

WAI Accessibility Guidelines: User Agent

W3C Working Draft 18-June-1998

This version:

<http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19980618>

Latest version:

<http://www.w3.org/TR/WD-WAI-USERAGENT>

Editors:

Jon Gunderson, University of Illinois at Urbana/Champaign <jongund@uiuc.edu>

Ian Jacobs <ij@w3.org>

Copyright:

Copyright © 1998 W3C (MIT, INRIA, Keio), All Rights Reserved.

Please see the Acknowledgments [p. 18] for a complete list of contributors.

Abstract

This document provides guidelines to user agent manufacturers (producers of browsers, players, etc.) for making their products more accessible to persons with disabilities. These guidelines are intended to improve Web browsing for able-bodied users as well.

In a later version of this document, developers will find a helpful checklist for identifying and prioritizing accessibility features.

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 WAI User Agent (UA) working group.

This document has been produced as part of the W3C WAI Activity, and is intended as a draft of a Proposed Recommendation for how to improve user agent accessibility. The goals of the WAI-UA Working Group are discussed in the WAI UA charter. A list of the current Working Group members is available.

Available formats

This document is available in the following formats:

HTML:

<http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19980618> [p. 1]

A plain text file:

<http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19980618/wai-useragent.txt>,

HTML as a gzip'ed tar file:

<http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19980618/wai-useragent.tgz>,

HTML as a zip file (this is a '.zip' file not an '.exe'):

<http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19980618/wai-useragent.zip>,

A PostScript file:

<http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19980618/wai-useragent.ps>,

A PDF file:

<http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19980618/wai-useragent.pdf>.

In case of a discrepancy between the various formats of the specification, <http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19980618> is considered the definitive version.

Comments

Please send comments about this document to the public mailing list:
w3c-wai-ua@w3.org.

Table of Contents

1	Rating and Classification	.5
2	General principles of accessible design	.5
3	Terms and definitions	.5
3.1	Documents, Elements, and Attributes	.5
3.2	Properties, Values, and Defaults	.6
3.3	Selection, Focus, and Events	.6
3.4	Alternate Text and Description Links	.7
3.5	Dependent and Independent User Agents	.7
4	Presentation Adjustability Guidelines	.7
4.1	Control over Browser Defaults and Author Styles	.7
4.2	Alternative Representations	.8
4.3	Alternative Representations of Images and Image Maps	.8
4.4	Alternative Representations of Audio, Video, Movies, and Animations	.9
4.5	Alternative Representations of Embedded Applications	.9
4.6	Alternative Representations of Tables	10
4.7	Alternative Views of a Document	10
5	Orientation Guidelines	10
5.1	Maintenance of Document View and Focus	10
5.2	Document Summary Information	11
5.3	Element and Event Identification	11
5.4	Title Information	12
6	Navigation and Control Guidelines	12
6.1	Sequential Navigation within a View	12
6.2	Hierarchical Navigation within a View	12
6.3	View Navigation	12
6.4	Page Navigation	12
6.5	Direct Access Navigation	13
6.6	Table Navigation	13
6.7	Elements with Associated (DHTML) Event Behavior	13
7	Visibility of Accessibility Features Guidelines	13
7.1	Configuration of Accessibility Features	13
7.2	Menu Commands	13
7.3	Documentation	14
8	Compatibility Guidelines	14
8.1	Cascading Style Sheets	14
8.2	Compatibility with HTML 4.0	15
8.3	Compatibility with Third-party Assistive Technology	15
	Standard OS Controls/Menus/Dialog boxes	16
	Built-in Accessibility Options	16
	Using Accessibility Application Programming Interfaces	16
9	Appendix A: Built-in Accessibility Features of Some Current Operating Systems	16
9.1	Microsoft Windows 95 and Windows NT 4.0	16
9.2	Apple Macintosh	17
9.3	AccessX/SUN Solaris 2.x	17
10	Appendix B: Accessibility APIs for Various Operating Systems	18

10.1 Microsoft Active Accessibility in Windows 95/NT versions	18
10.2 SUNSoft Java Accessibility API in Java Code	18
11 Appendix C: Closed Captioning Resources	18
11.1 The following is a list of resources for closed captioning information and service	18
12 Acknowledgments	18
13 References	19

1 Rating and Classification

Each guideline is classified according to the following rating system:

[Priority 1]

This guideline **must** be implemented by a browser, or one or more groups of users will find it impossible to access some information in a document. Implementing this guideline will significantly improve access to WWW documents.

