W3C WD-WAI-USERAGENT-TECH-19980210

Techniques for User Agent Accessibility Guidelines

W3C Working Draft *10-Feb-1999*

This version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent-tech
Latest version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT/wai-useragent-tech
Latest public version:
http://www.w3.org/TR/WD-WAI-USERAGENT/wai-useragent-tech
Previous version:
http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19981112
Latest "User Agent Accessibility Guidelines":
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT
Latest public version of "User Agent Accessibility Guidelines":
http://www.w3.org/TR/WD-WAI-USERAGENT

Editors:
Jon Gunderson <jongund@uiuc.edu>
Ian Jacobs <ij@w3.org>

Abstract

This document describes techniques that user agent developers may use to satisfy the checkpoints listed in "User Agent Accessibility Guidelines." Techniques relate to the accessibility of user interfaces, content rendering, program interfaces, and accessibility features of languages such as HTML, CSS and SMIL. Like "User Agent Accessibility Guidelines", the current document emphasizes the accessibility of interoperability between two important classes of user agents - graphical desktop browsers and dependent assistive technologies (including screen readers, screen magnifiers, and voice input software). However, it is meant to be applicable to user agents in general, including text and voice browsers, and multimedia players.

While "User Agent Accessibility Guidelines" has been designed to be a stable document of principles, the current document will continue to evolve as technologies change and user agent manufacturers discover more effective techniques for designing accessible user agents.

This document is part of a series of accessibility documents published by the W3C 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.

Although the organization of this document has begun to stabilize, the Working Group has not yet filled in the details of all the techniques. This draft does not distinguish those techniques that mainstream user agents are expected to implement natively from those that may be satisfied by providing information through an interface.

This document has been produced as part of the W3C WAI Activity and intends to improve user agent accessibility for all users. 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/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent-tech.html

A plain text file:
: http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent-tech.txt,

HTML as a gzip'ed tar file:
: http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent.tgz,

HTML as a zip file (this is a '.zip' file not an '.exe'):
: http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent.zip,

A PostScript file:
: http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent-tech.ps,

A PDF file:
: http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent-tech.pdf.

In case of a discrepancy between the various formats of the specification, http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent-tech.html 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 Guidelines, Checkpoints, and Techniques

The guidelines documents have been organized to address readers seeking abstract principles of accessible user agent design and readers seeking concrete solutions. The guidelines documents define three terms for different levels of abstraction:

Guideline
: A guideline is a general principle of accessible design. A guideline addresses the question "What accessibility issues should I be aware of?"

Checkpoint
: A checkpoint is a specific way of satisfying one or more guidelines. While checkpoints describe verifiable actions that may be carried out by the user agent developer, implementation details are described elsewhere. A checkpoint answers the question "What should I do to improve accessibility?"

Technique
: A technique is an implementation of one or more checkpoints in a given language (e.g., HTML, XML, CSS, DOM, ...). A technique answers the question "How do I do that in HTML or SMIL or CSS...?"

The current document contains techniques, according to different interfaces between the user agent and the operating system, the document, the user, and other software. Note that each technique in this document is numbered according to its number in the guidelines document.

2 The User Agent and the Operating System

4.1.4 Follow operating system conventions for user interface design, user agent configuration (including configuration profiles), product installation and documentation, and accessibility flags and interfaces. [Priority 2]

The operating system application programming interfaces (APIs) that support accessibility 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.

The WAI Working Group strongly recommends using and supporting APIs that improve accessibility and 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.

The following is an informative list of currently public APIs that support accessibility:

- Microsoft Active Accessibility in Windows 95/NT versions. Information on active accessibility can be found at the Microsoft WWW site on Active Accessibility.
- Sun Microsystems Java Accessibility API in Java Code. Information on Java Accessibility API can be found at Java Accessibility Utilities.

