W3C WD-WAI-USERAGENT-19981019

WAI Accessibility Guidelines: User Agent

W3C Working Draft *19-Oct-1998*

This version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019
Latest version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT
Previous version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19980814
Related documents:
User Agent Techniques

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

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

Abstract

This document provides guidelines to user agent manufacturers (producers of browsers, players, etc.) for making their products more accessible to people with disabilities. Since accessible design is good design, following these guidelines will improve Web browsing for able-bodied users as well.

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
A plain text file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent.txt,
HTML as a gzip'ed tar file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/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-19981019/wai-useragent.zip,
A PostScript file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent.ps,
A PDF file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019/wai-useragent.pdf.

In case of a discrepancy between the various formats of the specification, http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19981019 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 General principles of accessible design 4
3 Terms and definitions 5
3.1 Documents, Elements, and Attributes 5
3.2 Properties, Values, and Defaults 5
3.3 Views, Point of Regard, Selection, Focus, and Events 6
3.4 Alternative content 7
3.5 Dependent vs. Independent user agents 7
4 Provide mechanisms to control rendering 7
4.1 Allow the user to control document styles 8
4.2 Provide access to alternative content 8
4.3 Allow the user to choose formatting solutions 9
5 Help the user remain oriented 9
5.1 Provide information about the content and structure of a document . 10
5.2 Provide information about elements with associated scripts . . . 10
5.3 Provide information about relative position within a document or documents 10
5.4 Provide information about document view and focus 11
6 Help the user navigate 11
6.1 Allow sequential keyboard navigation to structures within and between views 12
6.2 Allow direct keyboard access to structures within and between views 12
6.3 Allow keyboard navigation of the document tree 13
6.4 Allow keyboard navigation of tables 13
6.5 Allow keyboard navigation and activation of elements with associated scripts 13
7 Ensure that user agent accessibility features are apparent 14
7.1 Ensure that user agent accessibility features are configurable . . 14
7.2 Provide obvious information about keyboard access 14
7.3 Provide accessible documentation 14
8 Ensure compatibility with existing accessibility recommendations and APIs 15
8.1 Implement the accessibility features of CSS 15
8.2 Implement the accessibility features of HTML 16
8.3 Compatibility with Third-party Assistive Technology 16
9 Acknowledgments 16
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 General principles of accessible design

The following general principles of accessible user agent design underlie the guidelines listed in this document:

All users must be able to access all functionalities of a user agent.
: Users access these functionalities through *controls* (menus, buttons, keyboard shortcuts, etc.). Controls should be arranged in a manner consistent with their importance.

The user interface should be accessible to all users.
: The interface must be easy to understand regardless of the user's experience, knowledge, language skills, or current concentration level. Where the interface is not accessible, inaccessible functionality must be made available through accessible controls. Part of accessible access includes prompting the user effectively for input and providing useful feedback during and after each task.

The user interface must accommodate a wide range of individual preferences and abilities.
: Therefore, user interfaces must have redundant controls for the same functionality (e.g., menu, mouse, and keyboard equivalents) and must avoid, for example, mouse-only controls. Redundancy includes offering visual as well as aural representations of a control. User agents must also allow users to customize and configure the controls to meet their needs.

The user agent must render information in accessible forms.

The user agent must allow users to control the style of rendering (colors, fonts, etc.), and the format of rendering (e.g., serialization of tables). Users must also be able to access both primary and alternative content [p. 7] .

The user agent must facilitate navigation and orientation on a page and between pages.

The user agent must allow keyboard navigation at all times. The user agent should provide Web page orientation information (the number of links on a page, the number of form controls, the number of images, etc.) so the user can quickly grasp content and context.

The user agent must be compatible with third-party [p. 16] assistive technologies.

3 Terms and definitions

The guidelines in this document rely on a certain number of common document and user agent features. The following sections define terms that describe these features. These definitions have been worded to make the guidelines usable but also to allow some freedom of interpretation.

3.1 Documents, Elements, and Attributes

A document may be seen as a hierarchy of *elements*. Elements are defined by a language specification (e.g., HTML 4.0 or an XML application). Each element may have content, which generally contributes to the document's content. Elements may also have *attributes* that take values. Some attributes are integral to document accessibility (e.g., the "alt", "title", and "longdesc" attributes in HTML).

