W3C WD-WAI-USERAGENT-TECH-19981019

Techniques for "WAI Guidelines: User Agent"

W3C Working Draft *19-Oct-1998*

This version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent-tech
Latest version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-TECH
Current version of "WAI Guidelines: User Agent":
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019
Latest "WAI Guidelines: User Agent":
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT
Previous version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19980814
Editors:
Jon Gunderson <jongund@uiuc.edu>
Ian Jacobs <ij@w3.org>

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

Abstract

This document is a list of techniques that implement the guidelines described in "WAI Guidelines: User Agent."

While "WAI Guidelines: User Agent" strives to be a stable document (as a W3C Recommendation), the current document will undoubtedly evolve as technologies change and browser manufacturers discover more effective techniques for designing accessible user agents.

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/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent-tech
A plain text file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent-tech.txt,
HTML as a gzip'ed tar file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent-tech.tgz,
HTML as a zip file (this is a '.zip' file not an '.exe'):
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent-tech.zip,
A PostScript file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent-tech.ps,
A PDF file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/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-19981019/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 Rating and Classification 4
2 User control of style information 4
2.1 User override of author and browser styles 4
2.2 User style profiles 4
2.3 Blinking text 5
3 Alternative content 5
3.1 General information about alternative content 5
3.2 Alternative content for images 6
3.3 Alternative content for multimedia objects 6
3.4 Alternative content for scripts 7
3.5 Alternative content for frames 7
3.6 Alternative content for tables 7
4 Provide alternative renderings of document information 8
5 Make sure the user can maintain their orientation to information in a document 10
5.1 Provide the user with information on the content and structure of a document 10
5.2 Provide the user with relative position information with in a document 10
5.3 Maintain document view and focus 10
6 Make sure the user can efficiently navigate the structure of a document . 11
6.1 Sequential keyboard navigation to structures within and between views 11
6.2 Direct keyboard navigation to structures within and between views . 11
6.3 Hierarchical keyboard navigation to structures within a view . . 12
6.4 Keyboard navigation of tabular (table) structures within a view . . 12
6.5 Allow keyboard navigation and activation of elements with associated scripts. 12
7 Make sure information on accessibility is highly visible and accessible . . 13
7.1 Ensure that user agent accessibility features are configurable . . 13
7.2 Provide obvious information about keyboard access 13
7.3 Provide accessible documentation 13
8 Make sure the User Agent is compatible with existing accessibility recommendations and APIs 13
8.1 Compatibility with HTML 4.0 14
8.2 Compatibility with Third-party Assistive Technology 15
9 Acknowledgments 17
10 References 17

1 Rating and Classification

Each guideline is classified according to the following rating system:

[Priority 1]
: This guideline **must** be implemented by user agents as a built-in feature or through compatibility with assistive technology, otherwise one or more groups of users with disabilities will find it impossible to access information.. Implementing this guideline is a basic requirement for some groups to be able to use the Web.

[Priority 2]
: This guideline **should** be implemented by user agents as a built-in feature or through compatibility with assistive technology, otherwise one or more groups of users will find it difficult to access information.. Implementing this guideline will significantly improve access to the Web.

[Priority 3]
: This guideline **may** be implemented by user agents as a built-in feature or through compatibility with assistive technology, to make it easier for one or more groups of users to access information. Implementing this guideline will improve access to the Web.

In this document, the guidelines are grouped by topic. Generally, a guideline may be "implemented" by one or more techniques. The priority levels for techniques mirror those for guidelines in weight. The priority level of a technique indicates how important it is as a means of satisfying the associated guideline. For example, a Priority 3 technique **must** be implemented to provide accessibility, while a Priority 3 technique, **may** be implemented to provide further assistance.

2 User control of style information

2.1 User override of author and browser styles

Technique: Allow the user to override author styles and browser defaults, and in particular those properties affecting: font face, font size, foreground and background colors, background images, user selection highlight colors, and user interaction (e.g., hover styles). [Priority 1] [Go to user-preference]

2.2 User style profiles

Technique: Allow the user to specify custom settings in profiles that may be shared (by other users or software). Preferably, this should be done via style sheets. 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. [Priority 3] [Go to user-portable]

The user should be able to easily transfer their profile between installations of the same user agent. For example the user could take their profile file for a particular user agent on a floppy disk or network location and use it to configure the user agent at work, home, the local public library or any other installation of that user agent.

