked style sheets rather than embedded styles, and avoid inline style sheets.
- Content developers should not write "!important" rules. Users should where necessary.
- Use the "em" unit to set font sizes.
- Use relative length units and percentages. CSS allows you to use relative units

even in absolute positioning. Thus, you may position an image to be offset by "3em" from the top of its containing element. This is a fixed distance, but is relative to the current font size, so it scales nicely.

- Only use absolute length units when the physical characteristics of the output medium are known.
- Always specify a fallback generic font.
- Use numbers, not names, for colors.

Some examples follow.

Example.

Use em to set font sizes, as in:

```
H1 { font-size: 2em }
```

rather than:

```
H1 { font-size: 12pt }
```

End example.

Example.

Use relative length units and percentages.

```
BODY { margin-left: 15%; margin-right: 10%}
```

End example.

Example.

Only use absolute length units when the physical characteristics of the output medium are known.

```
.businesscard { font-size: 8pt }
```

End example.

Example.

Always specify a fallback generic font:

```
BODY { font-family: "Gill Sans", sans-serif }
```

End example.

Example.

Use numbers, not names, for colors:

```
H1 {color: #808000}
H1 {color: rgb(50%,50%,0%)}
```

End example.

2.11 Frames

For visually enabled users, frames may organize a page into different zones. For non-visual users, relationships between the content in frames (e.g., one frame has a table of contents, another the contents themselves) must be conveyed through other means.

Frames as implemented today are problematic for several reasons:

- Without scripting, they tend to break the "previous page" functionality offered by browsers.
- It is impossible to refer to the "current state" of a frameset with a URI; once a frameset changes contents, the original URI no longer applies.
- Opening a frame in a new browser window can disorient or simply annoy users.

In the following sections, we discuss how to make frames more accessible. We also provide an alternative to frames [p. 47] that uses HTML 4.0 and CSS and addresses many of the limitations of today's frame implementations.

2.11.1 Title frames for easy orientation

[Checkpoint 12.1] Title each frame so that users can keep track of frames by title. [Priority 1]

Example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>A simple frameset document</TITLE>
</HEAD>
<FRAMESET cols="10%, 90%"
          title="Our library of electronic documents">
    <FRAME src="nav.html" title="Navigation bar">
    <FRAME src="doc.html" title="Documents">
    <NOFRAMES>
       <A href="lib.html" title="Library link">
            Select to go to the electronic library</A>
    </NOFRAMES>
</FRAMESET>
```

End example.

2.11.2 Long descriptions of frames

[Checkpoint 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]

Example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
  <HEAD>
    <TITLE>Today's news</TITLE>
```

```
  </HEAD>

  <FRAMESET COLS="10%,*,10%">

  <FRAMESET ROWS="20%,*">
    <FRAME SRC="promo.html" NAME="promo" title="promotions">
    <FRAME SRC="sitenavbar.html" NAME="navbar"
       title="Sitewide navigation bar" longdesc="frameset-desc.html#navbar">
  </FRAMESET>

  <FRAME SRC="story.html" NAME="story" title="Selected story - main content"
     longdesc="frameset-desc.html#story">

  <FRAMESET ROWS="*,20%">
    <FRAME SRC="headlines.html" NAME="index" title="Index of other
      national headlines" longdesc="frameset-desc.html#headlines">
    <FRAME SRC="ad.html" NAME="adspace" title="Advertising">
  </FRAMESET>

  <NOFRAMES>
    <p><a href="noframes.html">No frames version</a></p>
    <p><a href="frameset-desc.html">Descriptions of frames.</a></p>
  </NOFRAMES>

  </FRAMESET>
</HTML>
```

frameset-desc.html might say something like: #Navbar - this frame provides links to the major sections of the site: World News, National News, Local News, Technological News, and Entertainment News. #Story - this frame displays the currently selected story. #Index - this frame provides links to the day's headline stories within this section.

End example.

Note that if the a frame's contents change, the long descriptions will no longer apply. Also, links to a frame's longdesc ("d-links") ought to be provided with other alternative contents in the NOFRAMES element of a FRAMESET.

