User Agent Accessibility Guidelines

W3C Working Draft *10-Feb-1999*

This version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210
Latest version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT
Latest public version:
http://www.w3.org/TR/WD-WAI-USERAGENT
Previous version:
http://www.w3.org/TR/1998/WD-WAI-USERAGENT-19981112
Related documents:
Techniques for User Agent Accessibility Guidelines

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

Abstract

This document provides guidelines to user agent manufacturers for making their products more accessible to people with disabilities and for increasing usability for all users. This 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.

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.

Please note that previous versions of this document were entitled "WAI User Agent Guidelines".

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-19990210/wai-useragent.html
A plain text file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent.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.ps,
A PDF file:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19990210/wai-useragent.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.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 Introduction

For those unfamiliar with accessibility issues pertaining to user agents, consider that many users may browse the Web in contexts very different from yours:

- They may not be able to see, hear, move their hands to control, or understand pages as easily or in the same was as "typical" users.
- They may not use a keyboard or mouse.
- They may have a slow connection or a small screen.
- They may be using a different operating system.

The guidelines in this document are designed to help developers understand and thereby reduce accessibility barriers that impede access to the Web. The document provides practical solutions based on existing and upcoming technologies. Though developers may believe that implementing accessibility features in their products is difficult or of limited use, considering accessibility during the design phase of a product leads to more flexible, manageable, and extensible software: accessible design is good design.

These guidelines include information relevant to a wide class of user agents: graphical desktop browsers, screen readers, speech synthesizers, multimedia players, text browsers, voice browsers, etc., with a particular focus on two categories:

1. Graphical desktop browsers, called *independent user agents* in this document.
2. *Assistive technologies*, called *dependent user agents* in this document since they rely on the output of independent user agents. For instance, screen magnifiers, which benefit people will low vision, magnify a portion of what another browser has rendered on the screen. Similarly, screen readers interpret information sent to the screen by browsers and direct it either to speech output or refreshable braille output.

The guidelines emphasize interoperability between these two categories of user agents.

This document is organized according to several principles that will improve the design of any type of user agent:

Ensure that the user interface is accessible. [p. 9]

The topic of accessible user interfaces for computer software in general exceeds the scope of this document; user interfaces must be intuitive, simple, and tested. Features that are known to promote accessibility should be made obvious to users and easy to find.

This document addresses some aspects of user interface that directly impact accessibility, including device-independence, accessible product documentation, and configurability.

Ensure that content is accessible. [p. 12]

The user agent must allow users to control the style (colors, fonts, speech rate, speech volume, etc.) and format of a document. Users must also be able to access both primary and alternative representations of content. [p. 9]

The user agent must enable navigation among different parts of the document: links, form controls, tables, etc., in an input device-independent manner. Navigation should follow any tabbing order that has been specified by the author. How tabbing order is defined typically depends on markup language (e.g., the "tabindex" attribute in HTML).

Help the user remain oriented [p. 16]
: The user agent should provide users with information about views of a document, document structure, document summary information, and events to help the user remain oriented. Perhaps more importantly, the user agent should minimize the chances the user will become disoriented by allowing the user to control when windows are spawned, warning the user when events occur, avoiding sudden movements of the viewport, etc.

Use standard interfaces and operating system conventions. [p. 19]
: Not all user agents satisfy the above principles natively, so they must make information available to assistive technologies. Even when a user agent implements a feature natively, it should make information available to other software. Information must be made available through interoperable and standard interfaces and generally according to operating system conventions (except where those conventions hinder accessibility).

2 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 guidelines and checkpoints. The techniques document contains practical advice on how to satisfy the checkpoints.

2.1 Checkpoint priorities

Each checkpoint in this document is assigned a priority that indicates its importance for users.

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

[Priority 2]
: This checkpoint **should** be implemented by user agents as a native feature or through compatibility with assistive technology, otherwise one or more groups of users will find it difficult to access information. Satisfying this checkpoint will

significantly improve access to the Web for some individuals.

[Priority 3]

This checkpoint **may** be implemented by user agents as a native feature or through compatibility with assistive technology, to make it easier for one or more groups of users to access information. Satisfying this checkpoint will improve access to the Web for some individuals.

2.2 Conformance

The terms "must", "should", and "may" (and related terms) are used in this document in accordance with RFC 2119 ([RFC2119] [p. 20]).

This document defines two categories of conformance in order to promote a standard of accessibility within, and interoperability between, two important classes of user agents - graphical desktop browsers and dependent assistive technologies.