Technique: Allow the user, through a keyboard command, to switch between browser default values and the user profile. [Priority 3] [Go to user-switch]

Provide a keyboard shortcut command to allow users to switch between browser defaults and a user profile. This is important for people when they are receiving help from an able-bodied user who may not recognize the information being displayed using the users profile. The default settings usually render the document in the way the author intended.

2.3 Blinking text

Technique: Allow the user to turn off blinking text or images for all cases when the user agent knows that text or images are blinking. [Priority 1] [Go to user-blink]

For those sources of blinking text that may be detected by a user agent, the user agent must allow the users to turn off blinking. These include:

1. The BLINK element: This is a proprietary non-standard tag recognized by some user agents.
2. The CSS 'text-decoration' property

Blinking may occur from other sources (e.g., scripts) more difficult or impossible to detect by the user agent.

3 Alternative content

Authors may specify several types of alternative content: alternative text (or "Alt-text"), brief descriptions of content, and long descriptions of content.

Technique: Give the user access to alternative content (e.g., the "alt" or "longdesc" attributes in HTML or SMIL, the content of OBJECT in HTML 4.0). [Priority 1] [Go to alt-content]

Alternative content may come from element content (e.g., the OBJECT element in HTML 4.0), attribute values (e.g., the "alt" or "title" attributes in HTML), or resources designated by attributes (e.g., the "longdesc" attribute in HTML 4.0). Until the "longdesc" attribute is widely supported, authors may designate long descriptions with *D-Links* as well.

For more information about how and when to provide alternative content, please consult the "Techniques for WAI Guidelines: Page Authoring" ([WAI-GL-TECHNIQUES] [p. 18]).

3.1 General information about alternative content

Rendered text should be wrapped so the user doesn't need to scroll horizontally to read the description.

If the author specifies an empty string as the value of "alt" (`alt=""`), the user agent should suspend the rendering of any alt text. Authors may specify null string alt text when, for example, several images are used to create a single visual effect. The user need only specify alt text on the first image to convey the intended effect.

Allow users to hide the display of D-links used to provide access to long description information. [Priority 3] [Go to alt-d-link]

Users should be able to turn on/off a switch that causes the user agent to detect [Ed. How do they do that?] D-links and hide them (e.g., by using the CSS 'display' property).

3.2 Alternative content for images

Allow the user to specify that alternative text and/or long descriptions be rendered in place of primary content. [Priority 1] [Go to alt-rendering]

In the case of the IMG element (see [HTML40]), 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], 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.

When no alternative text representation is available, indicate what type of object is present. [Priority 1] [Go to alt-no-alt]

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

3.3 Alternative content for multimedia objects

In general, users must be able to control multimedia features both statically (through preferences) and dynamically (while the presentation is playing).

Technique: Allow the user to identify and turn on/off text captions of audio. [Priority 1] [Go to multimedia-caption]

Some users require captions anytime they are available, in which case users should be able to set a user preference to view captions. Other users require captions only in certain circumstances, in which case they need a mechanism to identify when captions are available, and to turn them on while a presentation is playing. A mechanism for identification of captions should also function non-visually as some users can neither hear audio files nor see captions, but can still access captions through screen-reading software and refreshable Braille display.

Technique: Allow the user to control the size, color, and background color of captions. [Priority 1] [Go to multimedia-rendering] .

Some users require specific font size, color, and contrast with caption background to be able to view captions.

Technique: Allow the user to identify and turn on/off audio descriptions of video. [Priority 1] [Go to multimedia-descriptions]

Users who cannot see a video media object need a non-visual way to identify that an audio description is available. - Technique: Provide a standard mechanism for notifying third party assistive technologies (e.g. screen-reading software) of the existence of an audio description for a video object. Provide a mechanism (which can function non-visually) for turning on or off the audio description.

Technique: Ensure that text media objects may be identified by third-party assistive technologies. [Priority 1] [Go to multimedia-compatibility]

Users who cannot see need a non-visual way to identify that a text media object is available to their third party assistive technology, particularly screen readers. - Technique: Text media objects should use a standard mechanism for notifying third party assistive technologies (e.g. screen-reading software] of their availability.

Technique: Make accessibility-related information from the OS user profile available to the media player. [Priority 1] [Go to multimedia-os-options]

Information from user preferences set in the operating system related to accessibility should be available to multimedia players. Players should use this information for determining how to render multimedia materials.