2.11.3 Invisible d-links

2.11.4 Ensure documents are readable without frames

Example.

In this example, if the user reads "top.html":

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>This is top.html</TITLE>
</HEAD>
<FRAMESET cols="50%, 50%" title="Our big document">
    <FRAME src="main.html" title="Where the content is displayed">
    <FRAME src="table_of_contents.html" title="Table of Contents">
    <NOFRAMES>
```

```
        <A href="table_of_contents.html">Table of Contents.</A>
     <!-- other navigational links that are available in main.html
          are available here also. -->
    </NOFRAMES>
</FRAMESET>
</HTML>
```

and the user agent is not displaying frames, the user will have access (via a link) to a non-frames version of the same information.

End example.

2.11.5 Always make the source of a frame an HTML document

Content developers must provide descriptions of frames so that their contents and the relationships between frames make sense. Note that as the contents of a frame change, so must change any description. This is not possible if an IMG is inserted directly into a frame, as in this deprecated example:

Deprecated example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>A bad frameset document</TITLE>
</HEAD>
<FRAMESET cols="100%" title="Static frameset">
   <FRAME name="badframe"
          src="apples.gif" title="Apples">
</FRAMESET>
</HTML>
```

Note that if, for example, a link causes a new image to be inserted into the frame:

```
  <P>Visit a beautiful grove of
  <A target="badframe" href="oranges.gif" title="Oranges">oranges</A>
```

the initial title of the frame ("Apples") will no longer match the current content of the frame ("Oranges").

[Checkpoint 6.2] Ensure that descriptions and text alternatives for dynamic content are updated when the dynamic content changes. [Priority 1] To solve this problem, content developers should always make the source ("src") of a frame an HTML file. Images may be inserted into the HTML file and their text alternatives will evolve correctly.

Example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>A correct frameset document</TITLE>
</HEAD>
<FRAMESET cols="100%" title="Evolving frameset">
<FRAME name="goodframe" src="apples.html" title="Apples">
</FRAMESET>
</HTML>
```

```
<!-- In apples.html -->
<P><IMG src="apples.gif" alt="Apples">
```

End example.

2.11.6 Avoid opening a new window as the target of a frame

[Checkpoint 10.1] Until user agents 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] For example, content developers should avoid specifying a new window as the target of a frame with target="_blank".

2.11.7 Alternatives to frames

One of the most common uses of frames is to split the user's browser window into two parts: a navigation window and a content window. As an alternative to frames, we encourage you to try the following:

1. Create one document for the navigation mechanism (call it "nav.html"). A separate document means that the navigation mechanism may be shared by more than one document.
2. In each document requiring the navigation mechanism, include it at the bottom of the document with the following (or similar) OBJECT markup:

   Example.

   ```
   <P>
   <OBJECT data="nav.html">
   Go to the <A href="nav.html">table of contents</A>
   </OBJECT>
   ```

   Putting the navigation mechanism at the end of the document means that when style sheets are turned off, users have access to the document's important information first.
3. Use style sheets to position the navigation mechanism where you want on the screen. For example, the following CSS rule floats the navigation bar to the left of the page and makes it take up 25% of the available horizontal space:

   ```
   OBJECT { float: left; width: 25% }
   ```

   The following CSS rule attaches the navigation mechanism to the bottom-left corner of the page of the page and keeps it there even if the user scrolls down the page:

```
OBJECT { position: fixed; left: 0; bottom: 0 }
```

Note. Navigation mechanisms or other content may be inserted in a document by means of server-side includes.

2.12 Forms

2.12.1 Make controls keyboard accessible

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

[Checkpoint 9.5] Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls. [Priority 3] See the section on keyboard access [p. 10] for more information.

2.12.2 Group form controls

Content developers should group information [p. 13] where natural and appropriate. When form controls can be grouped into logical units, use the FIELDSET element and label those units with the LEGEND element (both available in HTML 4.0):

Example.