An element's *rendered content* is that which a user agent renders for the element. Often, this is what lies between the element's start and end tags. However, some elements cause external data to be rendered (e.g., the IMG element in HTML). At times, a browser may render the value of an attribute (e.g., "alt", "title" in HTML) in place of the element's content. Rendering is not limited to only visual displays, but can also include audio (speech and sound) and tactile displays (Braille and haptic displays).

3.2 Properties, Values, and Defaults

A user agent renders a document by applying formatting algorithms and style information to the document's elements. Formatting depends on a number of factors, including whether the document is being rendered on the screen, on paper, through speakers, or on a braille device. Style information (e.g., font and color information) may come from the elements themselves (e.g., certain style attributes in HTML), from style sheets, or from user agent settings. For the purposes of these guidelines, each formatting or style option is governed by a *property* and each property may take one value from a set of legal values.

The value given to a property by a user agent when it is started up is called the property's *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.).

Once the user agent is running, the value of a property for a given document or part of a document may be changed from the default value. The value of the property at a given moment is called its *current value*. Note that changes in the current value of a property do not change its default value.

Current values may come from documents, style sheets, scripts, or the user interface. Values that come from documents, their associated style sheets, or via a server are called *author styles*. Values that come from user interface settings, user style sheets, or other user interactions are called *user styles*.

3.3 Views, Point of Regard, Selection, Focus, and Events

A browser offers one or more *views* of a document. A view is a way of perceiving a document: a full view, a summary view, a filtered view, etc. The user views the document through a *viewport*, which may be a window, frame, a piece of paper, a magnifying glass, etc. Each view has only one viewport, but several views may have the same viewport (e.g., the same browser window may be used for different views of different documents as one browses the Web).

The dimensions of the viewport may restrict the view of a document. User agents generally allow users to move the viewport over the complete view (e.g., through scrollbars or some other panning mechanism).

Some of the guidelines below involve tracking the user's *point of regard* in the view. The point of regard describes where the user's attention lies in the view. Generally, the point of regard is constrained to the part of the view exposed by the viewport. As the guidelines below state, user agents should avoid displacing the viewport away from the user's point of regard as this can disorient users.

When a document is rendered aurally (or serially in generally), the point of regard is defined as that which is "currently" being rendered. Identifying the point of regard in a two-dimensional space is a more difficult task as users may be "looking at" any point on the screen or page.

Depending on the guideline, the user's point of regard in a two-dimensional space may be identified by:

The *insertion point*.
: In an editor/browser, the insertion point is the location where text editing takes place.

The *user selection*
: The user selection is defined here as the part of a document (possibly spanning several elements) identified for cut/copy/paste operations. The user selection may be set by the user (e.g., by a pointing device or the keyboard) or through an application programming interface (API). A view may have only one user selection. When several views co-exist, each may have a user selection, but only one is active, called the *current user selection*.

  The user selection 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 to read. Most screen readers are sensitive to highlight colors. Third-party [p. 16] assistive technologies may provide alternative presentation of the selection through speech, enlargement, or dynamic Braille display.

  The user selection may be used, for example, to identify the "current cell" of a table that the user is navigating.

The *user focus*.
: The user focus designates an element that the user may interact with. Such elements are called *active elements*. In HTML, active elements are defined as links, form controls, elements with a value for the "longdesc" attribute, and elements with associated scripts. An element with the focus may be *activated* through any number

of mechanisms, including the mouse, keyboard, an API, etc. The meaning of activation depends on the element. For instance, when a link is activated, the user agent generally retrieves the linked resource, which may be another Web page, program, etc. When a form control is activated, it may change state (e.g., check boxes) or may take user input (e.g., a text field). Activating an element with a script assigned for that particular activation mechanism (e.g., mouse down event, key press event, etc.) causes the script to be executed.

A view has only one focus. When several views co-exist, each may have a focus, but only one is active, called the *current focus*.