Technique: Allow users to reposition captions. [Priority 2] [Go to multimedia-positioning]

Some multimedia presentations will include positioning conflicts between captions which can obscure key visual elements of video media objects. - Technique: Provide mechanisms to control caption display location dynamically and through user preferences.

Technique: Allow users to control (dynamically) the rendering rate of audio media objects. [Priority 2] [Go to multimedia-speed]

Users with aural-processing learning disabilities may require a slower pace of audio; users who are experienced users of synthesized speech may be able to tolerate a far faster pace of audio. - Technique: Provide mechanism for dynamic control of presentation pace.

3.4 Alternative content for scripts

Technique: Allow the user to specify that alternatives to a script be rendered (e.g., in HTML, the content of NOSCRIPT). [Priority 1] [Go to alt-no-script]

The user should have the option of the rendering of NOSCRIPT information instead of executing the script.

3.5 Alternative content for frames

Technique: Allow the user to specify that alternatives to a frame be rendered (e.g., in HTML, the content of NOFRAMES). [Priority 1] [Go to alt-no-frame]

3.6 Alternative content for tables

Technique: Allow the user to specify that alternatives to a table be rendered (e.g., the value of the "summary" attribute on TABLE in HTML 4.0). [Priority 1] [Go to alt-table-summary]

The user agent can select the rendering of the summary information after the caption information, which can be used to describe the contents of the table.

Example(s):

Example.

```
<TABLE summary="AAAAA AAAA  AAAAA AA AA
               AAA AA AAAA AAA  AAAA ">
  <CAPTION>BB BBBBB BBB BBB BB BBBBB BBB</CAPTION>
  <TR>
      <TH id="t1">CCC</TH>
      <TH id="t2">DDD</TH>
      <TH id="t3">EEE</TH>
      <TH id="t4">FFF</TH>
  <TR>
      <TD headers="t1">GGGG</TD>
      <TD headers="t2">HHHH</TD>
      <TD headers="t3">IIII</TD>
      <TD headers="t4">KKKK</TD>
  <TR>
      <TD headers="t1">LLLL</TD>
```

```
        <TD headers="t2">MMMM</TD>
        <TD headers="t3">NNNN</TD>
       <TD headers="t4">OOOO</TD>
</TABLE>
```

End example.

A user agent might render this tables as follows:

```
Caption: BB BBBBB BBB BBB BB BBBBB BBB
Summary: AAAAA AAAA AAAAA AA AA AAAA AA AAAA AAA  AAAA.
AAA: GGGG, BBB: HHHH, CCC: IIII, DDD: JJJJ
AAA: KKKK, BBB: LLLL, CCC: MMMM, DDD: NNNN
```

If no summary information is available from the author, the user agent should generate summary information based on the content of a table. The generated information should include information on th size of the table and if the table has any header (TH) elements.

Example(s):

Example.

```
 <TABLE>
   <CAPTION>BB BBBBB BBB BBB BB BBBBB BBB</CAPTION>
   <TR>
       <TH id="t1">CCC</TH>
       <TH id="t2">DDD</TH>
       <TH id="t3">EEE</TH>
       <TH id="t4">FFF</TH>
   <TR>
       <TD headers="t1">GGGG</TD>
       <TD headers="t2">HHHH</TD>
       <TD headers="t3">IIII</TD>
       <TD headers="t4">KKKK</TD>
   <TR>
       <TD headers="t1">LLLL</TD>
       <TD headers="t2">MMMM</TD>
       <TD headers="t3">NNNN</TD>
      <TD headers="t4">OOOO</TD>
</TABLE>
```

End example.

A user agent might render this tables as follows:

```
Caption: BB BBBBB BBB BBB BB BBBBB BBB
Summary: 4 columns, 3 rows, 1st row are headers
AAA: GGGG, BBB: HHHH, CCC: IIII, DDD: JJJJ
AAA: KKKK, BBB: LLLL, CCC: MMMM, DDD: NNNN
```

4 Provide alternative renderings of document information

Technique: Allow users to specify that tables be formatted serially, based on the type of information in the table. [Priority 1] [Go to table-serialization]

The following is an example of sample markup for a table with header information (provided with the TH element):

Example(s):

Example.

```
<TABLE>
<CAPTION>A 2x2 data table with headers</CAPTION>
<TR>
  <TH></TH>
  <TH>A</TH>
  <TH>B</TH>
</TR>
<TR>
  <TH>C</TH>
  <TD>Data 11</TD>
  <TD>Data 12</TD>
</TR>
<TR>
  <TH>D</TH>
  <TD>Data 21</TD>
  <TD>Data 22</TD>
</TR>
</TABLE>
```

End example.