```
<FORM action="http://somesite.com/adduser" method="post">
   <FIELDSET>
   <LEGEND>Personal information</LEGEND>
   <LABEL for="firstname">First name: </LABEL>
   <INPUT type="text" id="firstname" tabindex="1">
   <LABEL for="lastname">Last name: </LABEL>
   <INPUT type="text" id="lastname" tabindex="2">
   ...more personal information...
   </FIELDSET>
   <FIELDSET>
   <LEGEND>Medical History</LEGEND>
   ...medical history information...
   </FIELDSET>
</FORM>
```

End example.

2.12.3 Label form controls explicitly

[Checkpoint 12.4] Associate labels explicitly with their controls. [Priority 2] An example of LABEL used with "for" in HTML 4.0 is given in the previous section.

[Checkpoint 10.2] For all form controls with implicitly associated labels, ensure that the label is properly positioned. [Priority 2]

2.12.4 Group menu options

Content developers should group information [p. 13] where natural and appropriate. For long lists of menu selections (which are hard to remember), content developers should group items into a hierarchy using the OPTGROUP element (available in HTML 4.0).

Example.

```
<FORM action="http://somesite.com/prog/someprog" method="post">
 <P>
 <SELECT name="ComOS">
     <OPTGROUP label="PortMaster 3">
       <OPTION label="3.7.1" value="pm3_3.7.1">PortMaster 3 with ComOS 3.7.1
       <OPTION label="3.7" value="pm3_3.7">PortMaster 3 with ComOS 3.7
       <OPTION label="3.5" value="pm3_3.5">PortMaster 3 with ComOS 3.5
     </OPTGROUP>
     <OPTGROUP label="PortMaster 2">
       <OPTION label="3.7" value="pm2_3.7">PortMaster 2 with ComOS 3.7
       <OPTION label="3.5" value="pm2_3.5">PortMaster 2 with ComOS 3.5
     </OPTGROUP>
     <OPTGROUP label="IRX">
       <OPTION label="3.7R" value="IRX_3.7R">IRX with ComOS 3.7R
       <OPTION label="3.5R" value="IRX_3.5R">IRX with ComOS 3.5R
     </OPTGROUP>
 </SELECT>
</FORM>
```

End example.

2.12.5 Techniques for specific controls

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

Example.

```
<FORM action="http://somesite.com/prog/text-read" method="post">
     <P>
     <TEXTAREA name=yourname rows="20" cols="80">
     Please enter your name here.
     </TEXTAREA>
     <INPUT type="submit" value="Send"><INPUT type="reset">
     </P>
</FORM>
```

End example.

Provide alt-text [p. 7] for images used as "submit" buttons:

Example.

```
<FORM action="http://somesite.com/prog/text-read" method="post">
<P>
<INPUT type="image" name=submit src="button.gif" alt="Submit">
</FORM>
```

End example.

Refer also to checkpoint 3.8.

Also see the section on keyboard access [p. 10] since this applies to form controls.

2.12.6 Backwards compatibility issues for forms

The BUTTON element does not appear and <INPUT type="button"> will appear as a text input field in HTML3.2 browsers.

2.13 Scripts

Content developers must ensure that pages are accessible with scripts turned off or in browsers that don't support scripts.

2.13.1 Alternative presentation of scripts

[Checkpoint 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 mechanisms on an alternative accessible page. [Priority 1] One way to accomplish this is with the NOSCRIPT element (available in HTML 4.0). The content of this element is rendered with scripts are not enabled.

Example.