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

Certain *events* (document loading or unloading events, mouse press or hover events, keyboard events, etc.) may cause the browser to perform tasks. For instance, when a mouse button is released over a link, the link is activated, possibly causing a new document to be loaded into the browser. Events may occur due to user interaction or through other means such as scripting. **Note.** The interaction of HTML, style sheets, the Document Object Model [DOM1] [p. 17] and scripting is commonly referred to as "Dynamic HTML" (DHTML). However, as there is no W3C specification that formally defines DHTML, the guidelines will only use the term scripting.

3.4 Alternative content

Certain types of content (e.g., images, audio, video, applets, etc.) may not be accessible to all users, so user agents must ensure that users have access to author-supplied *alternative content*. The techniques document describes the different mechanisms authors may use to supply alternative content.

3.5 Dependent vs. Independent user agents

An *independent* user agent only requires the source document (and associated style sheets, etc.) to render the document's content. It may use an accessibility API to retrieve information about a document.

A *dependent* user agent relies on the output of some other user agent in order to render a page. Dependent user agents include screen magnifiers and screen readers, both of which rely on the visual output of another user agent.

---

4 Provide mechanisms to control rendering

In order to access a document, some users may require that it be rendered in a manner vastly different from that which the author intended. Users who are color-blind may not be able to perceive certain color combinations. Users with weak vision may require large fonts. Sightless users may require audio rendering.

User agents must therefore allow the user to have the final say over:

- The document's style (e.g., fonts, colors, etc.)
- The document's formatting: whether the document is presented textually, graphically, serially, aurally, tactiley, or some combination of these.
- The document's content. This means whether "normal" content or alternative content or both are rendered.

The following guidelines address user control over rendering.

4.1 Allow the user to control document styles

The user must be able to control the colors, fonts and other stylistic aspects of a document and to override author styles and user agent default styles. Otherwise, users who are **blind**, have **low vision**, some types of **learning disabilities,** or **any user who cannot or has chosen not to view the primary representation of information** will not know content of the information on the page.

Techniques

1. Allow the user to override author styles [p. 6] and browser defaults [p. 5] , and in particular those properties [p. 5] affecting: font face, font size, foreground and background colors, background images, user selection [p. 6] highlight colors, and user interaction (e.g., hover styles). [Priority 1] [Go to user-preference]
2. 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]
3. Allow the user to specify custom settings in profiles [p. 15] that may be shared (by other users or software). Preferably, this should be done via style sheets [p. 15] . 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]
4. Allow the user, through a keyboard command, to switch between browser default values [p. 5] and the user profile [p. 15] . [Priority 3] [Go to user-switch]

4.2 Provide access to alternative content

User agents must give users access to author-supplied alternative content [p. 7] (descriptions of images, video clips, sounds, applets, etc.) Some users cannot perceive the primary content due to a disability or a technological limitation (e.g., browser configured not to display images) If users cannot access alternative content, users who are **blind**, have **low vision**, some types of **learning disabilities,** or **any user** who cannot or has chosen not to view the primary representation of information will not understand the intention or content of the page.

General techniques

1. 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]
2. 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]
3. When no alternative text representation is available, indicate what type of object is present. [Priority 1] [Go to alt-no-alt]
4. Allow users to hide the display of D-links used to provide access to long description information. [Priority 3] [Go to alt-d-link]

Techniques for multimedia objects

In the following guidelines, multimedia players may be stand-alone software or plug-in applications.

1. Allow the user to identify and turn on/off text captions of audio. [Priority 1] [Go to multimedia-caption]
2. Allow the user to control the size, color, and background color of captions. [Priority 1] [Go to multimedia-rendering]
3. Allow the user to identify and turn on/off audio descriptions of video. [Priority 1] [Go to multimedia-descriptions]
4. Ensure that text media objects may be identified by third-party assistive technologies. [Priority 1] [Go to multimedia-compatibility]
5. Make accessibility-related information from the OS user profile available to the media player. [Priority 1] [Go to multimedia-os-options]
6. Allow users to reposition captions. [Priority 2] [Go to multimedia-positioning]
7. Allow users to control (dynamically) the rendering rate of audio media objects. [Priority 2] [Go to multimedia-speed]

Techniques for other types of objects

1. 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]
2. 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. 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]

4.3 Allow the user to choose formatting solutions

By offering several formatting solutions to users, user agents allow them to select the solution most adapted to their needs. For instance, users with screen readers will have difficulty understanding some tables whose cell contents exceed one line. In this case, if the independent user agent can serialize the table (i.e., render it one cell at a time), the screen reader's output will not cause confusion. Similarly, some users may require that a user agent present an outline view [p. 6] of a document so they may have an easier time navigating the document.