Desktop graphical user agents

To conform to this document, a desktop graphical user agent must:

1. Satisfy all the Priority 1 checkpoints listed in this document explicitly marked as applying to that class of user agent, and
2. Satisfy those checkpoints **natively** (i.e., no additional software is required) unless the checkpoint explicitly indicates that it may be satisfied through communication with other software.

Even for those checkpoints that must be satisfied natively, desktop graphical user agents should make information available to other software through standard interfaces (e.g., specialized dependent user agents may provide a better solution to a problem than a desktop user agent).

Dependent user agents

To conform to this document, a dependent user agent must:

1. Satisfy all the Priority 1 checkpoints listed in this document explicitly marked as applying to that class of user agent.
2. Satisfy those checkpoints **natively** (i.e., no additional software is required) unless the checkpoint explicitly indicates that it may be satisfied through communication with other software.

All user agents

User agents are not required to satisfy a checkpoint in the following cases:

1. Not all user agents render all content types (video, sound, applets, etc.) natively; rendering may be handled by a plug-in or external program, for example. A user agent is **not** required to satisfy a checkpoint that refers to a specific content type unless:
   - the user agent renders the content natively, and
   - the user agent is able to recognize content as being of that type (e.g., through MIME types, known markup, style sheet properties, etc.).
2. A user agent is **not** required to satisfy checkpoints related to the properties of an object (e.g., video or animation rate) unless it has access to those properties.

Verification that a checkpoint has been satisfied lies outside of the scope of this document and the activities of the WAI User Agent Working Group. The checkpoints are expressed in language intended to facilitate verification by other parties.

Please note that lack of conformance does not imply lack of accessibility. However, the WAI User Agent Working Group believes that a user agent that conforms to this document is more likely to be accessible than one that does not.

The conformance mechanisms defined here reflect the weight that the WAI User Agent Working Group assigns to the Priority 1 checkpoints. However, the Working Group also recommends that user agent developers satisfy as many checkpoints as possible, including Priority 2 and 3 checkpoints.

3 Terms and definitions

This section defines terms used in this document.

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 where the document is rendered: on screen, paper, through speakers, a braille device, a mobile device, etc. Style information (e.g., fonts, colors, voice inflection, etc.) 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 term "property" in this document has the meaning ascribed in the CSS2 Recommendation [p. 20] .)

The value given to a property by a user agent when it is started up is called the property's *default value.* User agents 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 and Viewports

User agents may handle different types of source information: documents, sound objects, video objects, etc. The user perceives the information through a *viewport*, which may be a window, frame, a piece of paper, a speaker, a virtual magnifying glass, etc. A viewport may contain another viewport (e.g., nested frames, plug-ins, etc.).

User agents may render the same source information in a variety of ways; each rendering is called a *view*. For instance, a user agent may allow users to view a document in one window and a generated list of headers for the document in another.

The view is *how* source information is rendered and the viewport is *where* it is rendered.

Generally, viewports give users access to all rendered information, though not always at once. For example, a video player shows a certain number of frames per second, but allows the user to rewind and fast forward. A visual browser viewport generally features scrollbars or some other paging mechanism that allows the user to bring the rendered content into view.

The content currently available in the viewport is called the user's *point of regard*. The point of regard may be a two dimensional area (e.g., for graphical rendering) or a single point (e.g., for aural rendering or voice browsing). User agents should not change the point of regard unexpectedly as this can disorient users.

3.4 Selection, Focus, and Insertion Point

User agents generally offer several mechanisms for referring to parts of content in a view:

The *user focus*

: The user focus designates an *active component* in a document. Which elements are active depends on the document language. In HTML documents, for example, active components are defined to be 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 effect of activation depends on the component. 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 a component 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.* The current focus is generally highlighted.

The *user selection*

: The user selection generally specifies a range of text in a document. The range may be restricted to the content of a single element or may span several elements. The selection may be used for a variety of purposes: for cut and paste operations, to designate a specific element in a document, to identify what a screen reader should read, etc.

  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 is generally highlighted. On the screen, the selection may be highlighted using colors, fonts, graphics, or other mechanisms. Highlighted text is often used by dependent user agents [p. 4] to indicate through speech or braille output what the user wants to read. Most screen readers are sensitive to highlight colors. Dependent user agents may provide alternative presentation of the selection through speech, enlargement, or dynamic braille display.

The *insertion point.*

The insertion point is the location where document editing takes place. The insertion point may be set by the user (e.g., by a pointing device or the keyboard editing keys) or through an application programming interface (API). A view may have only one insertion point. When several views co-exist, each may have an insertion point, but only one is active, called the *current insertion point*

The insertion point is generally rendered specially (e.g., on the screen, by a vertical bar or similar cursor).

