User Agent Accessibility Guidelines 1.0

W3C Working Draft 5-November-1999

This version:
http://www.w3.org/TR/1999/WD-WAI-USERAGENT-19991105
(plain text, postscript, pdf, gzip tar file of HTML, zip archive of HTML)
Latest version:
http://www.w3.org/TR/WAI-USERAGENT
Previous version:
http://www.w3.org/WAI/UA/WD-WAI-USERAGENT-19991029

Editors:
Jon Gunderson, University of Illinois at Urbana-Champaign
Ian Jacobs, W3C

Abstract

This document provides guidelines to user agent developers for making their products -- browsers, multimedia players, plugins -- accessible to people with disabilities. An accessible user agent allows users with disabilities to retrieve and view Web content or to enable access when used in conjunction with other software or hardware, called assistive technologies . These guidelines discuss the accessibility of the user agent as well as how the user agent communicates with assistive technologies such as screen readers, screen magnifiers, braille displays, and voice input software.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This is the "Last Call Working Draft" of "User Agent Accessibility Guidelines 1.0". The Last Call review period begins 5 November 1999 and ends 1 December 1999. Please send review comments before the review period ends to w3c-wai-ua@w3.org. Mailing list archives are available on the Web.

At their 3 November 1999 teleconference (3 November minutes), the Working Group decided to move the UA Guidelines to Last Call. By moving to Last Call, the Working Group asserts that it has met the requirement of its charter "to complete the development of user agent accessibility guidelines addressing accessibility of graphical, voice, and text browsers, multimedia players, and third-party assistive technologies which work in conjunction with browsers and multimedia players."

The Working Group anticipates asking the W3C Director to advance this document to Proposed Recommendation after the Working Group processes Last Call review comments and incorporates resolutions into the Guidelines.

The Working Group requests that reviewers consider the following issues:

- Checkpoints 10.1 and 10.2 require user agents to make available to users information about the current input configuration (e.g., keyboard input). These checkpoints have been assigned different priorities: Priority 1 for user-specified configuration and Priority 2 for author-specified configuration (e.g., access keys). The Working Group did not reach consensus on whether these two checkpoints should be merged into a single checkpoint, and what the priority of such a checkpoint would be.
- Checkpoint 6.1 (Priority 1) asks user agents to implement the accessibility features of supported specifications. In the Authoring Tool Guidelines Proposed Recommendation, checkpoints that refer to content accessibility do so by "Relative Priority". This means that the priority of the checkpoint in the UAGL depends on how much you wish to conform to the Web Content Accessibility Guidelines. There has been a suggestion to make this a checkpoint with a Relative Priority rather than Priority 1. The Working Group did not reach consensus on whether the burden of doing so (complicating the priority definition) outweighed the benefit of consistency among the three sets of Guidelines. Also, it is not clear that a Priority 3 requirement in WCAG would always be a Priority 3 requirement in UAGL (i.e., it may be more important to implement a feature than for the author to supply it). Comments on the proposal to make this checkpoint a Relative Priority checkpoint are welcome.

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 Web Accessibility Initiative. The goals of the WAI UA Working Group are discussed in the WAI UA charter. A list of the UA Working Group participants is available.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Table of Contents

An appendix to this document [UA-CHECKLIST] lists all checkpoints for convenient reference.

1. Introduction

For those unfamiliar with accessibility issues pertaining to user agent design, consider that many users may be accessing the Web in contexts very different from your own:

- They may not be able to see, hear, move, or may not be able to process some types of information easily or at all.
- They may have difficulty reading or comprehending text.
- They may not have or be able to use a keyboard or mouse.
- They may have a text-only screen, a small screen, or a slow Internet connection.
- They may not speak or understand fluently the language in which content is written or spoken.
- They may be in a situation where their eyes, ears, or hands are busy or interfered with (e.g., driving to work, working in a loud environment, etc.).