Many major operating systems have built-in accessibility features for improving the usability of the standard operating system by persons with disabilities. When designing software that runs above an underlying operating system, developers should ensure that the application:

1. Makes use of operating system level features. See the appendix of accessibility features [p. 15] for some comment operating systems.
2. Inherits operating system settings related to accessibility. Pertinent settings include font and color information as well as other pieces of information discussed in this document.

[Ed. We Should put a more exhaustive list here.]

4.1.1 Ensure that the software may be installed in a device-independent manner for all supported input and output devices. [Priority 1]

Ensure that software may be installed, when possible, according to the conventions of the operating system.

3 The User Agent and the Document

3.1 Identification

In order for the user agent to satisfy many of the techniques in this document (related to rendering, notification, navigation, etc.), it must be able to identify certain types of information in addition to recognizing object types such as images, video, sound, and elements of the document language.

Sources of alternative representations of information

User agents must be able to recognize sources of alternative representations of content.

In HTML

- For the IMG element: The "alt", "title", and "longdesc" attributes
- For the OBJECT element: The content of the element and the "title" attribute.
- For the APPLET element: The "alt" attribute and the content of the element.
- For the AREA element: The "alt" attribute.
- For the INPUT element: The "alt" attribute.
- For the ACRONYM and ABBR elements: The "title" attribute may be used for the acronym or abbreviation expansion.
- For the TABLE element, the "summary" attribute
- For frames, the NOFRAMES element and the "longdesc" attribute on FRAME and IFRAME.
- For scripts, the NOSCRIPT element.

In SMIL

The "alt", "title", "longdesc", and "abstract" attributes. *To be completed.*.

Sources of blinking text and animation

User agents that recognize the following sources of blinking text and animations must enable users to freeze that text.

In HTML

- The BLINK element. **Note.** The BLINK element is not defined by a W3C specification.
- The MARQUEE element. **Note.** The MARQUEE element is not defined by a W3C specification.

In CSS

- The 'blink' value of the 'text-decoration' property.

In GIF animated images.

To be completed.

Sources of event handlers

Certain elements of the document language may have associated event handlers that are triggered when certain events occur. User agents must be able to identify those elements with event handlers statically associated (i.e., associated in the document source, not in a script).

In HTML

All of the attributes beginning with the prefix "on": "onblur", "onchange", "onclick", "ondblclick", "onkeydown", "onkeypress", "onkeyup", "onload", "onmousedown", "onmousemove", "onmouseout", "onmouseover", "onmouseup", "onreset", "onselect", "onsubmit", and "onunload".

[Ed. Other sources?]

[Ed. To be completed.]

Information about the document

User agents must be able to compute the following information about a document:

- Has the document been fully loaded?
- What is the size in bytes of the document?
- How many active components are in the document?
- Where is the point of regard?

3.2 Rendering alternative representations of content

Alternative representations of information for images, video, applets

See the section on sources of alternative information [p. 5]

5.2.1 Ensure that the user has access to alternative representations of content (e.g., the value of "alt" in HTML or SMIL, the resource designated by "longdesc", or the content of OBJECT in HTML 4.0, the "summary" attribute for tables in HTML, etc.). [Priority 1]

5.2.2 When no alternative text representation is available, indicate what type of object is present. [Priority 2]

5.2.3 When null alternative text has been defined, suppress the rendering of the alternative representation. [Priority 3]

Textual equivalents for audio and video

5.2.5 Allow the user specify that captions for audio be rendered at the same time as the audio. [Priority 2]

5.2.6 Allow the user specify that captions for video be rendered at the same time as the video. [Priority 1]

5.1.10 Allow the user to control the position of captions. [Priority 1]

Audio equivalents for video

5.2.7 Allow the user specify that audio descriptions of video be rendered at the same time as the video. [Priority 2]

Alternative representations for scripts

In HTML, content of NOSCRIPT.

Alternative representations for frames

[Ed. See Scott's suggestions]

1. In HTML, content of NOFRAMES.
2. Otherwise, list of frames. Use "title" as name of frame, otherwise "name".

Alternative representations for tables

In HTML, "summary" attribute. Also, the "abbr" attribute for headers (see the section on table cells and headers).