A serialization based on **row order** might be rendered as:

```
A 2x2 data table with headers

Row: C Column: A
Data 11

Row: C Column: B
Data 12

Row: D Column: A
Data 21

Row: D Column: B
Data 22
```

A serialization based on **column order** might be rendered as:

```
A 2x2 data table with headers

Column: A Row: C
Data 11

Column: A Row: D
Data 12

Column: B Row: C
Data 21

Column: B Row: D
Data 22
```

Tabular information can be confusing to users when they are using certain dependent user agents. 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 order or column order.

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. 17]) and the CSS 2 specification (see [CSS2], chapters 12, 17, and 19 [p. 17]) describe mechanisms for structuring tables, identifying headers, and rendering them in accessible ways.

User agents should allow users to select from several table formatting options for a given table to find the best rendering of the table. This is important for poorly formatted tables, for the user to have some repair strategy available through the user agent.

Technique: Allow users to view a document outline constructed from its structural elements (e.g., from header and list elements in HTML). [Priority 2] [Go to document-outline]

5 Make sure the user can maintain their orientation to information in a document

5.1 Provide the user with information on the content and structure of a document

Technique: Provide a mechanism for assistive technologies to identify which elements have associated scripts. [Priority 1] [Go to orientation-scripting]

Technique: Provide information about document changes resulting from the execution of a script. [Priority 1] [Go to orientation-scripting-changes]

Technique: Provide information when a script is executed. [Priority 2] [Go to orientation-elements]

Technique: When a document is loaded or when requested by the user, make available document summary information. [Priority 2] [Go to orientation-summary]

Technique: Provide a mechanism to indicate visually the presence of an "accesskey" attribute defined for a link or form control. [Priority 2] [Go to orientation-accesskey]

Technique: Provide the user with audio feedback about document loading information. Such information includes whether loading has stalled, whether enough of the page has loaded to begin navigating, whether following a link involves a fee, etc. [Priority 3] [Go to orientation-audio-feedback]

Technique: Provide a mechanism to distinguish visited links from unvisited links. [Priority 3] [Go to orientation-visited-links]

Technique: Allow the user to specify that images used in links must have borders. [Priority 3] [Go to orientation-link-borders]

5.2 Provide the user with relative position information with in a document

Technique: Provide the user with information about how much of the document has been viewed. [Priority 2] [Go to orientation-doc-position]

Technique: Provide the user with information about which table cell is the current table cell. [Priority 2] [Go to orientation-table-position]

5.3 Maintain document view and focus

Technique: Provide a mechanism for highlighting and identifying the current view, focus, and user selection. [Priority 1] [Go to orientation-doc-view]

Technique: Keep track of the user's point of regard in each view and put it within the viewport when the user returns to the view. [Priority 1] [Go to orientation-previous-focus]

Technique: Allow the user to specify that a view's focus should follow changes in the viewport. [Priority 1] [Go to orientation-focus-follow]

Technique: Allow user to be prompted before spawning a new window. [Priority 2] [Go to orientation-spawning-browser]

Technique: [Ed. How is this identified?] Allow the user to turn on and off automatic page forwarding. [Priority 3] [Go to orientation-page-forwarding]

6 Make sure the user can efficiently navigate the structure of a document

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

6.1 Sequential keyboard navigation to structures within and between views

Technique: Allow the user to navigate sequentially between links (including elements with long descriptions) or between form controls in the same view. [Priority 1] [Go to navigation-sequentially]

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

6.2 Direct keyboard navigation to structures within and between views

Technique: Allow the user to search for an element in the current document by its text content. [Priority 1] [Go to navigation-search]

If the search text is found, the selection should be moved to the occurrence. If the text occurs within an link, the focus should be changed to the anchor.

Technique: Allow the user to search for a link in the current document based on its link text or alt text. [Priority 2] [Go to navigation-search-links]

Text associated with links (including ALT text for images that are links) in the current document can be searched using keyboard commands for a given phrase. If the search text is found, the focus should be moved to the occurrence.