User agents must be designed to take into account the diverse functional requirements of users with disabilities. Software that follows the guidelines in this document will not only benefit users with disabilities, it will be more flexible, manageable, and extensible. The guidelines have been chosen according to some basic principles of accessible design, presented below.

1.1 Principles of Accessible Design

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.

The user must have access to the functionalities offered by the user agent through its user interface. Since some users cannot use some parts of the the user interface, it needs to be adaptable to their particular functional needs.

One requirement is that users be able to operate the user agent with a variety of input devices (mouse, keyboard, speech input, etc.) and ouptut devices (graphical display, speech output, etc.). Redundant input and output methods (accomplished through the standard input and output APIs supported by the user agent) help users operate the functionalities built into the tool (made available through menus, dialogs, toolbars and other user interface components) as well as those included as part of content (made available through links, form controls, applets, and other active elements).

User agents should provide access to functionalities in different ways to meet the skills and needs of different audiences:

- Contextual access (e.g., through cascading menus, through help systems, etc.) helps users with cognitive impairments and any users unfamiliar with the tool.
- Direct access (e.g., through keyboard or voice shortcuts) helps some users with motor limitations and speeds up use by experienced users.

The general topic of user interface accessibility for computer software exceeds the scope of this document. The guidelines do discuss some important user interface topics such as device-independence, configurability, and accessible product documentation. Software developers should also remember that 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. The Techniques Document ([UA-TECHNIQUES]) includes some references to general software accessibility guidelines.

Ensure that the user has access to content.

User agents must ensure access to content:

- By ensuring access to all text, video, sound, and other content, including alternative equivalents for content (e.g., "alt" attribute values in HTML, external long descriptions, etc.) and relationships among content (e.g., table cells and their headers).
- By allowing users to control content rendering parameters (text size, colors, synthesized speech rate and volume, etc.).
- By allowing users to navigate the content (e.g., with scrollbars, navigation of active elements , structured views, etc.).
- By making Web content and user agent information available to assistive technology through standard APIs.

Help orient the user.

User agents can help the user remain oriented in a page or site by supplying context, including:

- Browsing context. Users benefit from knowing the number of frames, the title of the current frame, whether loading for a page or video clip has finished or stalled, etc. Graphical clues about browsing context such as frames, proportional scroll bars, a visually highlighted selection, etc. help some, but not all users, so the context information must be available in a device-independent manner.
- Element context. For instance, users with blindness who navigate by surfing only the links of a page or presentation benefit from information that will help them decide quickly whether to follow the link: whether the link has already been visited, the type of the target resource, the length of an audio or video clip that will be started, whether the link involves a fee, etc.
- Summary information about specific elements (e.g., the dimensions of a table, the length of an audio clip, the structure of a form, etc.)

The user agent should also minimize the chances the user will become disoriented. User agents should:

- For changes to the content or viewport that the user does not initiate, allow the user to request notification when these changes occur. (e.g., when a viewport opens, a script is executed, etc.).
- Allow the user to return to known states. The "back" functionality is a valuable "undo" tool for returning to a known state.

Follow operating system standards and conventions and use open specifications.

Following platform and operating system standards and guidelines promotes accessibility in a number of ways:

- Observing platform and operating system conventions in user interface design, software installation, and software documentation improves usability.
- Using standard platform and operating system interfaces makes it possible for assistive technologies to access information predictably.

Communication through standard interfaces is particularly important for graphical desktop browsers, which must make information available to assistive technologies . Even when a user agent implements a feature natively, it should make relevant information available through standard interfaces. This will benefit assistive technologies, scripting tools, and automated test engines. It will also will promote modularity and software reuse.

1.2 How the Guidelines are Organized

The twelve guidelines in this document state general principles for the development of accessible user agents. Each guideline includes:

- The guideline number.
- The statement of the guideline.
- The rationale behind the guideline and some groups of users who benefit from it.
- A list of checkpoint definitions.