Techniques

1. Allow users to specify that tables be formatted serially, based on the type of information in the table. [Priority 1] [Go to table-serialization]
2. 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 Help the user remain oriented

Orientation - the sense of where one is within a document or series of documents - is fundamental to a successful Web experience. Authors may contribute to orientation through sight maps, navigation bars, visual associations between documents using frames, etc. User agents also facilitate orientation with proportional scrollbars on viewports [p. 6] . Not all users can use visual orientation clues, however, so user agents must complement them by:

- Providing support for a point-of-regard [p. 6] for people relying on assistive technology.
- Providing summary information about certain elements (e.g., active elements [p. 6]) in all or part of a document.
- Supporting, when appropriate for the target medium, speech and tactile identification of elements.

If users cannot orient themselves to the types of elements in a document, users who are **blind**, have **low vision**, some types of **learning disabilities,** or **any user who cannot or has chosen not to view the author's representation of information** will have incomplete knowledge of the content of the document.

5.1 Provide information about the content and structure of a document

Techniques

1. When a document is loaded or when requested by the user, make available document summary information. [Priority 2] [Go to orientation-summary]
2. 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]
3. 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]
4. Provide a mechanism to distinguish visited links from unvisited links. [Priority 3] [Go to orientation-visited-links]
5. Allow the user to specify that images used in links must have borders. [Priority 3] [Go to orientation-link-borders]

5.2 Provide information about elements with associated scripts

Techniques

1. Provide a mechanism for assistive technologies to identify which elements have associated scripts. [Priority 1] [Go to orientation-scripting]
2. Provide information when a script is executed. [Priority 2] [Go to orientation-elements]
3. Provide information about document changes resulting from the execution of a script. [Priority 1] [Go to orientation-scripting-changes]

5.3 Provide information about relative position within a document or documents

Users that are viewing documents through serial channels of perception like speech (since speech is temporal in nature) and tactile displays (current tactile technology is limited in the amount of information that can be displayed) have difficultly maintaining a sense of their relative position in a document. The meaning of "relative position" depends on the situation. It may mean the distance from the beginning of the document to the point of

regard [p. 6] (how much of the document has been read), it may mean the cell currently being examined in a table, or the position of the current document in a set of documents.

Techniques

1. Provide the user with information about how much of the document has been viewed. [Priority 2] [Go to orientation-doc-position]
2. Provide the user with information about which table cell is the current table cell [p. 13] . [Priority 2] [Go to orientation-table-position]
3. Support the "rel" and "rev" attributes defined in HTML ([HTML40] [p. 17] , section 12.1.2). [Priority 2] [Go to compatibility-html-rel-rev]

5.4 Provide information about document view and focus

For people with visual impairments, it is important that the point of regard [p. 6] remain as stable as possible. For instance, when returning to a document previously viewed, the user's previous point of regard on the document should be restored. The user agent should not disturb the user's point of regard by shifting focus to a different frame or window when an event occurs without notifying the user of the change. Disturbing the user's point of regard may cause problems for users who have **movement** impairments, who are **blind**, who have **low vision**, who have certain types of **learning disabilities,** or for **any user who cannot or has chosen not to view the authors representation of information** .

Techniques

1. Provide a mechanism for highlighting and identifying the current view, focus, and user selection. [Priority 1] [Go to orientation-doc-view]
2. 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]
3. Allow the user to specify that a view's focus should follow changes in the viewport. [Priority 1] [Go to orientation-focus-follow]
4. Allow user to be prompted before spawning a new window. [Priority 2] [Go to orientation-spawning-browser]
5. [Ed. How is this identified?] Allow the user to turn on and off automatic page forwarding. [Priority 3] [Go to orientation-page-forwarding]

6 Help the user navigate

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

- By changing the position of the viewport [p. 6] (e.g., by scrolling down the page).
- By shifting focus from one active element to the next of the same type (e.g., from link to link).
- By navigating among documents.