Technique: Allow the user to move the focus directly to a specific link in the current document. [Priority 2] [Go to navigation-link-list]

Keyboard commands to move the focus directly to links in a document by either entering a numerical value associated with the link from a numerically coded list of links or by using keyboard commands to search a list containing only the link text. Numbering should be in source order, not rendered order.

Technique: Allow the user to move the user selection directly to a specific element in the current document that is not an active element. [Priority 2] [Go to navigation-direct-elements]

Keyboard commands to move the selection directly to non-link and non-control elements in a document by either entering a numerical value associated with a numerically coded list of an element text and/or by using keyboard commands to search a list containing only the text associated with that element. Numbering should be in source order, not rendered order.

Technique: Allow the user to search for an element in the current document by its alternative content (e.g., the value of the "alt" and "title" attributes). [Priority 2] [Go to navigation-search-alt]

Technique: Allow the user to include the text contents of any long descriptions in the text searches described above. However, if matched text occurs within a long description, focus should be moved to the first element in the main document for which the long description was written. [Priority 3] [Go to navigation-search-longdesc]

6.3 Hierarchical keyboard navigation to structures within a view

Technique: Allow the user to use the keyboard to navigate the document tree. [Priority 2] or [Priority 3] [Go to navigation-hierarchical]

The user should be able to use keyboard or pointing input commands to navigate, expand, or contract the hierarchy. The hierarchy is defined by block level HTML elements like H1-H6, OL, UL, TABLE, MENU, DIV and etc.. For example a view could start by just showing H1 level headers. If the user selects the first H1 header the user agent would expose the headers and other block level items between the first H1 and the next H1.

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).

6.4 Keyboard navigation of tabular (table) structures within a view

Technique: Provide a mechanism for designating the *current cell* of a table. The current table cell may be designated with the user selection. [Priority 1] [Go to table-current-cell]

Technique: Allow the user to navigate among table cells (notably left/right within a row and up/down within a column). [Priority 1] [Go to navigation-table]

Technique: Allow the user to navigate to a specific table cell through its row/column coordinates, header information, or its content. [Priority 1] [Go to navigation-cell]

6.5 Allow keyboard navigation and activation of elements with associated scripts.

Technique: Allow the user to navigate elements with associated scripts (through the document language). Both sequential and direct navigation should be possible. [Priority 1] [Go to nav-scripts]

Technique: Allow the user to trigger events through redundant means. Users must be able to trigger mouse events with the keyboard and vice-versa. [Priority 1] [Go to nav-trigger-event]

7 Make sure information on accessibility is highly visible and accessible

7.1 Ensure that user agent accessibility features are configurable

Technique: Allow users to configure accessibility features easily and directly. [Priority 2] [Go to visibility-configuration]

Technique: Furnish predefined accessibility profiles for common disabilities. [Priority 2] [Go to visibility-profiles]

7.2 Provide obvious information about keyboard access

Technique: Display keyboard navigation shortcut commands in customizable menus. [Priority 2] [Go to visibility-menu-commands]

Technique: Provide a list of all keyboard commands (organized by key or by topic) in an accessible format. [Priority 2] [Go to visibility-keyboard-doc]

7.3 Provide accessible documentation

Technique: Provide a description of accessibility features in the on-line documentation. [Priority 2] [Go to visibility-online-accessinfo]

Technique: Provide a description of accessibility in printed documentation. [Priority 2] [Go to visibility-print-access]

Technique: Ensure that the online documentation interface is accessible. [Priority 2] [Go to visibility-online-accessible]

Technique: Provide print and on-line information in alternative formats for people with print impairments. [Priority 2] [Go to visibility-documentation-alt]

8 Make sure the User Agent is compatible with existing accessibility recommendations and APIs

User agents typically operate in within an operating system and in the context of other standards. User agents need to be compatible with operating accessibility

The following guidelines apply to user agents that implement Cascading Style Sheets (see CSS, level 1 [p. 17] and CSS, level 2 [p. 17]). 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.

Technique: Completely implement Cascading Style Sheets, level 1 [Priority 1] [Go to compatibility-css1]

Technique: Allow the user to turn off author styles represented by author style sheets. [Priority 1] [Go to compatibility-css2-style]

Technique: Allow the user to adjust default values represented by browser style sheets. [Priority 1] [Go to compatibility-css2-default]

Technique: Support the :before and :after pseudo-elements as defined in CSS2 ([CSS2], 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. [Priority 1] [Go to compatibility-css2-pseudo]

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.