The checkpoint definitions in each guideline explain how the guideline applies to particular user agent features or behavior. Each checkpoint definition includes:

- The checkpoint number.
- The statement of the checkpoint.
- The priority of the checkpoint. Priority 1 checkpoints are highlighted through the use of style sheets.
- Optional informative notes, clarifying examples, and cross references to related guidelines or checkpoints.

- A link to a corresponding section of the Techniques Document ([UA-TECHNIQUES]), where the checkpoint is discussed in detail, including information about implementation and examples.

Each checkpoint is intended to be specific enough so that someone reviewing a user agent may verify that the checkpoint has been satisfied. **Note.** While the checkpoints have been designed to be verifiable, some may be difficult to verify without documentation from vendors about what features and APIs they support.

An appendix to this document [UA-CHECKLIST] lists all checkpoints for convenient reference.

1.3 Related Documents

A related document, entitled "Techniques for User Agent Accessibility Guidelines 1.0" ([UA-TECHNIQUES]), suggests techniques for satisfying each requirement. The Techniques Document has been designed to track changes in technology and implementation solutions and is expected to be updated more frequently than the current document.

"User Agent Accessibility Guidelines 1.0" is part of a series of accessibility guidelines published by the Web Accessibility Initiative (WAI) . The series also includes "Web Content Accessibility Guidelines 1.0" ([WAI-WEBCONTENT]) and "Authoring Tool Accessibility Guidelines 1.0" ([WAI-AUTOOLS]).

1.4 Document conventions

The following editorial conventions are used throughout this document:

- HTML element names are in uppercase letters (e.g., H1, BLOCKQUOTE, TABLE, etc.)
- HTML attribute names are quoted in lowercase letters (e.g., "alt", "title", "class", etc.)
- Links to definitions are highlighted through the use of style sheets.

1.5 Priorities

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

[Priority 1]
: This checkpoint **must** be satisfied by user agents as a native feature, 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 satisfied by user agents as a native feature, otherwise one or more groups of users will find it difficult to access information.

Satisfying this checkpoint will remove significant barriers to accessing Web documents.

[Priority 3]

This checkpoint **may** be satisfied by user agents as a native feature 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.

1.6 Conformance

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

User agents must satisfy natively all the applicable checkpoints for a chosen conformance level .

Conformance levels

This section defines three levels of conformance to this document.

- **Conformance Level "A"**: all Priority 1 checkpoints are satisfied
- **Conformance Level "Double-A"**: all Priority 1 and 2 checkpoints are satisfied
- **Conformance Level "Triple-A"**: all Priority 1, 2, and 3 checkpoints are satisfied

Note. Conformance levels are spelled out in text so they may be understood when rendered as speech.

Claims of conformance to this document must use one of the following two forms.

Form 1: Specify:

- The guidelines title: "User Agent Accessibility Guidelines 1.0"
- The guidelines URI: http://www.w3.org/TR/1999/WD-WAI-USERAGENT-19991105
- The conformance level satisfied: "A", "Double-A", or "Triple-A".
- The version number and operating system of the software covered by the claim.
- The date of the claim.
- A list of checkpoints that have been satisfied and which are considered not applicable. The appendix list of checkpoints may be used for this purpose (in tabular format or list format).

Example of Form 1:

This product conforms to W3C's "User Agent Accessibility Guidelines 1.0", available at http://www.w3.org/TR/1999/WD-WAI-USERAGENT-19991105, level Double-A.

Form 2: Include, on product packaging or documentation, one of three icons provided by W3C and for Web documentation, link the icon to the appropriate W3C explanation of the claim.

Note. *In the event this document becomes a W3C Recommendation, information about the icons and how to use them will be available at the W3C Web site. At that time the Working Group will also indicate how claims of conformance can deliver completed list of checkpoints.*

2. User Agent Accessibility Guidelines

Guideline 1. Support input and output device-independence

Ensure that the user can interact with the user agent (and the content it renders) through all of the input and output APIs used by the user agent.