[Priority 2]

This guideline **should** be implemented by a browser, or one or more groups of users will find it difficult to access some information in a document. Implementing this guideline will improve access to WWW documents.

[Priority 3]

This guideline **should** be implemented by a browser to make it easier for one or more groups of users to access some information in a document. Implementing this guideline is not critical to accessibility, however.

2 General principles of accessible design

The guidelines in this document have been organized around the following general principles of accessible browser design:

1. Allow the user to customize the presentation of documents to meet special needs (e.g., only use large fonts, use certain color combinations, present selected information or the focus in a specific way, etc.).
2. Allow users to override author-specified styles and browser default styles.
3. Provide access to alternate representations of all information ("alt" text of images, transcripts of video, etc.) so that a page may be used with or without vision as well as with or without hearing.
4. Provide tools to navigate the document: from link to link, form control to form control, frame to frame, etc. Allow navigation through keyboard at all times.
5. Provide tools to improve orientation within the document so the user can quickly grasp content and context.
6. Render a page so that other software -- generically called "third-party assistive technology" -- may interpret the rendered page in a manner useful to the user. A screen reader is one example of such technology; it synthesizes rendered lines of text in succession as speech.

3 Terms and definitions

This document does not specify how browsers should implement the guidelines listed herein (although in some instances, sample implementation strategies are indicated). The guidelines do depend, however, on the following document and user agent characteristics.

3.1 Documents, Elements, and Attributes

A document is a series of *elements* that are defined by a language (e.g., HTML 4.0 or an XML application). Each element consists of a name that identifies the type of element, optional *attributes* that take values, and (possibly empty) content. Some attributes are integral to document accessibility (e.g., the "alt", "title", and "longdesc" attributes in

HTML).

The *rendered content* is that which an element actually causes to be rendered by the user agent. This may differ from the element's structural content. For example, some elements cause external data to be rendered (e.g., the IMG element in HTML), and in some cases, browsers may render the value of an attribute (e.g., "alt", "title") in place of the element's content.

3.2 Properties, Values, and Defaults

The presentation of a document is described by *properties* (e.g., font face, font sizes for different headers, paragraph justification, text color, etc.).

Each property has a *current value* at any moment (e.g., "Helvetica" for font face, "12 point" for font size, "black" for text color, etc.) The current value comes from one of the following sources: browser, document, or user.

The value given to a property when the browser is first "turned on" is called the *default value*. Browsers may allow users to change default values through a variety of mechanisms (e.g., the user interface, style sheets, initialization files, etc.). Setting the current value of a property does not change the default value.

A property may receive its current value when the user agent reads a document: through style sheets within the document or linked externally, through the presentation attributes of an element, via a server, etc. These values are called *author styles*.

Finally, the user may set the current value through user style sheets or the user interface; these are called *user styles*.

3.3 Selection, Focus, and Events

Browsers allow users to interact with the rendered document through several mechanisms.

A browser may offer several *views* of the same document. For instance, one view may show a table of contents and a second the actual contents. A user accesses the information in a view through a window, frame, or other *viewport*. The size of the viewport may prevent the user from accessing all of the information in the view at once; viewports may offer scrolling mechanisms to give access to all of the view's information.

A *selection* is a set of elements identified for a particular operation. The user selection identifies a set of elements for certain types of user interaction (e.g., cut, copy, and paste operations). The user selection may be established by the user (e.g., by a pointing device or the keyboard) or via an accessibility API. A view may have several selections, but only one user selection. When several views co-exist, each may have a user selection, but only the *current user selection* is active.

Selections may be rendered specially (e.g., visually highlighted). Highlighted text is often used by third-party assistive technologies to indicate through speech or Braille output what the user wants read. Most screen readers are sensitive to highlight colors. Third-party [p. 15] assistive technologies may provide alternative presentation of the selection through speech, enlargement, or dynamic Braille display.

The *focus* designates the element (link, form control, element with associated scripts) in a view [p. 6] that will react when the user next interacts with the document. The user is said to *activate* the element with the focus. For instance, a user may activate a link (which usually causes the user agent to follow the link), a form control (which usually causes a change of state in the form control), or an element with associated scripts. A view has only one focus. When several views co-exist, each may have a focus, but only the *current focus* is active.

Both the current focus and the current user selection must be in the same view, called the *current view*.