Generated content

Generated content specified through style sheets can help orient the user. For instance, if, through style sheets, the user can insert the word "Link" before every link, this text may be used by a speech synthesizer or Braille devices. See the section on the support for CSS2 accessibility features [p. 14] below.

3.3 Styles and formatting

[Ed. Based on comments from Jason, add information about UAs allowing users to suppress the rendering of certain elements (e.g., for audio browsers). This can be done with the CSS1 property 'display: none'.]

Tables

5.4.1 Provide access to the contents of a specific cell. [Priority 1]
5.4.2 Provide access to header information for a specific table cell. [Priority 1]
Possible solutions:

- Provide info through an API.
- Ignore table markup (screen reader)
- Render cells as blocks (screen reader)
- Allow navigation and querying of cell/header information.
- Allow user to read one table or row at a time (to help find headers).

[Ed. Define algorithm for finding "header information" here. Does it come from THEAD, TH, attributes, HTML 4.0 algorithm for finding header information, etc. Allow the user to choose either "abbr" or what is calculated by the header algorithm.]
[Ed. Discuss repair strategies for finding header information?]

Frames

Possible solutions:

- Provide info through an API.
- Frames as blocks.
- Frameset as list of links to individual frames (based on frame name).
- NOFRAMES content (when?)

Links

5.3.6 Provide a mechanism (e.g., through style sheets) to distinguish visited links from unvisited links. [Priority 3]

5.3.7 Allow the user to specify (e.g., through style sheets) that images used in links must have borders. [Priority 3]

[Ed. Talk about CSS pseudo-classes for :hover]

[Ed. Talk about using :before to clearly indicate that something is a link (e.g., 'A:before { content : "LINK:" }')]

Forms

Audio and video

[Ed. Talk about EMBED? Charles recommends that it be put inside an OBJECT if used.]

Highlighting

6.1.1 Provide a mechanism for highlighting and identifying (through a standard interface where available) the current view. [Priority 1]

6.1.2 Provide a mechanism for highlighting and identifying (through a standard interface where available) the user selection. [Priority 1]

6.1.3 Provide a mechanism for highlighting and identifying (through a standard interface where available) the current focus. [Priority 1]

Document

6.2.1 Allow the user to view a document outline constructed from its structural elements (e.g., from header and list elements). [Priority 2]

4 Functionality offered by the User Agent to allow the User to work with the Document

4.1 Access to all functionality

In general, this requirement may be met by providing keyboard access to all functionality and by exposing that functionality to third party technology.

4.2 Selection and focus

6.1.4 Ensure that when the selection changes, it is in the viewport after the change. [Priority 2]

6.1.5 Ensure that when the focus changes, it is in the viewport after the change. [Priority 2]

6.1.7 Keep track of the user's point of regard in each view and restore it when the user returns to the view. [Priority 1]

4.3 Navigation

There are different ways a user may want to navigate while browsing the Web, including:

- By changing the position of the viewport (e.g., by scrolling down the page).
- By shifting focus and/or selection from one active component to the next of the same type (e.g., from link to link).
- By navigating among documents.

[Ed. Describe tabbing navigation here.]

Document navigation

6.1.6 Allow the user to navigate views (notably those with frame viewports). Navigating into a view makes it the current view. [Priority 1]

6.2.2 Allow the user to navigate the document tree. [Priority 2]

Document object navigation

[Ed. Discuss tabbing navigation here. How tabbing navigation is specified in HTML: "tabindex".]