Since not all users make use of the same hardware for input or output, software must be designed to work with the widest possible range of devices. For instance, not all users have pointing devices, so software must not rely on them for operation. Users must be able to reach all functionalities offered by the user agent interface with all input device APIs provided by the operating system.

The best way to make this possible is to design software that follows operating system conventions and uses standard APIs for user input and output. When user agents use these standard interfaces, other software can programmatically trigger mouse or keyboard events. For instance, some users who may not be able to enter text easily through a standard physical keyboard can still use voice input, an on-screen keyboard, or other special devices to operate the user agent.

Access to user agent functionality through the operating system's standard keyboard API (where available) is vital to ensure compatibility between graphical desktop browsers and assistive technologies . Access through the keyboard is available to many users and is widely supported.

Since not all users have speakers or the ability to hear, software must not rely on audio output alone for messages and alerts. Any output provided in audio should also be available through other means (e.g., visual flashes for beeps, text messages for spoken messages, etc.). Text is perhaps the most accessible output medium, since most alternative output mechanisms rely on the presence of system-drawn text on the screen.

Standard interfaces make it possible for users to use a variety of input and output devices, including pointing devices, keyboards, braille devices, head wands, microphones, touch screens, speech synthesizers, and more.

1.1 Ensure that every functionality offered through the user interface is available through every input device API used by the user agent. User agents are not required to reimplement low-level functionalities (e.g., for character input or pointer motion) that are inherently bound to a particular API and most naturally accomplished with that API. [Priority 1]

> **Note.** The device-independence required by this checkpoint applies to functionalities described by the other checkpoints in this document unless otherwise stated by individual checkpoints. This checkpoint does not require user agents to use all operating system input device APIs, only to make the software accessible through those they do use.

1.2 Use the standard input and output device APIs of the operating system. [Priority 1]

For example, do not directly manipulate the memory associated with information being rendered since screen review utilities, which monitor rendering through the standard APIs, will not work properly.

1.3 Ensure that the user can interact with all active elements in a device-independent manner. [Priority 1]

For example, users who are blind or have motor impairments must be able to activate the links in a client-side image map without a pointing device. One technique for doing so is to render client-side image maps as text links. **Note.** This checkpoint is an important special case of checkpoint 1.1.

1.4 Ensure that every functionality offered through the user interface is available through the standard keyboard API . [Priority 1]

The keystroke-only command protocol of the user interface should be efficient enough to support production use. Functionalities include being able to show, hide, resize and move graphical viewports created by the user agent. **Note.** This checkpoint is an important special case of checkpoint 1.1.

1.5 Ensure that all messages to the user (e.g., informational messages, warnings, errors, etc.) are available through all output device APIs used by the user agent. Do not bypass the standard output APIs when rendering information (e.g., for reasons of speed, efficiency, etc.). [Priority 1]

For instance, ensure that information about how much content has been viewed is available through output device APIs. Proportional navigation bars may provide this information graphically, but the information must be available (e.g., as text) to users relying on synthesized speech or braille output.

Guideline 2. Ensure user access to all content

Ensure that users have access to all content, notably author-supplied alternative equivalents for content (descriptions of images, closed captions for video or audio, etc.)

Users may not be able to perceive primary content due to a disability or a technological limitation or configuration (e.g., browser configured not to display images). Markup languages may provide a number of mechanisms for specifying alternative representations of content: through attribute values, element content, or as separate resources. User agents must also take into account markup related to natural language rendering, using appropriate fonts, text directionality, and synthesized speech elements.

In dynamic presentations such as synchronized multimedia presentations created with SMIL (refer to [SMIL]), content changes over time. Users with cognitive or physical disabilities may not be able to interact with a presentation within the time frames designed by the author. To ensure that a presentation remains accessible, user agents rendering synchronized presentations must either provide access to content in a time-independent manner or allow users to control the playback rate of the presentation. For information about SMIL accessibility, please refer to

[SMIL-ACCESS] .

User agents should allow users to specify whether primary content should be rendered, or alternative equivalents supplied by the author, or both.