Events, which may occur to a document or part of a document, cause the browser to behave in a certain way. For instance, an event occurs when a document is loaded into the browser or unloaded, when a mouse button is depressed or released (when released, the browser performs the action specified by the button), when a link is activated (it is generally followed), etc. Events may occur with or without user action, especially with documents using "Dynamic HTML" (DHTML) and scripting programs.

3.4 Alternate Text and Description Links

Certain types of content may not be accessible to all users (e.g., images), so user agents should ensure that *alternate textual representations* ("Alt-text") of information be available to the user. Alternate text can come from element content (e.g., the OBJECT element) or attributes (e.g., "alt" or "title"). When not supplied by the author, the user agent should generate alternate text. Approaches in such cases may vary. For instance, the user agent may simply substitute the word "IMAGE" or "APPLET" for the object in question. More sophisticated (but potentially costly) solutions in the case of an image that is content of a link involve following the link and retrieving the title of the target document.

A description link, or *D-Link*, is an author-supplied link to additional information about a piece of content that might otherwise be difficult to access (image, applet, video, etc.). The "WAI Accessibility Guidelines: Page Authoring" document (see [WAI-PAGEAUTH] [p. 19]) describes when and how to insert description links in a document.

D-links should be identified in the document source by giving the "rel" attribute the value "dlink".

3.5 Dependent and Independent User Agents

This document distinguishes two types of user agents as they relate to accessibility and calls them *dependent* and *independent* user agents.

A dependent user agent relies on the rendering of another user agent to perform its tasks. Dependent user agents include screen magnifiers and screen readers, both of which rely on the visual output of another user agent.

An independent user agent only requires the source document (and associated style sheets, etc.) to perform its tasks. It may use an accessibility API to retrieve information about a document.

Most of the guidelines in this document apply to dependent user agents.

Note. Future versions of this document may make clearer distinctions about which guidelines are meant for which type of user agent.

4 Presentation Adjustability Guidelines

4.1 Control over Browser Defaults and Author Styles

1. [PRIORITY 1]

Allow the user to override author styles [p. 6] and browser defaults [p. 6] for the following properties [p. 6] : font face, font size, foreground and background colors, background images, selection [p. 6] highlight colors, and other properties

related to user interaction (e.g., hover styles).

2. [PRIORITY 3]

Allow the user, through a keyboard command, to switch between browser default values [p. 6] and user styles [p. 6] .

3. [PRIORITY 3]

Allow the user to specify custom settings in profiles [p. 14] that may be shared. Preferably, this should be done via style sheets [p. 14] . Furthermore, for convenience, users should be able to name groups of settings and be able to apply them all at once (e.g., by selecting a group by name from a menu). This should also be achieved with style sheets.

4.2 Alternative Representations

1. [PRIORITY 2]

or [PRIORITY 3]

Allow the user to specify how and where to render alternate representations [p. 7] of information.

Users should be allowed to select whether they want the user agent to render the principal content (e.g., an image or applet), alternative representations (e.g., a long description), or both.

Users should also be allowed to specify where the information should be rendered: inline or in a separate view. For instance, users with low vision may want to look at an image, but may require that the image be rendered in a separate view. Or, users may require that a long description be rendered inline (as a D-link [p. 7] or in a separate view).

Similarly, users with certain disabilities may want both embedded applications and descriptive text information to be rendered simultaneously.

2. [PRIORITY 3]

Allow users to turn on/off the display of D-links [p. 7] in a document.

4.3 Alternative Representations of Images and Image Maps

1. [PRIORITY 1]

Allow the user to turn off the display of images in favor of descriptive text.

In the case of the IMG element (see [HTML40] [p. 19]), there are two potential sources of short descriptive text (in order of preference): the "alt" and "title" attributes. The latter may be used as a "tool tip" (information displayed when the user hovers over an element with a pointing device).

In the case of the OBJECT element, the text content of the OBJECT is considered its alternative text (see [HTML40] [p. 19] , section 13.3.1 for complete rules about rendering OBJECT elements, embedded OBJECT elements, etc.). If the element has no content, the value of the "title" attribute should be used.

The entire descriptive text should be rendered, whatever its source or the dimensions specified for the original image. Rendered text should be wrapped so the user doesn't need to scroll horizontally to read the description.

If the author has not supplied alternative content with an element, the browser may render the word "Image".

2. [PRIORITY 1]

When an IMG element has a value for the "longdesc" attribute and the user has turned off the display of images, give the user access to the long description.