5.3.1 Allow the user to navigate sequentially among links. [Priority 1]

5.5.2 Allow the user to navigate among forms in a document. [Priority 1]

5.5.1 Allow the user to navigate among form controls within a form. [Priority 1]

6.3.1 Allow the user to navigate among elements with associated event handlers. [Priority 1]

6.2.3 Allow the user to navigate sequentially among headers. [Priority 2]

6.2.4 Allow the user to navigate sequentially among block elements (e.g., paragraphs, lists and list items, etc.) [Priority 2]

5.3.8 Allow the user to navigate among elements with associated long descriptions. [Priority 1]

5.4.3 Allow the user to navigate among tables in a document. [Priority 1]

5.4.4 Allow the user to navigate among table cells of a table (notably left/right within a row and up/down within a column). [Priority 1]

4.4 Searching

5.3.2 Allow the user to search for a link in the current document based on its link text. [Priority 1]

5.3.3 Allow the user to search for a link based on its attribute values. [Priority 1]

5.3.4 Allow the user to move the focus to a link based on its integer (tabbing-order) position. [Priority 1]

5.5.3 Allow the user to search for a form control based on its text content. [Priority 1]
5.5.4 Allow the user to search for a form control based on its attribute values. [Priority 1]
6.2.5 Allow the user to search for an element in the current document based on its text content. In case of a match, the selection should be moved to the element. [Priority 2]
5.3.9 Allow the user to search linked long description text (i.e., in another document). In case of a match, the focus should be moved to the link in the current document. [Priority 2]
5.4.5 Allow the user to search for a table cell based on its contents or header information. [Priority 2]

4.5 Querying

5.3.5 Provide the user with information about the number of links in a document. [Priority 2]
5.5.5 Provide the user with information about the number of forms in a document. [Priority 2]
5.5.6 Provide the user with information about the number of controls in a form. [Priority 2]
5.4.8 Allow the user to find out the number of tables in a document. [Priority 2]
5.4.6 Allow the user to select a table cell and find out its row/column coordinates in the table. [Priority 2]
5.4.7 Allow the user to find out the dimensions of a table. [Priority 2]
6.1.9 When a document is loaded or when requested by the user, make available document summary information. [Priority 2]
[Ed. What does summary information include?]
6.1.8 Provide the user with information about the number of viewports. [Priority 2]
6.1.10 Provide the user with information about how much of the document has been viewed (i.e., where the point of regard is with respect to the beginning of the document). [Priority 2]
4.3.2 Ensure that user can find out about all keyboard bindings. [Priority 2]

4.6 Activation

4.2.3 Ensure that the user can activate the links in a document in an input device-independent manner. [Priority 1]
4.2.4 Ensure that the user can activate the form controls in a document in an input device-independent manner. [Priority 1]
4.2.5 Ensure that the user can activate the event handlers in a document in an input device-independent manner. [Priority 1]

5 The User Agent and the User

5.1 Accessible user interface

4.2.1 Ensure that all functionalities offered by the user agent interface are available through all supported input devices. These include the installation procedure, access to product documentation and help, and software configuration. [Priority 1]

4.2.2 Ensure that all messages to the user (e.g., warnings, errors, etc.) are available in an output device-independent manner for all supported interactive output devices. [Priority 1]

5.2 Visibility of accessibility features

Universal design means that access to features that help accessibility should be integrated into normal menus. User agents should avoid regrouping access to accessibility features into specialized menus. However, documentation must include a section or chapter on accessibility where all available functionalities and mechanisms are described.

4.1.3 Describe product features known to promote accessibility in a section of the product documentation. [Priority 2]

Documentation includes installation documentation. Users must have access to installation information, either in electronic form (diskette, over the Web), by fax, or by telephone.

4.1.2 Ensure that product documentation is available in at least one accessible, open standard electronic format (e.g., HTML, XML, ASCII). [Priority 2]

Vendors should provide documentation in a format that may be tailored to a user's particular needs. Helpful formats include:

- ASCII
- A format defined by a W3C specification such as HTML, XML.

Vendors should provide several alternatives to hardcopy documentation when possible: electronic, large print, braille, audio cassette, etc.

4.3.3 Provide a mechanism to render the presence of a keyboard binding for an active element. [Priority 2]

5.3 Configuration

User control of style

To ensure accessibility, users must have **final control** over certain renderings.