```
<SCRIPT type="text/tcl">
 ...some Tcl script to show a billboard of sports scores...
</SCRIPT>
<NOSCRIPT>
   <P>Results from yesterday's games:</P>
   <DL>
      <DT>Bulls 91,  Sonics 80.
      <DD><A href="bullsonic.html">Bulls vs. Sonics game highlights</A>
      ...more scores...
   </DL>
</NOSCRIPT>
```

End example.

[Checkpoint 6.4] For scripts and applets, until user agents provide device-independent means to activate event handlers, ensure that event handlers are keyboard operable. [Priority 2]

2.13.2 Device-independent event handlers

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

An event handler is a script that is invoked when a certain event occurs (e.g, the mouse moves, a key is pressed, the document is loaded, etc.). In HTML 4.0, event handlers are attached to elements via script attributes [p. 6] (the attributes beginning with "on", as in "onkeyup").

Some event handlers, when invoked, produce purely decorative effects such as highlighting an image or changing the color of an element's text. Other event handlers produce much more substantial effects, such as carrying out a calculation, providing important information to the user, or submitting a form. For event handlers that do more than just change the presentation of an element, content developers should do the following:

1. Use application-level event triggers rather than user interaction-level triggers. In HTML 4.0, application-level event attributes are "onfocus", "onblur" (the opposite of "onfocus"), and "onselect". Note that these attributes are also device-independent.
2. Otherwise, if you must use device-dependent attributes, provide redundant input mechanisms (i.e., specify two handlers for the same element):
   - Use "onmousedown" with "onkeydown".
   - Use "onmouseup" with "onkeyup"
   - Use "onclick" with "onkeypress"

   Note that there is no keyboard equivalent to double-clicking ("ondblclick") in HTML 4.0.
3. Do not write event handlers that rely on mouse coordinates since this prevents device-independent input.

---

3 Acknowledgments

Web Content Guidelines Working Group Co-Chairs:
Chuck Letourneau, Starling Access Services
Gregg Vanderheiden, Trace Research and Development
W3C Team contacts:
Judy Brewer and Daniel Dardailler
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, 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, Jaap van Lelieveld, and Jason White

The original draft of this document is based on "The Unified Web Site Accessibility Guidelines" ([UWSAG] [p. 53]) compiled by the Trace R & D Center at the University of Wisconsin. That document includes a list of additional contributors.

4 Reference specifications

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

[CSS1]
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. The CSS1 Recommendation is available at: 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. The CSS2 Recommendation is available at: 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. The DOM Level 1 Recommendation is available at: 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

[HTML40]
"HTML 4.0 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds., 17 December 1997, revised 24 April 1998. The HTML 4.0 Recommendation is available at: http://www.w3.org/TR/1998/REC-html40-19980424.