If the long description is rendered as a D-link [p. 7] , provide keyboard access to locate and activate [p. 6] it (in addition to pointer access for able-bodied users).

Note. If the author has already provided a D-link [p. 7] for the image, the user agent should not render the long description specified by "longdesc" as a D-link.

3. [PRIORITY 1]

Allow users to access alternate text supplied for image maps. User agents must give users access to the value of the "alt" attribute for the AREA element.

4.4 Alternative Representations of Audio, Video, Movies, and Animations

1. [PRIORITY 1]

Allow the user to turn on/off audio descriptions of videos, movies, and animations. User agents should notify multimedia players that they should play audio descriptions of visual actions and events.

2. [PRIORITY 1]

Allow the user to turn on/off closed captioning [p. 18] of audio, video, movies, and animations. User agents should notify multimedia players that they should display captioning information. If a mechanism is available in the operating system to indicate the user's need for closed captioning, the browser should obey the system-level setting (e.g., "show sounds" in Windows 95/NT).

4.5 Alternative Representations of Embedded Applications

1. [PRIORITY 1]

Allow the user to turn off the presentation of embedded applications in favor of descriptive text. Users with some disabilities may not be able to use an application, but need to know about its existence, purpose, and function.

In the case of the OBJECT element, the text content of the OBJECT is considered its alternative text (see [HTML40] [p. 19] , section 13.3.1 for complete rules about rendering OBJECT elements, embedded OBJECT elements, etc.). If the element has no content, the value of the "title" attribute should be used.

In the case of the APPLETT element, the values of the "alt" or "title" attributes (in order of preference) should be used. **Note.** The APPLETT element is *deprecated* in HTML 4.0 (see [HTML40] [p. 19]) and authors should use the OBJECT element instead.

For either element, the entire descriptive text should be rendered, whatever its source or the dimensions specified for the original image. Rendered text should be wrapped so the user doesn't need to scroll horizontally to read the description.

If the author has not supplied alternative content the browser may display the word "APPLETT".

If an element specifying an application could not be loaded, the browser should provide the user with an explanation (e.g., missing object, server not ready, etc.).

Note. If applets written in Java have been designed for accessibility (see SUNSoft Java Accessibility [p. 18] in the Appendix), the user agent should provide that access.

4.6 Alternative Representations of Tables

1. [PRIORITY 2]

Give access to table summary information. User agents must give users access to the value of the "summary" attribute of the TABLE element.

2. [PRIORITY 2]

Allow the user to create and control serialized views of tables.

Tabular information can confuse users with certain dependent user agents [p. 7] . User agents should be able to serialize the table -- render it one cell at a time -- to reduce confusion.

Users should be able to specify whether they want the cells rendered row by row or column by column.

Users may choose different mechanisms for rendering cell header information (e.g., render row and column header information before each cell, render row header information once at the beginning of the row, etc.) The HTML 4.0 specification (see [HTML40], chapter 11 and in particular section 11.4.3 [p. 19]) and the CSS 2 specification (see [CSS2], chapters 12, 17, and 19 [p. 19]) describe mechanisms for structuring tables, identifying headers, and rendering them in accessible ways.

See also the section on table navigation [p. 13] for related information.

Note. Table serialization is important for tables used to structure tabular information as well as those used to lay out blocks of information.

4.7 Alternative Views of a Document

1. [PRIORITY 2]

Provide a hierarchical view of key structural elements of a page. For example, when used properly, header and list elements may be used to create an outline of major topics (see also hierarchical navigation [p. 12]).

If the browser provides more than one view [p. 6] , the user should be able to toggle between alternative views of the document. Selections [p. 6] between views should be synchronized.

2. [PRIORITY 2]

Provide a "serialized", text-only view of page. This means no images, tables, applets, or anything that cannot be rendered as a stream of characters. All non-textual information should be minimized (i.e., made available but discrete) and alternative text should replace images, applets, etc.

5 Orientation Guidelines

5.1 Maintenance of Document View and Focus

1. [PRIORITY 1]

Provide a mechanism for highlighting and identifying the current view [p. 7] , focus [p. 6] , and user selection [p. 6] . For example, if the current view is in a frame, the frame border might have a special border or its title rendered. Or, the user should be able to identify the focus with a thick-bordered box, by changing the foreground and/or background colors, by causing it to appear at the top of the page, by changing the size of the element's content, etc. Style sheets should be used to realize this highlighting (see, for example, CSS2 element focus outlines in [CSS2]

[p. 19] , section 18.4).