Both the current focus and the current user selection must be in the same view, called the *current view.* The current view is generally highlighted when several views co-exist.

3.5 Events and scripting

When certain *events* occur (document loading or unloading events, mouse press or hover events, keyboard events, etc.), user agents often perform some task (e.g., execute a script). For instance, in most user agents, when a mouse button is released over a link, the link is activated and the linked resource retrieved. User agents may also execute author-defined scripts when certain events occur. The script bound to a particular event is called an *event handler.* **Note.** The interaction of HTML, style sheets, the Document Object Model [DOM1] [p. 20] and scripting is commonly referred to as "Dynamic HTML" or DHTML. However, as there is no W3C specification that formally defines DHTML, this document will only refer to event handlers and scripts.

3.6 Alternative representations of 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 representations of content.* The techniques document describes the different mechanisms authors may use to supply alternative representations of content.

---

4 Ensure that the user interface is accessible

One of the keys to an accessible user interface is device-independence. Users must be able to access the functionality offered by a user agent’s user interface with a mouse, or the keyboard or whatever input devices the user agent supports.

Input devices include pointing devices, keyboards, braille devices, head wands, etc. Output devices include monitors, speakers, braille devices, and printers. Please note that "device-independent support" does not mean that user agents must support every input or output device, only that for those supported, the user agent offers redundant input and output mechanisms.

Note. In general, software should be designed, installed, configured, and documented according to operating environment conventions. However, if these conventions hinder accessibility, they should be avoided.

4.1 Ensure accessible product installation, documentation, and configuration

Documentation must be accessible so that users may learn about software features, notably those that relate to accessibility. Users who may not be able to access print material, including individuals with visual impairments, learning disabilities, or movement impairments, may be able to use accessible electronic documentation or documents in alternative hardcopy formats.

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

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

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

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

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

4.2 Support input and output device-independence

Users must be able to interact with the software and the document in a device-independent manner. This applies to input (e.g., no mouse-only commands) as well as output (e.g., error messages from the user agent should not be visual-only).

4.2.1 [Priority 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.

4.2.2 [Priority 1]
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.

4.2.3 [Priority 1]
Ensure that the user can activate the links in a document in an input device-independent manner.

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

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

4.3 Support accessible keyboard input

For those user agents that support keyboard input, the input methods must be accessible. Some users require single-key access, others require that keys activated in combination be close together, while others require that they be spaced far apart. Configurability is vital to accessible keyboard input.

The more apparent the keyboard commands are to all users, the more likely it is that new users with disabilities will find them and use them.

4.3.1 [Priority 2]
Allow the user to configure keyboard access to user agent functionalities. Configuration includes the ability to specify single- as well as multi-key access.
4.3.2 [Priority 2]
Ensure that user can find out about all keyboard bindings.
4.3.3 [Priority 2]
Provide a mechanism to render the presence of a keyboard binding for an active element.
4.3.4 [Priority 3]
Display keyboard bindings in menus.

4.4 Ensure that users can disable features that might interfere with accessibility

Users must be able to turn on and off support for features that may interfere with accessibility. User agents are only expected to provide control for content that it recognizes as an image, blinking text, etc. For example, an applet may cause text to blink but the user agent may not be able to detect this. A user agent should recognize text that blinks because of markup or style sheets. Details are provided in the techniques document.

See also the checkpoints related to user control of document styles. [p. 12]

4.4.1 [Priority 1]
Allow the user to turn on and off rendering of images.
4.4.2 [Priority 1]
Allow the user to turn on and off rendering of background images.
4.4.3 [Priority 1]
Allow the user to turn on and off rendering of video.
4.4.4 [Priority 1]
Allow the user to turn on and off rendering of sound.
4.4.5 [Priority 1]
Allow the user to turn on and off rendering of captions.
4.4.6 [Priority 1]
Allow the user to turn on and off animated or blinking text.
4.4.7 [Priority 1]
Allow the user to turn on and off animations and blinking images.

4.4.8 [Priority 1]
Allow the user to turn on and off support for scripts and applets.
4.4.9 [Priority 1]
Allow the user to turn on and off support for user style sheets.
4.4.10 [Priority 1]
Allow the user to turn on and off support for author style sheets.
4.4.11 [Priority 2]
Allow the user to turn on and off rendering of frames.
4.4.12 [Priority 1]
Allow the user to turn on and off support for spawned windows.
4.4.13 [Priority 3]
Allow the user to turn on and off automatic page forwarding.

5 Ensure that content is accessible

In order to access a document, some users may require that it be rendered in a manner other than what the author intended. Users who are color-deficient may not be able to perceive certain color combinations. Users with low vision may require large fonts. Users who are blind may require audio or tactile rendering. Users who are deaf may require captions for audio files.