The latest version of HTML 4.0 is available at: http://www.w3.org/TR/REC-html40.

[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.

[MATHML]
"Mathematical Markup Language", P. Ion and R. Miner, eds., 7 April 1998. The MathML 1.0 Recommendation is available at: http://www.w3.org/TR/1998/REC-MathML-19980407.
The latest version of MathML 1.0 is available at: http://www.w3.org/TRREC-MathML.

[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.

[RDF]
"Resource Description Framework (RDF) Model and Syntax Specification", O. Lassila, R. Swick, eds., 22 February 1999. The RDF Recommendation is available at: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

The latest version of RDF 1.0 is available at:
http://www.w3.org/TR/REC-rdf-syntax

[SMIL]
"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", P. Hoschka, ed., 15 June 1998. The SMIL 1.0 Recommendation is available at:
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-AUTOOLS]
"Authoring Tool Accessibility Guidelines", J. Treviranus, J. Richards, I. Jacobs, C. McCathieNevile, eds. The latest Working Draft of these guidelines for designing accessible authoring tools is available at:
http://www.w3.org/TR/WD-WAI-AUTOOLS/

[WAI-USERAGENT]
"User Agent Accessibility Guidelines", J. Gunderson and I. Jacobs, eds. The latest Working Draft of these guidelines for designing accessible user agents is available at: http://www.w3.org/TR/WD-WAI-USERAGENT/

[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. This document is available at:
http://www.tracecenter.org/docs/html_guidelines/version8.htm

[XML]
"Extensible Markup Language (XML) 1.0.", T. Bray, J. Paoli, C.M. Sperberg-McQueen, eds., 10 February 1998. The XML 1.0 Recommendation is available at: 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

5 Services

Note. *W3C cannot maintain stability for any of the following references outside of its control. These references are included for convenience.*

[DVS]
DVS Descriptive Video Services.

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

[CSSVAL]
The W3C CSS Validation Service.

[HTMLVAL]
The W3C HTML Validation Service.

[IBMJAVA]
IBM Guidelines for Writing Accessible Applications Using 100% Pure Java are available from IBM Special Needs Systems.

[JAVAACCESS]
Information about Java Accessibility and Usability is available from the Trace R&D Center.
[LIGHTHOUSE]
The Lighthouse provides information about accessible colors and contrasts.
[LYNX]
Lynx is a text-only browser.
[LYNXME]
Lynx-me is a Lynx emulator.
[LYNXVIEW]
Lynx Viewer is a Lynx emulator.
[MACROMEDIA]
Flash OBJECT and EMBED Tag Syntax from Macromedia.
[NCAM]
The National Center for Accessible Media includes information about captioning and audio description on the Web.
[TRACE]
The Trace Research & Development Center. Consult this site for a variet of information about accessibility, including a scrolling Java applet that may be frozen by the user.

---

6 Checkpoint Map

This index lists each checkpoint and the sections in this document where it is discussed.

Guideline 1:

Checkpoint 1.1 [p. 29]
Refer to "1.2 Text equivalents and descriptions [p. 7] " and "2.7.1 Equivalent text for images [p. 29] "
Checkpoint 1.2 [p. 32]
Refer to "1.2 Text equivalents and descriptions [p. 7] " and "2.7.6 Client-side image maps [p. 32] "