This is particularly important for third-party assistive technologies (see the section on compatibility [p. 14] for more information.)

2. [PRIORITY 1]

Allow the user to specify that a view's focus [p. 6] should follow changes in the viewport [p. 6] . For example, as the user scrolls down the page, the focus might jump to the first visible element in the viewport that may take the focus. Thus, after browsing, if the user uses keyboard commands to move or select the focused element, the viewport does not abruptly change to another portion of the document with the focused element.

3. [PRIORITY 3]

Allow the user to turn on/off automatic page forwarding. The META element may be used to forward users from page to another, but this is very disorienting for users with screen readers. Instead, for examples, user agents may render META elements as ordinary links to another document.

5.2 Document Summary Information

1. [PRIORITY 2]

When a page is loaded and on user demand, make available document summary information. Such information includes the size of the document, the number of structural elements related to the document, etc. The information might be displayed on a status line or in a separate and consistent window.

2. [PRIORITY 2]

Provide the user with information about how much of the document has been viewed. This information may be a percentage of how much of the document has been read (i.e., up to the information at the beginning of the current view). For paging user agents, it may also be the number of the current page with respect to the total number of pages.

5.3 Element and Event Identification

1. [PRIORITY 1]

Provide a mechanism for assistive technologies to identify which elements have associated DHTML events [p. 7] . This should be done by exposing DHTML events through accessibility APIs [p. 18] (and is not meant to be achieved through a visual effect).

2. [PRIORITY 2]

Render information about elements [p. 5] and DHTML events [p. 7] when certain events occur (e.g., focus [p. 6] , hover, etc.). This may be done in a variety of ways: audible alerts, status line information, etc.

Note. At the writing of this document, there is no general and standard mechanism for associating style with events. CSS may be used, however, for certain events (namely hover, focus, and link activation).

3. [PRIORITY 3]

Provide a mechanism to distinguish visited links from unvisited links. This may be done with style sheets.

4. [PRIORITY 3]

Allow the user to specify that images used in links must have borders. This may be done with style sheets.

5.4 Title Information

1. [PRIORITY 2]
Make the document TITLE available to the user.

6 Navigation and Control Guidelines

To navigate a document may involve displaying different parts of it (e.g., by scrolling) or shifting focus [p. 6] to different elements [p. 5] . One of the key issues related to navigation and control is the ability to use the keyboard to access all links, form controls and DHTML events. This includes the emulation of DHTML-based mouse events.

6.1 Sequential Navigation within a View

1. [PRIORITY 1]
Allow the user to use keyboard commands to move sequentially from one link, element with "longdesc", or form control to the next in the same view [p. 6] .

6.2 Hierarchical Navigation within a View

1. [PRIORITY 2]
or [PRIORITY 3]
Allow the user to use the keyboard to navigate within a hierarchical view [p. 10] of the document.
The user should be able to use keyboard or pointing input commands to navigate, expand, or contract the hierarchy.
The user should be able to navigate specifically those elements related as parents/children. The user should also be able to navigate specifically those elements related as siblings.
The focus should be highlighted within the hierarchy in a way that is compatible with third-party assistive technology (see the section on compatibility [p. 14]).

6.3 View Navigation

1. [PRIORITY 1]
Allow the user to use the keyboard to navigate views [p. 6] (notably frames).
Navigating into a view makes it the current view [p. 7] .
2. [PRIORITY 1]
Maintain state information so that whenever a view becomes the current view [p. 7] , it's previous focus and user selection become the current focus [p. 6] and current selection [p. 6] , respectively.

6.4 Page Navigation

1. [PRIORITY 2]
Allow the user to use the keyboard to browse a history of visited documents, and to select and visit a document on the list.

6.5 Direct Access Navigation

1. [PRIORITY 2]
Allow the user to use the keyboard to move the focus [p. 6] directly to links and controls on a page. Users should be able to move the focus to links and controls in two ways:
 1. By searching for an element's text or Alt-text.
 2. By identifying the numerical position of an element in the document (source order, not rendered order).
2. [PRIORITY 2]
Allow the user to use the keyboard to move the selection [p. 6] directly to elements [p. 5] that are not links or form controls.

6.6 Table Navigation

1. [PRIORITY 2]
Allow the user to use the keyboard to move the selection [p. 6] between cells in a table (notably left/right within a row and up/down within a column).
A table navigation mechanism should also provide the user with information about the current table cell (see the guidelines on alternative representations of tables [p. 10]).