User agents must therefore allow the user to control:

- The document's style (e.g., fonts, colors, aural parameters, etc.)
- The document's formatting: whether the document is presented textually, graphically, linearly, aurally, for tactile use, or some combination of these.
- The document's content. This means whether primary content or alternative representations of content or both are rendered.
- The user interface. Since authors may make changes to the user interface through scripting (e.g., by spawning new windows, causing dialog boxes to appear, etc.), users must be able to override changes that make the user agent or document inaccessible.

The following guidelines address user control over rendering.

5.1 Ensure that the user can 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 visual impairments, 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.

For the following checkpoints, the user must be able to override both author styles [p. 7] and user agent defaults [p. 7] . They apply to alternative representations of content [p. 14] as well as primary content.

Checkpoints for fonts and colors:

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

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

Checkpoints for applets and animations:

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

Checkpoints for video.

5.1.8 [Priority 1]
Allow the user to override video frame rates.
5.1.9 [Priority 2]
Allow the user to start, stop, pause, and rewind video.
5.1.10 [Priority 1]
Allow the user to control the position of captions.

Checkpoints for audio:

5.1.11 [Priority 1]
Allow the user to override audio playback rate.
5.1.12 [Priority 2]
Allow the user to start, stop, pause, and rewind audio.
5.1.13 [Priority 2]
Allow the user to override audio volume.

Checkpoints for speech:

5.1.14 [Priority 1]
Allow the user to override speech playback rate.
5.1.15 [Priority 2]
Allow the user to override speech volume.

Checkpoints for changes to the user interface:

5.1.16 [Priority 2]
When new windows or user interface components are spawned, allow the user to control window size and position.

5.2 Provide access to alternative representations of content and control of its rendering

User agents must give users access to author-supplied alternative representations of content [p. 9] (descriptions of images, captions for video or audio, etc.) since some users cannot perceive the primary content due to a disability or a technological limitation (e.g., browser configured not to display images).

See also the section on links to long descriptions of content [p. 15] .

General checkpoints:

5.2.1 [Priority 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.).

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

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

Checkpoints for audio:

5.2.4 [Priority 1]
Allow the user to choose from among available audio tracks.

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

Checkpoints for video:

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

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

5.3 Ensure that links are accessible

For all link navigation or searching, the focus [p. 8] is moved to the target link.

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

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

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

5.3.4 [Priority 1]
Allow the user to move the focus to a link based on its integer (tabbing-order) position.
5.3.5 [Priority 2]
Provide the user with information about the number of links in a document.
5.3.6 [Priority 3]
Provide a mechanism (e.g., through style sheets) to distinguish visited links from unvisited links.
5.3.7 [Priority 3]
Allow the user to specify (e.g., through style sheets) that images used in links must have borders.

The following checkpoints refer to links to long descriptions of content. See also the section on alternative representations of content. [p. 9]

5.3.8 [Priority 1]
Allow the user to navigate among elements with associated long descriptions.
5.3.9 [Priority 2]
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.

5.4 Ensure that tables are accessible

Tables can be complex, multi-dimensional structures. As such, they may be difficult to understand for users that browse in essentially one dimension, i.e. for whom tables are rendered serially. Large tables pose particular problems since remembering cell position and header information becomes more difficult as the table grows.

Tables pose further problems because in many cases on the Web, they are used to lay out pages rather than organize true tabular data. It is difficult for assistive technologies to distinguish tables used for layout from "real tables", and therefore renderings may confuse users.

User agents can facilitate browsing of tables by providing access to specific table cells and their associated header information. How headers are associated with table cells is markup language-dependent.

Tabular navigation is required by people with visual impairments and some types of learning disabilities to determine the content of a particular cell and spatial relationships between cells (which may convey information). If table 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.

There may be a variety of ways for a user to identify a specific table in a document (e.g., with the selection). Similarly, there may be a variety of ways to identify a specific cell (e.g., with the selection or via its row column coordinates).

5.4.1 [Priority 1]
Provide access to the contents of a specific cell.
5.4.2 [Priority 1]
Provide access to header information for a specific table cell.
5.4.3 [Priority 1]
Allow the user to navigate among tables in a document.

5.4.4 [Priority 1]
Allow the user to navigate among table cells of a table (notably left/right within a row and up/down within a column).
5.4.5 [Priority 2]
Allow the user to search for a table cell based on its contents or header information.
5.4.6 [Priority 2]
Allow the user to select a table cell and find out its row/column coordinates in the table.
5.4.7 [Priority 2]
Allow the user to find out the dimensions of a table.
5.4.8 [Priority 2]
Allow the user to find out the number of tables in a document.