Checkpoint 1.3 [p. 37]
Refer to "2.9.2 Visual information and motion [p. 37] "
Checkpoint 1.4 [p. 35]
Refer to "2.8.1 Audio and Video produced by dynamic objects [p. 35] "
Checkpoint 1.5 [p. 30]
Refer to "1.2 Text equivalents and descriptions [p. 7] " and "2.7.3 Ascii art [p. 30] "

Guideline 2:

Checkpoint 2.1 [p. 40]
Refer to "2.10.5 Colors [p. 40] "
Checkpoint 2.2 [p. 40]
Refer to "2.10.5 Colors [p. 40] "

Guideline 3:

Checkpoint 3.1 [p. 19]
Refer to "2.3.4 Text markup rather than images [p. 19] "
Checkpoint 3.2 [p. 17]
Refer to "2.1.2 Section headers [p. 17] "
Checkpoint 3.3 [p. 19]
Refer to "2.4 Lists [p. 19] "
Checkpoint 3.4 [p. 19]
Refer to "2.3.3 Quotations [p. 19] "
Checkpoint 3.5 [p. 16]
Refer to "2.1 Document structure and metadata [p. 16] "
Checkpoint 3.6 [p. 38]
Refer to "1.2 Text equivalents and descriptions [p. 7] " and "2.10 Style and style sheets [p. 38] "
Checkpoint 3.7 [p. 38]
Refer to "2.10.1 General style sheet techniques [p. 38] "
Checkpoint 3.8 [p. 34]
Refer to "1.2 Text equivalents and descriptions [p. 7] " and "2.7.7 Server-side image maps [p. 33] " and "2.12.5 Techniques for specific controls [p. 49] "

Guideline 4:

Checkpoint 4.1 [p. 18]
Refer to "2.2 Language information [p. 18] "
Checkpoint 4.2 [p. 18]
Refer to "2.3.2 Acronyms and abbreviations [p. 18] "
Checkpoint 4.3 [p. 18]
Refer to "2.2 Language information [p. 18] "

Guideline 5:

Checkpoint 5.1 [p. 22]
Refer to "2.5 Tables [p. 22] "
Checkpoint 5.2 [p. 22]
Refer to "2.5 Tables [p. 22] "

Checkpoint 5.3 [p. 27]
Refer to "2.5.3 Avoid tables for layout [p. 27] "
Checkpoint 5.4 [p. 22]
Refer to "2.5 Tables [p. 22] "
Checkpoint 5.5 [p. 22]
Refer to "2.5 Tables [p. 22] "
Checkpoint 5.6 [p. 22]
Refer to "2.5 Tables [p. 22] "

Guideline 6:

Checkpoint 6.1 [p. 38]
Refer to "2.10.1 General style sheet techniques [p. 38] "
Checkpoint 6.2 [p. 46]
Refer to "2.11.5 Always make the source of a frame an HTML document [p. 46] "
Checkpoint 6.3 [p. 50]
Refer to "2.13.1 Alternative presentation of scripts [p. 50] "
Checkpoint 6.4 [p. 50]
Refer to "2.13.1 Alternative presentation of scripts [p. 50] "
Checkpoint 6.5 [p. 9]
Refer to "1.3 Alternative pages [p. 9] " and
"2.11.4 Ensure documents are readable without frames [p. 45] "

Guideline 7:

Checkpoint 7.1 [p. 39]
Refer to "2.10.3 Text style [p. 39] "
Checkpoint 7.2 [p. 39]
Refer to "2.10.3 Text style [p. 39] "
Checkpoint 7.3 [p. 38]
Refer to "2.9.2 Visual information and motion [p. 37] "
Checkpoint 7.4 [p. 14]
Refer to "1.8 Automatic page refresh [p. 14] "
Checkpoint 7.5 [p. 14]
Refer to "1.8 Automatic page refresh [p. 14] "

Guideline 8:

Checkpoint 8.1 [p. 36]
Refer to "2.8.1 Audio and Video produced by dynamic objects [p. 35] " and
"2.8.3 Directly accessible applets [p. 36] "

Guideline 9:

Checkpoint 9.1 [p. 32]
Refer to "2.7.6 Client-side image maps [p. 32] "
Checkpoint 9.2 [p. 11]
Refer to "1.4 Keyboard access [p. 10] "
Checkpoint 9.3 [p. 50]
Refer to "2.13.2 Device-independent event handlers [p. 50] "
Checkpoint 9.4 [p. 48]
Refer to "2.12.1 Make controls keyboard accessible [p. 48] "
Checkpoint 9.5 [p. 48]
Refer to "2.12.1 Make controls keyboard accessible [p. 48] "

Guideline 10:

Checkpoint 10.1 [p. 47]
Refer to "2.11.6 Avoid opening a new window as the target of a frame [p. 47] "
Checkpoint 10.2 [p. 48]
Refer to "2.12.3 Label form controls explicitly [p. 48] "