6.7 Elements with Associated (DHTML) Event Behavior

1. [PRIORITY 1]
Allow the user to use keyboard commands to browse a list of elements and their associated DHTML events [p. 7] , and to select and execute an event on the list.

7 Visibility of Accessibility Features Guidelines

7.1 Configuration of Accessibility Features

1. [PRIORITY 2]
Make available to users a central dialog box for the configuration of accessibility features.
2. [PRIORITY 2]
Furnish predefined accessibility profiles [p. 14] for common disabilities. Sample profiles will serve as models and may be copied and fine-tuned to meet an individual's particular needs. As much accessibility configuration as possible should be done through CSS, if the user agent supports CSS.

7.2 Menu Commands

1. [PRIORITY 2]
Display keyboard shortcut navigation commands in customizable menus.
2. [PRIORITY 2]
If menu options exist for element navigation, these menus should also be used to display summary information for elements reached through that menu.

For example, if a browser has a menu item labeled "Headers", that menu should allow navigation among header elements. The display of the "Headers" menu could read "N Headers" where "N" is the number of headers in the current document. The sub-menu items might include "Next Header", "Previous Header", "All Headers", and a dynamically created list of the first 10 headers in the document.

7.3 Documentation

1. [PRIORITY 2]

Provide a description of accessibility features in the on-line documentation. The on-line documentation needs to include information on features that can be used to improve the usability of the browser by persons with disabilities and provide a list of all keyboard commands. The search index should include references to pertinent keywords (e.g., "disability", "handicapped", "accessibility", "impairment", "keyboard" and "shortcut").

Obviously, the tool used to consult on-line documentation should be accessible to users with disabilities.

2. [PRIORITY 2]

Provide a description of accessibility in printed documentation. If printed manuals are distributed with the browser, they should include information about accessibility. The table of contents and the index should have entries that clearly identify disability access features (see the keywords of the previous guideline).

3. [PRIORITY 2]

Print and on-line information should be available in alternative formats for people with print impairments. This includes large print, accessible electronic formats, audio tape, and Braille. Information on how to obtain information in alternative formats should be available in both on-line and print materials.

8 Compatibility Guidelines

8.1 Cascading Style Sheets

The following guidelines apply to user agents that implement Cascading Style Sheets (see CSS, level 1 [p. 19] and CSS, level 2 [p. 19]). Cascading Style Sheets may be part of a source document or linked externally.

Stand-alone style sheets are useful for implementing *user profiles* in public access computer environments where several people use the same computer. User profiles allow for convenient customization and may be shared by a group.

Overlapping rules from author, user, and browser style sheets cascade (i.e., combine) to produce a single rule that assigns a current value [p. 6] to a property.

1. [PRIORITY 1]

Allow the user to turn off author styles [p. 6] represented by author style sheets.

2. [PRIORITY 1]

Allow the user to adjust default values [p. 6] represented by browser style sheets.

3. [PRIORITY 1]

Support the :before and :after pseudo-elements as defined in CSS2 ([CSS2] [p. 19], section 12.1). These pseudo-elements generate content that can help orient the user by identifying the element that is being spoken or presented in Braille by

third-party assistive technology.

4. [PRIORITY 1]
Support the 'outline' property as defined in CSS2 ([CSS2] [p. 19] , section 18.4). Outlines may be used to customize the focus display (see also orientation information [p. 10]).
5. [PRIORITY 2]
Allow the user to specify user styles [p. 6] through style sheets (see [CSS2] [p. 19] , section 6.4).
6. [PRIORITY 2]
Implement the !important rule as defined in CSS2 ([CSS2] [p. 19] , section 6.4.2). These rules offer a way for users to override author styles [p. 6] and browser defaults [p. 6]
7. [PRIORITY 2]
Support aural cascading style sheets (see [CSS2] [p. 19] , chapter 19) for the auditory presentation of documents.