5.5 Ensure that forms are accessible

For all form control navigation or searching, the focus [p. 8] is moved to the target form control.

5.5.1 [Priority 1]
Allow the user to navigate among form controls within a form.
5.5.2 [Priority 1]
Allow the user to navigate among forms in a document.
5.5.3 [Priority 1]
Allow the user to search for a form control based on its text content.
5.5.4 [Priority 1]
Allow the user to search for a form control based on its attribute values.
5.5.5 [Priority 2]
Provide the user with information about the number of forms in a document.
5.5.6 [Priority 2]
Provide the user with information about the number of controls in a form.
5.5.7 [Priority 3]
Allow the user to request to be prompted before a form is submitted.

6 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. Visual clues that help orientation often abound:

- proportional scroll bars indicate how much of the document has been read
- frames suggested relationships among documents
- graphical site maps describe page organization

Not all users can use visual orientation clues, however, so user agents must complement them with other mechanisms, including:

- Contextual information about the document or views (e.g., how many views, how whether the document has finished loading, how much of the document has been read).
- Summary information about specific elements (e.g., the dimensions of a table).
- Context information about elements in the document (e.g., a cell’s position in a

table, the number of links in a document).
- Navigation sequentially among parts of the document (e.g., headers). See also checkpoints related to navigation among links [p. 14] , tables [p. 15] , and forms [p. 16] .
- Searching for information in the document. See also checkpoints related to searching in links [p. 14] , tables [p. 15] , and forms [p. 16] .
- Alerts when events occur during browsing (e.g., a window opens, a script is executed, etc.).

6.1 Provide information about the document view

Users that are viewing documents through linear 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. 8] (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.

For people with visual impairments, it is important that the point of regard [p. 8] 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 visual impairments, who have certain types of learning disabilities, or for any user who cannot or has chosen not to view the authors representation of information.

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

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

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

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

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

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

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

6.1.8 [Priority 2]
Provide the user with information about the number of viewports.

6.1.9 [Priority 2]
When a document is loaded or when requested by the user, make available document summary information.
6.1.10 [Priority 2]
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).
6.1.11 [Priority 3]
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.)

6.2 Provide information about document structure

Hierarchical navigation (through the document tree) is useful for efficiently navigating the major topics and sub-topics of a document.

6.2.1 [Priority 2]
Allow the user to view a document outline constructed from its structural elements (e.g., from header and list elements).
6.2.2 [Priority 2]
Allow the user to navigate the document tree.
6.2.3 [Priority 2]
Allow the user to navigate sequentially among headers.
6.2.4 [Priority 2]
Allow the user to navigate sequentially among block elements (e.g., paragraphs, lists and list items, etc.)
6.2.5 [Priority 2]
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.

6.3 Provide information about events

It is important to alert users, in an output device-independent fashion, when important events occur during a browsing session. To avoid confusion that the effects of scripts may cause, users should be notified when scripts are executed (or be able to disable scripts [p. 12] entirely). This is also important for security reasons; users should be able to decide whether to allow scripts to execute on their machines.

6.3.1 [Priority 1]
Allow the user to navigate among elements with associated event handlers.
6.3.2 [Priority 2]
Alert the user when scripts or applets are executed.
6.3.3 [Priority 3]
Provide information about document changes resulting from the execution of a script.

7 Follow W3C Recommendations and use standard interfaces

Many types of software: assistive technologies, scripting tools, automated test engines, etc., benefit from having access to user agent information.

7.1 Support language accessibility features

A user agent that supports a language should implement the accessibility features for the language. The techniques document lists the accessibility features of the following languages, defined by W3C specifications:

7.1.1 [Priority 1]
Support accessibility features defined by W3C Recommendations for HTML.
7.1.2 [Priority 1]
Support accessibility features defined by W3C Recommendations for CSS.
7.1.3 [Priority 1]
Support accessibility features defined by W3C Recommendations for SMIL.

7.2 Use and provide accessible interfaces to other technologies

When a user agent communicates with other software (dependent user agents, the operating system, plug-ins), must do so through applicable interfaces. To promote interoperability, open standards should be used wherever possible.

7.2.1 [Priority 1]
Implement DOM Level 1 [p. 20] and expose it to dependent user agents through a public interface.
7.2.2 [Priority 2]
Support other accessibility application programming interfaces (APIs) (e.g., Microsoft Active Accessibility, Sun Microsystems Java Accessibility)
7.2.3 [Priority 2]
Use standard interfaces defined for the operating system.

8 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.

9 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.