For text and color:

5.1.1 Allow the user to override font family. [Priority 1]

5.1.2 Allow the user to override font size. [Priority 1]

5.1.3 Allow the user to override foreground color. [Priority 1]

5.1.4 Allow the user to override background color. [Priority 1]

5.1.5 Allow the user to override selection highlighting (e.g., foreground and background color). [Priority 1]

5.1.6 Allow the user to override focus highlighting (e.g., foreground and background color). [Priority 1]

[Ed. These may be rendered in a variety of ways. How do we specify rendering?]

For images, applets, and animations:

4.4.2 Allow the user to turn on and off rendering of background images. [Priority 1]

5.1.7 Allow the user to override animation rate. [Priority 2]

For user agents rendering video:

5.1.8 Allow the user to override video frame rates. [Priority 1]

5.1.9 Allow the user to start, stop, pause, and rewind video. [Priority 2]

For user agents rendering audio:
5.1.11 Allow the user to override audio playback rate. [Priority 1]
5.1.12 Allow the user to start, stop, pause, and rewind audio. [Priority 2]
5.1.13 Allow the user to override audio volume. [Priority 2]
5.2.4 Allow the user to choose from among available audio tracks. [Priority 1]
For user agents rendering speech:
5.1.15 Allow the user to override speech volume. [Priority 2]
5.1.14 Allow the user to override speech playback rate. [Priority 1]
User interface:
5.1.16 When new windows or user interface components are spawned, allow the user to control window size and position. [Priority 2]

User profiles

A configuration profile allows users to save their user agent settings and re-apply them easily, which is particularly valuable in an environment where several people may use the same machine.

4.1.5 Allow the user to configure the user agent in named profiles that may be shared (by other users or software). [Priority 2]

The user should be able to easily transfer their profiles between installations of the same user agent. One way to facilitate this is to follow operating system conventions for profiles where applicable.

Users should be able to switch rapidly between profiles (or the default settings). This is helpful when:

- Several people use the same machine.
- A user with a disability is being helped by able-bodied user who may not recognize the information being displayed using the user's profile.

User agents may apply a profile when the user logs in. They may also allow users to apply settings interactively, for example by allowing them to choose from a list of named profiles in a menu.

Sample profiles (based on common usage scenarios) can assist users in the initial set up of the user agent. These profiles can serve as models and may be copied and fine-tuned to mean an individual's particular needs.

Feature control

[Ed. Add note here that while useful to turn off support, say for all images, it's also useful to be able to view one particular image.]
4.4.1 Allow the user to turn on and off rendering of images. [Priority 1]
4.4.3 Allow the user to turn on and off rendering of video. [Priority 1]
4.4.4 Allow the user to turn on and off rendering of sound. [Priority 1]
4.4.5 Allow the user to turn on and off rendering of captions. [Priority 1]
4.4.6 Allow the user to turn on and off animated or blinking text. [Priority 1]
4.4.7 Allow the user to turn on and off animations and blinking images. [Priority 1]
4.4.8 Allow the user to turn on and off support for scripts and applets. [Priority 1]
4.4.10 Allow the user to turn on and off support for author style sheets. [Priority 1]
4.4.9 Allow the user to turn on and off support for user style sheets. [Priority 1]

4.4.11 Allow the user to turn on and off rendering of frames. [Priority 2]
4.4.12 Allow the user to turn on and off support for spawned windows. [Priority 1]
User agents may:

- Allow users to turn of support for spawned viewports entirely
- Prompt them before spawning a viewport

For example, user agents may recognize the HTML construct `target="_blank"` and spawn the window according to the user's preference.
4.4.13 Allow the user to turn on and off automatic page forwarding. [Priority 3]
Although no HTML specification defines this behavior formally, some user agents support the use of the META element to refresh the current page after a specified number of seconds, with the option of replacing it by a different URI.