8.2 Compatibility with HTML 4.0

1. [PRIORITY 1]
Support the "longdesc" attribute defined for IMG elements ([HTML 4.0] [p. 19] , section 13.2). This attribute may be used to attach additional descriptive information to images.
2. [PRIORITY 2]
Support the "rel" and "rev" attributes defined for the A and LINK elements ([HTML40] [p. 19] , section 12.1.2). These attributes may be used to identify D-links [p. 7] and other document relations pertinent for accessibility.
3. [PRIORITY 2]
Support the CAPTION element ([HTML40] [p. 19] , section 11.2.2) for rich table captions.
4. [PRIORITY 2]
Support the "summary" attribute for TABLE ([HTML40] [p. 19] , section 11.2.1) for table summary information.
5. [PRIORITY 2]
Support the NOSCRIPT and NOFRAMES elements ([HTML40] [p. 19] , section 18.3.1 and 16.4.1) for accessible alternatives to scripts and frames.
6. [PRIORITY 3]
Support the "tabindex" attribute ([HTML40] [p. 19] , section 17.11.1) for assigning the order of keyboard navigation within a document.
7. [PRIORITY 3]
Support the "accesskey" attribute ([HTML40] [p. 19] , section 17.11.2) for assigning keyboard commands to active [p. 6] elements such as links, and form controls.

8.3 Compatibility with Third-party Assistive Technology

Standard OS Controls/Menus/Dialog boxes

1. [PRIORITY 1]

Use standard rather than custom controls when designing browsers. Third-party assistive technology developers are more likely able to access standard controls than custom controls. If you must use custom controls, review them for accessibility and compatibility with third-party assistive technology.

Built-in Accessibility Options

1. [PRIORITY 1]

Ensure that products are compatible with operating system built-in accessibility features. Built-in features typically found in operating systems include: sticky keys, filter keys, toggle keys, high contrast screen colors, system font size and face, and mouse keys. Developers should test their programs with the various features to ensure that their technology is compatible. See Appendix A [p. 16] for a list of features available in several current operating systems.

Using Accessibility Application Programming Interfaces

1. [PRIORITY 1]

Support operating system accessibility application programming interfaces (APIs). Accessibility APIs are designed to provide a bridge between the standard user interface supported by the operating system and alternative user interfaces developed by third-party assistive technology vendors to provide access to persons with disabilities. Applications supporting these APIs are therefore generally more compatible with third-party assistive technology. A list of currently available accessibility APIs can be found in Appendix B [p. 18].

9 Appendix A: Built-in Accessibility Features of Some Current Operating Systems

The following appendix is informative only.

Many major operating systems have built-in accessibility features for improving the usability of the standard operating system by persons with disabilities. When designing an application program, developers should test to see if their product is compatible with the features in the target operating system. This should not be a problem if developers use standard development tools and standard software design practices.

9.1 Microsoft Windows 95 and Windows NT 4.0

The accessibility options can be adjusted from the control panel.

- **Sticky Keys:** Allows user to temporarily hold down the shift, alt, and control keys to allow one finger typing.
- **Filter Keys:** Allows user to change keyboard timing to accept a keypress and repeat rate.
- **Toggle Keys:** Provides auditory feedback when modifier or lock keys are pressed.
- **Mouse Keys:** Allows the user to emulate pointer movement and button press operations using the numeric keypad.

- **High Contrast Display Mode:** Changes the colors used on the display to a high-contrast color combination.
- **Sound Sentry:** Monitors the system sounds and flashes the menu bar when sound output is detected. Used by the hearing impaired to know when sound is being generated by the system.
- **Show Sounds:** A flag that is available to applications (including browsers) to notify them that audio information should be presented in a visual form for the hearing impaired. This requires the application program to provide the alternative format of information. This can include closed captioning information on animations and video clips.

9.2 Apple Macintosh

The accessibility options can be adjusted from the control panels through the Easy Access option and the Closeview option.

- **Sticky Keys:** Allows user to temporarily hold down the shift, open apple, and control keys to allow one finger typing.
- **Slow Keys:** Allows user to change keyboard timing to accept a keypress and repeat rate.
- **Mouse Keys:** Allows the user to emulate pointer movement and button press operations using the numeric keypad.
- **Closeview:** Closeview is a screen enlargement and enhancement program used by persons with low vision to magnify the information on the visual display and change the colors used by the system.

9.3 AccessX/SUN Solaris 2.x

Disability access server features, known as AccessX, provide basic workstation accessibility, typically used by people with mobility impairments. AccessX became a supported part of the X Windows server in version X11/R6. The built-in server level access features include:

- **StickyKeys:** Provides locking or latching of modifier keys (Shift, Control, etc.) so that they can be used without simultaneously pressing the keys being modified. This allows single finger operation of multiple key combinations.
- **RepeatKeys:** Delays the onset of key repeat, allowing users with limited coordination time to release keys before multiple characters are sent.
- **SlowKeys:** Requires a key to be held down for a set period before keypress acceptance. This allows users with limited coordination to accidentally press keys without sending keypress events.
- **MouseKeys:** An alternative to the mouse that provides keyboard-based explicit control of cursor movement and all mouse button press/release events
- **ToggleKeys:** Indicates locking key state with a tone when pressed, (e.g., Caps Lock).
- **BounceKeys:** Requires a delay between keystrokes before accepting the next keypress so users with tremors can prevent the system from accepting inadvertent keypresses.