For all of the navigation techniques described below, the user agent must allow the user to navigate via the keyboard.

6.1 Allow sequential keyboard navigation to structures within and between views

Often, users wish to navigate a linear sequence of elements or related documents (frames) sequentially, e.g., to the next or previous document. If sequential navigation is not available users with some types of **movement impairments**, **visual impairments** and **learning disabilities** may not be able to access the links and controls in a document.

Techniques

1. 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]
2. 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 Allow direct keyboard access to structures within and between views

For comfortable browsing, users may with to navigate to a particular element of a certain type. For instance, in a document with many links, sequential navigation will require a great number of key strokes to reach a desired element. In addition to being wearisome, this may be physically difficult for some users. Without direct access (e.g, by element identifier or element position), users with some types of **movement impairments**, **visual impairments** and **learning disabilities** may not be able to access the links and controls and other elements in a document efficiently.

[Ed. In the following text searches, what happens when the search succeeds? Is the user selection updated? The focus? Is the viewport moved?]

Techniques

1. Allow the user to search for an element in the current document by its text content. [Priority 1] [Go to navigation-search]
2. 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]
3. Allow the user to move the focus directly to a specific link in the current document. [Priority 2] [Go to navigation-link-list]
4. Allow the user to move the user selection directly to a specific element in the current document that is not an active element [p. 6] . [Priority 2] [Go to navigation-direct-elements]
5. Allow the user to search for an element in the current document by its alternative content [p. 7] (e.g., the value of the "alt" and "title" attributes). [Priority 2] [Go to navigation-search-alt]
6. 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 Allow keyboard navigation of the document tree

Hierarchical navigation (through the document tree) is useful for quickly navigating the major topics of a document. Topical navigation (section by section) conveys significant information about the organization of a document. If hierarchical navigation is not available users with some types of **movement impairments**, **visual impairments** and **learning disabilities** may not be to access the links and controls on a page efficiently nor understand the topics and topic relationships within a document.

Techniques

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

6.4 Allow keyboard navigation of tables

Tabular navigation is required by people with **visual impairments** and some types of **cognitive disabilities** to determine the content of a particular cell and spatial relationships between cells (which may convey information). If sequential navigation is not available users with some types of **visual impairments** and **learning disabilities** may not be able to understand the purpose of a table or table cell.

Techniques

1. 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]
2. 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]
3. 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

Users must be able to emulate events when their user agents do not allow them to trigger events in the anticipated way. For instance, users without mice must be able to trigger the same events that users with mice can trigger. Furthermore, some users may not realize when an event occurs because the event has an effect the user cannot perceive (e.g., a visual effect). The user agent must be able to inform users when events take place and to a certain extent, what has been the effect of the event.

Techniques

1. 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]
2. 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 Ensure that user agent accessibility features are apparent

Ideally, able-bodied as well as disabled users would be aware of user agent features that improve accessibility and experienced peers could share their knowledge with new users. In reality, most able-bodied peers do not know about features that improve accessibility or do not find them useful for their own use and so ignore them. Consequently, users with disabilities must find help information on their own to optimize the user agent to their preferences. This information must be readily available and in accessible formats.

7.1 Ensure that user agent accessibility features are configurable

The organization and layout of accessibility feature controls has a major impact on helping users with disabilities find and learn about configuration options and the range of browsing options available to them. If the user agent interface does not offer coherent and consistent means for configuring the software, users with disabilities may not be able to optimize the user interface to their preferences.

Techniques

1. Allow users to configure accessibility features easily and directly. [Priority 2] [Go to visibility-configuration]
2. Furnish predefined accessibility profiles [p. 15] for common disabilities. [Priority 2] [Go to visibility-profiles]

7.2 Provide obvious information about keyboard access

Keyboard access can be a boon to users having any of several disabilities. The more apparent the keyboard commands are to **all** users, the more likely it is that new users with disabilities ill find them. Readily available information about keyboard access is crucial to its effective use by users with **visual impairments** and some types of **movement impairments**, otherwise a user with disabilities may not think they can accomplish a particular task or may try to use a very inefficient technique with a pointing device, like a mouse.