Keyboard configuration

4.3.1 Allow the user to configure keyboard access to user agent functionalities. Configuration includes the ability to specify single- as well as multi-key access. [Priority 2]
4.3.4 Display keyboard bindings in menus. [Priority 3]
[Ed. Discuss keyboard access here. How access is specified in HTML: "accesskey".]
[Ed. New section. Users must be allowed to control keyboard configuration based on specific needs. For poor motor control, keys far apart. For poor mobility, keys close together. General principle: fewest keystrokes, short distance to move.]

5.4 Notification of events and changes

6.1.11 Provide the user with information about document loading status (e.g., whether loading has stalled, whether enough of the page has loaded to begin navigating, whether following a link involves a fee, etc.) [Priority 3]
6.3.2 Alert the user when scripts or applets are executed. [Priority 2]
6.3.3 Provide information about document changes resulting from the execution of a script. [Priority 3]
5.5.7 Allow the user to request to be prompted before a form is submitted. [Priority 3]

6 The User Agent and Assistive Technologies

7.2.1 Implement DOM Level 1 and expose it to dependent user agents through a public interface. [Priority 1]
Use DOM Level 1 [DOM1] [p. 19] for access to HTML and XML document information. However, as the DOM specification indicates:

> The DOM Level 1 does not include mechanisms to access and modify style specified through CSS 1. Furthermore, it does not define an event model for HTML documents. This functionality is planned to be specified in a future Level of this specification.

It is important for user agents to provide access to style and scripting information in a document, so the following techniques should be used to achieve this.

7.2.2 Support other accessibility application programming interfaces (APIs) (e.g., Microsoft Active Accessibility, Sun Microsystems Java Accessibility) [Priority 2]

7.2.3 Use standard interfaces defined for the operating system. [Priority 2]

Use standard rather than custom controls when designing user agents. 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.

7 Support for accessibility features of certain languages

7.1 HTML

[Ed. Integrate Charles' notes on accessibility features of HTML. Sort by priority. See his list of key elements.]

7.1.1 Support accessibility features defined by W3C Recommendations for HTML. [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. [Priority 1]
- Support the CAPTION element ([HTML40] [p. 19] , section 11.2.2) for rich table captions. [Priority 2]
- Support the ACRONYM and ABBR elements ([HTML40] [p. 19] , section 9.2.1) for acronyms and abbreviations. [Priority 2]
- Support the "summary" attribute for TABLE ([HTML40] [p. 19] , section 11.2.1) for table summary information. [Priority 2]
- Support the NOSCRIPT element ([HTML40] [p. 19] , sections 18.3.1 and 16.4.1) for accessible alternatives to scripts. [Priority 2]
- Support the NOFRAMES element ([HTML40] [p. 19] , sections 18.3.1 and 16.4.1) for accessible alternatives to frames. [Priority 2]
- Support the "lang" attribute ([HTML40] [p. 19] , section 8.1). [Priority 2]
- Support the "tabindex" attribute ([HTML40] [p. 19] , section 17.11.1) for assigning the order of keyboard navigation within a document. [Priority 3]
- Support the "accesskey" attribute ([HTML40] [p. 19] , section 17.11.2) for assigning keyboard commands to active components such as links, and form controls. [Priority 3]

7.2 CSS

7.1.2 Support accessibility features defined by W3C Recommendations for CSS. [Priority 1]

The following techniques 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.

- Completely implement Cascading Style Sheets, level 1 [Priority 1]
- Allow the user to turn off author styles represented by author style sheets. [Priority 1]
- Allow the user to adjust default values represented by user agent style sheets. [Priority 1]
- Support the :before and :after pseudo-elements as defined in CSS2 ([CSS2] [p. 19] , section 12.1). [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. [Priority 1]
- Allow the user to specify user styles through style sheets (see [CSS2] [p. 19] , section 6.4). [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 and user agent defaults [Priority 2]
- Support aural cascading style sheets (see [CSS2] [p. 19] , chapter 19) for the auditory presentation of documents. [Ed. Expand this into individual properties. Also add 'speak-header'.][Priority 2]

7.3 SMIL

7.1.3 Support accessibility features defined by W3C Recommendations for SMIL. [Priority 1]

[Ed. Ensure that in SMIL, users can stop temporal links. See proposal from Marja.]