10 Appendix B: Accessibility APIs for Various Operating Systems

The following appendix is informative only.

The following is a list of currently public accessibility APIs that are available for various operating systems. The inclusion of this list is not an endorsement of any particular accessibility API by the W3C in general or WAI in particular. The information is provided only as a convenience to browser developers.

The WAI Working Group strongly recommends using accessibility APIs to promote compatibility with 3rd party assistive technology. Third-party assistive technology can use the accessibility information provided by the APIs to provide an alternative user interface for various disabilities.

10.1 Microsoft Active Accessibility in Windows 95/NT versions

Information on active accessibility can be found at the Microsoft WWW site on Active Accessibility.

10.2 SUNSoft Java Accessibility API in Java Code

Information on Java Accessibility API can be found at Java Accessibility Utilities.

11 Appendix C: Closed Captioning Resources

The following appendix is informative only.

11.1 The following is a list of resources for closed captioning information and service

- The Caption Center - a non-profit service of the WGBH Educational Foundation and the world's first captioning agency.
- Closed Captioning Web - links to closed captioning service providers, resources, facts, comments, and links to other captioning-related Web sites.
- National Captioning Institute
- FAQ - Closed Captioning - This FAQ, maintained by Gary Robson, contains questions about all aspects of closed captioning for television and on-line broadcasts.

12 Acknowledgments

WAI User Agent Working Group Chair:

Jon Gunderson, University of Illinois at Urbana/Champaign

W3C Team contacts:

Judy Brewer and Daniel Dardailler

In addition, we would like to thank the following people who have contributed through review and comment: James Allen, Irene Au, Kitch Barnicle, Kevin Carey, Wendy Chisholm, Chetz Colwell, Neal Ewers, Geoff Freed, Larry Goldberg, Jon Gunderson, Mark Hakkinen, Chris Hasser, Phill Jenkins, Leonard Kasday, George Kerscher, Marja-Riitta Koivunen, Josh Krieger, Greg Lowney, Scott Luebking, William Loughborough, Charles McCathieNevile, Masafumi Nakane, Charles Opperman, Mike Paciello, David Pawson, Helen Petrie, David Poehlman, Michael Pieper, Jan Richards, Greg Rosmaita, Liam Quinn, T.V. Raman, Robert Savellis, Constantine Stephanidis, Jim Thatcher, Jutta Treviranus, Steve Tyler, Gregg Vanderheiden, Jaap van Lelieveld, Jon S. von Tetzchner, Ben Weiss, Evan Wies, Chris Wilson, Henk Wittingen, and Tom Wlodkowski.

If you have contributed to the UA guidelines and your name does not appear please contact the editors to add your name to the list.

13 References

[CSS1]

"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds. The CSS1 Recommendation 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. The CSS2 Recommendation is available at:
<http://www.w3.org/TR/REC-CSS2/>.

[CSS2-WAI]

"WAI Resources: CSS2 Accessibility Improvements", I. Jacobs and J. Brewer, eds. This document, which describes accessibility features in CSS2, is available at:
<http://www.w3.org/WAI/References/CSS2-access>.

[HTML40]

"HTML 4.0 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds. The HTML 4.0 Recommendation is available at:
<http://www.w3.org/TR/REC-html40/>.

[HTML4-WAI]

"WAI Resources: HTML 4.0 Accessibility Improvements", I. Jacobs, J. Brewer, and D. Dardailler, eds. This document, which describes accessibility features in HTML 4.0, is available at:
<http://www.w3.org/WAI/References/HTML4-access>.

[SMIL]

"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", P. Hoschka, editor. The SMIL 1.0 Recommendation is available at:
<http://www.w3.org/TR/REC-smil/>

[WAI-PAGEAUTH]

"WAI Accessibility Guidelines: Page Authoring", G. Vanderheiden, W. Chisholm, and I. Jacobs, eds. These guidelines for designing accessible documents are available at:
<http://www.w3.org/TR/WD-WAI-PAGEAUTH>.