Techniques

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

Documentation must be available in accessible formats for people with print impairments. If users with **visual impairments** and **learning disabilities** and **movement impairments** that limit the use of paper documents cannot access on-line or printed documentation efficiently, they may not know about user agent features or be able to use the user agent efficiently or at all.

Techniques

1. Provide a description of accessibility features in the on-line documentation. [Priority 2] [Go to visibility-online-accessinfo]
2. Provide a description of accessibility in printed documentation. [Priority 2] [Go to visibility-print-access]
3. Ensure that the online documentation interface is accessible. [Priority 2] [Go to visibility-online-accessible]
4. Provide print and on-line information in alternative formats for people with print impairments. [Priority 2] [Go to visibility-documentation-alt]

8 Ensure compatibility 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

8.1 Implement the accessibility features of CSS

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.

Techniques

1. Completely implement Cascading Style Sheets, level 1 [Priority 1] [Go to compatibility-css1]
2. Allow the user to turn off author styles [p. 6] represented by author style sheets. [Priority 1] [Go to compatibility-css2-style]
3. Allow the user to adjust default values [p. 5] represented by browser style sheets. [Priority 1] [Go to compatibility-css2-default]
4. Support the :before and :after pseudo-elements as defined in CSS2 ([CSS2] [p. 17] , 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]
5. Support the 'outline' property as defined in CSS2 ([CSS2] [p. 17] , section 18.4). Outlines may be used to customize the focus display (see also orientation information [p. 9]). [Priority 1] [Go to compatibility-css2-outline]
6. Allow the user to specify user styles [p. 6] through style sheets (see [CSS2] [p. 17] , section 6.4). [Priority 2] [Go to compatibility-css2-user]
7. Implement the !important rule as defined in CSS2 ([CSS2] [p. 17] , section 6.4.2). These rules offer a way for users to override author styles [p. 6] and browser defaults [p. 5] [Priority 2] [Go to compatibility-css2-important]
8. Support aural cascading style sheets (see [CSS2] [p. 17] , chapter 19) for the auditory presentation of documents. [Priority 2] [Go to compatibility-css2-aural]

8.2 Implement the accessibility features of HTML

Techniques

1. Support the "longdesc" attribute defined for IMG elements ([HTML 4.0] [p. 17] , section 13.2). This attribute may be used to attach additional descriptive information to images. [Priority 1] [Go to compatibility-html-longdesc]
2. Support the CAPTION element ([HTML40] [p. 17] , section 11.2.2) for rich table captions. [Priority 2] [Go to compatibility-html-caption]
3. Support the "summary" attribute for TABLE ([HTML40] [p. 17] , section 11.2.1) for table summary information. [Priority 2] [Go to compatibility-html-summary]
4. Support the NOSCRIPT element ([HTML40] [p. 17] , sections 18.3.1 and 16.4.1) for accessible alternatives to scripts. [Priority 2] [Go to compatibility-html-noscript]
5. Support the NOFRAMES element ([HTML40] [p. 17] , sections 18.3.1 and 16.4.1) for accessible alternatives to frames. [Priority 2] [Go to compatibility-html-noframes]
6. Support the "lang" attribute ([HTML40] [p. 17] , section 8.1). [Priority 3] [Go to compatibility-html-lang]
7. Support the "tabindex" attribute ([HTML40] [p. 17] , section 17.11.1) for assigning the order of keyboard navigation within a document. [Priority 3] [Go to compatibility-html-tabindex]
8. Support the "accesskey" attribute ([HTML40] [p. 17] , section 17.11.2) for assigning keyboard commands to active [p. 6] elements such as links, and form controls. [Priority 3] [Go to compatibility-html-accesskey]

8.3 Compatibility with Third-party Assistive Technology

Most operating systems have standard user interface controls and application programming interfaces that support third-party assistive technology in providing alternative user interfaces. Many operating systems also have built-in accessibility features that should be tested for compatibility with user agents.

1. Standard OS Controls/Menus/Dialog boxes [Priority 1] [Go to compatibility-os-controls]
2. Built-in Accessibility Options [Priority 1] [Go to compatibility-os-built-in]
3. Accessibility Application Programming Interfaces [Priority 1] [Go to compatibility-os-accessibility-api]

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.