Checkpoint 10.3 [p. 41]
Refer to "2.10.6 Layout, positioning, layering, and alignment [p. 41] "
Checkpoint 10.4 [p. 49]
Refer to "2.12.5 Techniques for specific controls [p. 49] "
Checkpoint 10.5 [p. 33]
Refer to "2.7.6 Client-side image maps [p. 32] "

Guideline 11:

Checkpoint 11.1 [p. 16]
Refer to "1.11 Browser Support [p. 16] "
Checkpoint 11.2 [p. 5]
Refer to "1.1 Structure vs. Presentation [p. 4] "
Checkpoint 11.3 [p. 14]
Refer to "1.7 Content negotiation [p. 14] "
Checkpoint 11.4 [p. 9]
Refer to "1.3 Alternative pages [p. 9] "

Guideline 12:

Checkpoint 12.1 [p. 44]
Refer to "1.2 Text equivalents and descriptions [p. 7] " and "2.11.1 Title frames for easy orientation [p. 44] "
Checkpoint 12.2 [p. 44]
Refer to "1.2 Text equivalents and descriptions [p. 7] " and "2.11.2 Long descriptions of frames [p. 44] "

Checkpoint 12.3 [p. 13]
Refer to "1.6 Comprehension [p. 12] "
Checkpoint 12.4 [p. 48]
Refer to "2.12.3 Label form controls explicitly [p. 48] "

Guideline 13:

Checkpoint 13.1 [p. 27]
Refer to "2.6 Links [p. 27] "
Checkpoint 13.2 [p. 12]
Refer to "1.5 Navigation [p. 11] "
Checkpoint 13.3 [p. 12]
Refer to "1.5 Navigation [p. 11] "
Checkpoint 13.4 [p. 11]
Refer to "1.5 Navigation [p. 11] "
Checkpoint 13.5 [p. 12]
Refer to "1.5 Navigation [p. 11] "
Checkpoint 13.6 [p. 28]
Refer to "2.6 Links [p. 27] "
Checkpoint 13.7 [p. 12]
Refer to "1.5 Navigation [p. 11] "
Checkpoint 13.8 [p. 12]
Refer to "1.5 Navigation [p. 11] "
Checkpoint 13.9 [p. 15]
Refer to "1.9 Other topics [p. 15] "
Checkpoint 13.10 [p. 30]
Refer to "1.2 Text equivalents and descriptions [p. 7] " and "2.7.3 Ascii art [p. 30] "

Guideline 14:

Checkpoint 14.1 [p. 12]
Refer to "1.6 Comprehension [p. 12] "
Checkpoint 14.2 [p. 12]
Refer to "1.6 Comprehension [p. 12] "
Checkpoint 14.3 [p. 11]
Refer to "1.5 Navigation [p. 11] "

7 Index of HTML elements and attributes

7.1 Elements

This index lists all elements in HTML 4.0 and whether they are defined in earlier versions of HTML. Elements that are deprecated in HTML 4.0 ([HTML40] [p. 52]) are followed by an asterisk (*). Elements that are obsolete in HTML 4.0 or don’t exist in a W3C specification of HTML (2.0, 3.2, 4.0) do not appear in this table. An entry of

"N/A" means that an element is not discussed in this document.

Name (links to 4.0)	Also defined in	Related HTML topics
A	2.0, 3.2	Keyboard access [p. 10] , Links [p. 27]
ABBR		Acronyms and abbreviations [p. 18]
ACRONYM		Acronyms and abbreviations [p. 18]
ADDRESS	2.0, 3.2	Metadata [p. 16]
APPLET*	3.2	Applets and other objects [p. 34]
AREA	3.2	Keyboard access [p. 10] , Images and image maps [p. 28] ,
B	2.0, 3.2	Text style [p. 39] , Emphasis [p. 18]
BASE	2.0, 3.2	Links [p. 27]
BASEFONT*	3.2	Fonts [p. 40]
BDO		N/A
BIG	3.2	Text style [p. 39]
BLOCKQUOTE	2.0, 3.2	Quotations [p. 19] , Text style [p. 39]
BODY	2.0, 3.2	N/A
BR	2.0, 3.2	Layout, positioning, and alignment [p. 41]
BUTTON		Keyboard access [p. 10] , Server-side image maps [p. 33] , Forms [p. 48]
CAPTION	3.2	Tables [p. 22]
CENTER*	3.2	Layout, positioning, and alignment [p. 41]
CITE	2.0, 3.2	N/A
CODE	2.0, 3.2	N/A
COL		Tables [p. 22]
COLGROUP		Tables [p. 22]
DD	2.0, 3.2	Lists [p. 19]
DEL		N/A
DFN	3.2	N/A

DIR*	2.0, 3.2	N/A
DIV	3.2	N/A
DL	2.0, 3.2	Lists [p. 19]
DT	2.0, 3.2	Lists [p. 19]