8 Appendix: Accessibility features of some operating systems

Several of the more popular mainstream operating systems now include a common suite of built-in accessibility features that are designed to assist individuals with varying abilities. Despite operating systems differences, the built-in accessibility features use a similar naming convention and offer similar functionalities, within the limits imposed by each operating system (or particular hardware platform).

The following is a list of built-in accessibility features from several platforms:

StickyKeys
: These allow users to perform a multiple simultaneous key sequence by pressing and releasing each key in sequential order. StickyKeys is designed to work with only those keys defined as modifier keys. Modifier keys are pressed in combination with other keys, to change the outcome of the second (or more) pressed keys. For example, the SHIFT key is defined as a modifier key, since it is commonly used to create upper case characters. Each operating system or hardware platform typically defines a set of keys which can act as modifier keys. The most common modifier keys include SHIFT, CONTROL, and ALTERNATE.

MouseKeys
: These allow users to move the mouse cursor and activate the mouse button(s) from the keyboard.

RepeatKeys
: These allow users to set how fast a key repeats (e.g., sometimes referred to as typematic rate) when the key is held pressed (e.g., Repeat Rate), and also allows

control over how quickly the key starts to repeat after the key has been pressed (e.g., delay Until Repeat). Key repeating may also be eliminated.

SlowKeys
: These instruct the computer not to accept a key as pressed until it has been pressed and held down for a specific user adjustable length of time.

BounceKeys
: These prevent extra characters from being typed if the user bounces (e.g., tremor) on the same key when pressing or releasing it.

ToggleKeys
: These provide an audible indication for the status of keys that have a toggled state (e.g., keys that maintain status after being released). The most common toggling keys include Caps Lock, Num Lock, and Scroll Lock.

SoundSentry
: These monitor the operating system and applications for sounds, and attempt to provide a visual indication when a sound is being played. Older versions of Sound Sentry may have flashed the entire display screen for example, while newer versions of SoundSentry provide the user with a selection of options, such as flashing the active window or flashing the active window caption bar.

The next three built in accessibility features are not as commonly available as the above group of features, but are included here for definition, completeness, and future compatibility.

ShowSounds
: These are user setting or software switches that are available for the operating system and application (including user agents) APIs to read, to notify them that the user wishes audio information to also be presented in a visual format.

High Contrast
: These automatically change the display fonts and colors to choices which should provide for easier reading.

TimeOut
: These allow the built-in accessibility features to automatically turn off if the computer is unused for a specified length of time, and is intended for use when the computer is in a public setting (e.g., library). TimeOut might also be referred to as reset or automatic reset.

The next accessibility feature listed here is not considered to be a built in accessibility feature (since it only provides an alternate input channel) and is presented here only for definition, completeness, and future compatibility.

SerialKeys
: These allow a user to perform all keyboard and mouse functions from an external assistive device (such as communication aid) communicating with the computer via a serial character stream (e.g., serial port, IR port, etc.) rather than or in conjunction with, the keyboard, mouse, and other standard input devices/methods.

8.1 Microsoft Windows 95, Windows 98, and Window NT 4.0

The following accessibility features can be adjusted from the Accessibility Options Control Panel:

- StickyKeys: modifier keys include SHIFT, CONTROL, and ALTERNATE
- FilterKeys: grouping term for SlowKeys, RepeatKeys, and BounceKeys
- MouseKeys:
- ToggleKeys:
- SoundSentry:
- ShowSounds:
- Automatic reset: term used for TimeOut
- High Contrast:
- SerialKeys:

Additional accessibility features available in Windows 98:

Magnifier
: This is a windowed, screen enlargement and enhancement program used by persons with low vision to magnify an area of the visual display (e.g., by tracking the text cursor, focus, etc.). Magnifier can also invert the colors used by the system within the magnification window.

Accessibility Wizard
: This is a setup tool intended to assist a person with making choices which setting up the accessibility features on a workstation.