Technique: Support the ’outline’ property as defined in CSS2 ([CSS2], section 18.4). Outlines may be used to customize the focus display (see also orientation information). [Priority 1] [Go to compatibility-css2-outline]

Outlines may be used to customize the focus display (see also orientation information).

Technique: Allow the user to specify user styles through style sheets (see [CSS2], section 6.4). [Priority 2] [Go to compatibility-css2-user]

Technique: Implement the !important rule as defined in CSS2 ([CSS2], section 6.4.2). These rules offer a way for users to override author styles and browser defaults [Priority 2] [Go to compatibility-css2-important]

Technique: Support aural cascading style sheets (see [CSS2], chapter 19) for the auditory presentation of documents. [Priority 2] [Go to compatibility-css2-aural]

8.1 Compatibility with HTML 4.0

Technique: Support the "longdesc" attribute defined for IMG elements ([HTML 4.0], section 13.2). This attribute may be used to attach additional descriptive information to images. [Priority 1] [Go to compatibility-html-longdesc]

This attribute may be used to attach additional descriptive information to images.

Technique: Support the "rel" and "rev" attributes defined in HTML ([HTML40], section 12.1.2). [Priority 2] [Go to compatibility-html-rel-rev]

These attributes may be used to identify D-links [p. 5] and other document relations pertinent for accessibility.

Technique: Support the CAPTION element ([HTML40], section 11.2.2) for rich table captions. [Priority 2] [Go to compatibility-html-caption]

The caption element is used to provide a table information for a table. It should be rendered before the table.

Technique: Support the "summary" attribute for TABLE ([HTML40], section 11.2.1) for table summary information. [Priority 2] [Go to compatibility-html-summary]

Technique: Support the NOSCRIPT element ([HTML40], sections 18.3.1 and 16.4.1) for accessible alternatives to scripts. [Priority 2] [Go to compatibility-html-noscript]

Technique: Support the NOFRAMES element ([HTML40], sections 18.3.1 and 16.4.1) for accessible alternatives to frames. [Priority 2] [Go to compatibility-html-noframes]

Technique: Support the "lang" attribute ([HTML40], section 8.1). [Priority 3] [Go to compatibility-html-lang]

Technique: Support the "tabindex" attribute ([HTML40], section 17.11.1) for assigning the order of keyboard navigation within a document. [Priority 3] [Go to compatibility-html-tabindex]

Technique: Support the "accesskey" attribute ([HTML40], section 17.11.2) for assigning keyboard commands to active elements such as links, and form controls. [Priority 3] [Go to compatibility-html-accesskey]

8.2 Compatibility with Third-party Assistive Technology

Standard OS Controls/Menus/Dialog boxes

Technique: Standard OS Controls/Menus/Dialog boxes [Priority 1] [Go to compatibility-os-controls]

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.

Accessibility Application Programming Interfaces

Technique: Accessibility Application Programming Interfaces [Priority 1] [Go to compatibility-os-accessibility-api]

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

Built-in Accessibility Options

Technique: Built-in Accessibility Options [Priority 1] [Go to compatibility-os-built-in]

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.

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 lock keys are pressed (Caps Lock, Scroll

Lock, Num Lock).
- 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.

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, option (alt) 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.

AccessX/The X Window System

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, Num Lock, Scroll Lock..).
- BounceKeys: Requires a delay between keystrokes before accepting the next keypress so users with tremors can prevent the system from accepting inadvertent keypresses.

9 Acknowledgments

Many thanks to the following people who have contributed through review and comment: Paul Adelson, James Allen, Denis Anson, Kitch Barnicle, Harvey Bingham, Judy Brewer, Kevin Carey, Wendy Chisholm, Chetz Colwell, Daniel Dardailler, Neal Ewers, Geoff Freed, Larry Goldberg, Markku Hakkinen, Chris Hasser, Kathy Hewitt, Phill Jenkins, Leonard Kasday, George Kerscher, Marja-Riitta Koivunen, Josh Krieger, Greg Lowney, Scott Luebking, William Loughborough, Napoleon Maou, 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, 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.

[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.

[WAI-GL-TECHNIQUES]
Techniques for "WAI Accessibility Guidelines: Page Authoring", G. Vanderheiden, W. Chisholm, and I. Jacobs, eds. This evolving document is available at:
http://www.w3.org/WAI/GL/wai-gl-techniques.