EM	2.0, 3.2	Emphasis [p. 18] , Text style [p. 39]
FIELDSET		Grouping form controls [p. 48]
FONT*	3.2	Fonts [p. 40]
FORM	2.0, 3.2	Forms [p. 48]
FRAME		Frames [p. 44]
FRAMESET		Frames [p. 44]
H1-H6	2.0, 3.2	Section headers [p. 17]
HEAD	2.0, 3.2	N/A
HR	2.0, 3.2	Rules and border [p. 41] , Section headers [p. 17]
HTML	2.0, 3.2	N/A
I	2.0, 3.2	Text style [p. 39] , Emphasis [p. 18]
IFRAME		Frames [p. 44]
IMG	2.0, 3.2	List bullets [p. 21] ,
INPUT	2.0, 3.2	Images and image maps [p. 28] , Keyboard access [p. 10] , Layout, positioning, and alignment [p. 41] , Rules and borders [p. 41]
INS		N/A
ISINDEX*	2.0, 3.2	N/A
KBD	2.0, 3.2	N/A
LABEL		Keyboard access [p. 10] , Form labels [p. 48]
LEGEND		Keyboard access [p. 10] ,
LI	2.0, 3.2	Lists [p. 19]
LINK	2.0, 3.2	Links [p. 27] , Metadata [p. 16]
MAP	3.2	Images and image maps [p. 28]

MENU*	2.0, 3.2	N/A
META	2.0, 3.2	Metadata [p. 16]
NOFRAMES		Frames [p. 44]
NOSCRIPT		Scripts [p. 50]
OBJECT		Keyboard access [p. 10] , Images and image maps [p. 28] , Applets and other objects [p. 34]
OL	2.0, 3.2	Lists [p. 19]
OPTGROUP		Forms [p. 48]
OPTION	2.0, 3.2	Forms [p. 48]
P	2.0, 3.2	N/A
PARAM	3.2	N/A
PRE	2.0, 3.2	Layout, positioning, and alignment [p. 41] , Text style [p. 39]
Q		Quotations [p. 19]
S		Text style [p. 39]
SAMP	2.0, 3.2	N/A
SCRIPT	3.2 (place-holder)	Scripts [p. 50]
SELECT	2.0, 3.2	Keyboard access [p. 10] , Forms [p. 48]
SMALL	3.2	Text style [p. 39]
SPAN		N/A
STRIKE*	3.2	Text style [p. 39]
STRONG	2.0, 3.2	Emphasis [p. 18] , Text style [p. 39]
STYLE	3.2 (place-holder)	Keyboard access [p. 38] ,
SUB	3.2	N/A
SUP	3.2	N/A
TABLE	3.2	Tables [p. 22]
TBODY		Tables [p. 22]
TD	3.2	Tables [p. 22]

TEXTAREA	2.0, 3.2	Keyboard access [p. 10] , Forms [p. 48]
TFOOT		Tables [p. 22]
TD	3.2	Tables [p. 22]
TD	3.2	Tables [p. 22]
THEAD		Tables [p. 22]
TITLE	2.0, 3.2	Metadata [p. 16]
TR	3.2	Tables [p. 22]
TT	2.0, 3.2	Text style [p. 39]
U	3.2	Text style [p. 39]
UL	2.0, 3.2	Lists [p. 19]
VAR	2.0, 3.2	N/A

7.2 Attributes

This index lists some attributes in HTML 4.0 that affect accessibility and what elements they apply to. Elements that are deprecated in HTML 4.0 ([HTML40] [p. 52]) are followed by an asterisk (*). Elements that are obsolete in HTML 4.0 or don't exist in a W3C specification of HTML (2.0, 3.2, 4.0) do not appear in this table. Attributes that apply to most elements of HTML 4.0 are indicated as such; please consult the HTML 4.0 specification for the exact list of elements with this attribute.

Attributes that affect presentation [p. 6] , most of which are deprecated in HTML 4.0, are not are not listed in this table. Content developers should use style sheets instead of these attributes.

Attributes that associate scripts with events [p. 6] are not are not listed in this table. Please see the section on device-independent event handlers [p. 50] for more detail.

Name (links to HTML 4.0)	Applies to elements	Related HTML topics
abbr	TD, TH	
accesskey	A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA	
alt	APPLET*, AREA, IMG, INPUT	
axis	TD, TH	
class	Most elements	
for	LABEL	
headers	TD, TH	
hreflang	A, LINK	
id	Most elements	
lang	Most elements	
longdesc	IMG, FRAME, IFRAME	
name	FRAME (and others)	
scope	TD, TH	
style	Most elements	
summary	TABLE	
tabindex	A, AREA, BUTTON, INPUT, OBJECT, SELECT, TEXTAREA	
title	Most elements	
usemap	IMG, INPUT, OBJECT	