8.2 Apple Macintosh Operating System

The following accessibility features can be adjusted from the Easy Access Control panel (Note: Apple convention uses a space within the accessibility feature names.)

- Sticky Keys: modifier keys include the SHIFT, OPEN APPLE (COMMAND), OPTION (ALT) and CONTROL keys.
- Slow Keys:
- Mouse Keys:

The following accessibility features can be adjusted from the Keyboard Control Panel.

- Key Repeat Rate (e.g., part of RepeatKeys)
- Delay Unit Repeat (e.g., part of RepeatKeys)

The following accessibility feature can be adjusted from the Sound or Monitors and Sound Control Panel (depends upon which version of the OS).

- Adjusting the volume to off or mute causes the Macintosh to flash the title bar whenever the operating system detects a sound (e.g., SoundSentry)

Additional accessibility features available for the Macintosh OS:

CloseView
: This is a full screen, screen enlargement and enhancement program used by persons with low vision to magnify the information on the visual display, and it can also change the colors used by the system.

SerialKeys
: This is available as freeware from Apple and several other Web sites.

8.3 AccessX, X Keyboard Extension (XKB), and the X Window System

(Note: AccessX became a supported part of the X Window System X Server with the release of the X Keyboard Extension in version X11R6.1)

The following accessibility features can be adjusted from the AccessX graphical user interface X client on some DEC, SUN, and SGI operating systems. Other systems supporting XKB may require the user to manipulate the features via a command line parameter(s).

- StickyKeys: modifier keys are platform dependent, but usually include the SHIFT, CONTROL, and META keys.
- RepeatKeys:
- SlowKeys:
- BounceKeys:
- MouseKeys:
- ToggleKeys:

8.4 DOS (Disk Operating System)

The following accessibility features are available from a freeware program called AccessDOS, which is available from several Internet Web sites including IBM, Microsoft, and the Trace Center, for either PC-DOS or MS-DOS versions 3.3 or higher.

- StickyKeys: modifier keys include the SHIFT, CONTROL, and ALTERNATE keys.
- Keyboard Response Group: grouping term for SlowKeys, RepeatKeys, and BounceKeys
- MouseKeys:
- ToggleKeys:
- SoundSentry (incorrectly name ShowSounds):
- SerialKeys:
- TimeOut:

9 Acknowledgments

Many thanks to the following people who have contributed through review and comment: Paul Adelson, James Alan, Denis Anson, Kitch Barnicle, Harvey Bingham, Judy Brewer, Kevin Carey, Wendy Chisholm, David Clark, Chetz Colwell, Wilson Craig, Daniel Dardailler, Neal Ewers, Geoff Freed, Larry Goldberg, Markku Hakkinen, Earle Harrison, Chris Hasser, Kathy Hewitt, Phill Jenkins, Leonard Kasday, George Kerscher, Marja-Riitta Koivunen, Josh Krieger, Catherine Laws, Greg Lowney, Scott Luebking, William Loughborough, Napoleon Maou, Charles McCathieNevile, Masafumi Nakane, Charles Oppermann, Mike Paciello, David Pawson, Michael Pederson, Helen Petrie, David Poehlman, Michael Pieper, Jan Richards, Hans Riesebos, Joe Roeder, Greg Rosmaita, Liam Quinn, T.V. Raman, Robert Savellis, Constantine Stephanidis, Jim Thatcher, Jutta Treviranus, Claus Thøgersen, 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.

10 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.

[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. The DOM Level 1 Recommendation 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. The HTML 4.0 Recommendation is available at:
http://www.w3.org/TR/REC-html 40/.

[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.

[RFC2119]
"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt

[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-AUTOOLS]
"Authoring Tool Accessibility Guidelines", J. Treviranus, J. Richards, I. Jacobs, and C. McCathieNevile, eds. These guidelines for designing accessible authoring tools are available at:
http://www.w3.org/TR/WD-WAI-AUTOOLS.

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