Mechanisms for specifying alternative content vary according to markup language. For instance, in HTML or SMIL, the "alt" attribute specifies alternative text for many elements. In HTML, the content of the OBJECT element is used to specify alternative content, the "summary" attribute applies to tables, etc. In HTML, the NOFRAMES element specifies alternative content for frames. The ability to access frame alternatives is important for users of some screen readers and users with some cognitive impairments.

2.1 Ensure that the user has access to all content, including alternative representations of content. [Priority 1]

> **Note.** Although it is not a requirement that alternative content be available at the same time as primary content, some users may benefit from simultaneous access. For instance, users with low vision may want to view images (even imperfectly) but require alternative text for the image to be rendered in a very large size or as speech.

2.2 If more than one alternative equivalent is available for content, allow the user to choose from among the alternatives. This includes the choice of viewing no alternatives. [Priority 1]

> For example, if a multimedia presentation has several tracks of closed closed captions (or subtitles) available (e.g., in different languages, different levels of detail, etc.) allow the user to choose from among them.

2.3 Render content according to natural language identification. [Priority 1]

> Natural language may be identified by markup (e.g., the "lang" attribute in HTML [HTML40] or "xml:lang" in [XML]) or HTTP headers. Refer also to checkpoint 2.9 and checkpoint 5.3.

2.4 Provide time-independent access to time-dependent active elements or allow the user to control the timing of changes. [Priority 1]

2.5 Allow the user to specify that continuous equivalent tracks (e.g., closed captions , auditory descriptions , video of sign language, etc.) be rendered at the same time as audio and video tracks. [Priority 1]

2.6 If a technology allows for more than one audio track, allow the user to choose from among tracks. [Priority 1]

2.7 When no text equivalent has been specified, indicate what type of object is present. [Priority 2]

2.8 When alternative text has been specified explicitly as empty (i.e., an empty string), render nothing. [Priority 3]

2.9 For identified but unsupported natural languages, notify the user of language changes when configured to do so. [Priority 3]

Guideline 3. Allow the user to turn off rendering or behavior that may reduce accessibility

Ensure that the user may turn off rendering or behavior specified by the author that may obscure content or disorient the user.

Some content or behavior specified by the author may make the user agent unusable or may obscure information. For instance, flashing content may trigger seizures in people with photosensitive epilepsy. Blinking can affect screenreader users, since screenreaders (in conjunction with speech synthesizers or braille displays) may repeat the text every time it blinks. Noisy background images or sounds make make it impossible for users to see or hear other content. Some color combinations may affect users with some visual impairments.

Dynamically changing Web content, scripts that open viewports , automatically forwarded or refreshed pages, and similar changes unanticipated by the user may disorient some users with cognitive disabilities and may cause problems for some assistive technologies .

Users may need to turn off these effects in order to have access to content. Please also refer to guideline 4 and guideline 10.

3.1 Allow the user to turn on and off rendering of background images. [Priority 1]
3.2 Allow the user to turn on and off rendering of background audio. [Priority 1]
3.3 Allow the user to turn on and off rendering of video. [Priority 1]
3.4 When the user agent renders audio natively, allow the user to turn on and off rendering of audio. [Priority 1]
3.5 Allow the user to turn on and off animated or blinking text. [Priority 1]
3.6 Allow the user to turn on and off animations and blinking images. [Priority 1]
3.7 Allow the user to turn on and off support for scripts and applets. [Priority 1]

> **Note.** This is particularly important for scripts that cause the screen to flicker, since people with photosensitive epilepsy can have seizures triggered by flickering or flashing in the 4 to 59 flashes per second (Hertz) range. Users should be able, for security reasons, to prevent scripts from executing on their machines.

3.8 Allow the user to turn on and off rendering of images. [Priority 3]
3.9 Allow the user to turn on and off author-specified forwards that occur after a time delay and without user intervention. [Priority 3]

> For example, when forwarding has been turned off, offer a static link to the target.

3.10 Allow the user to turn on and off automatic content refresh. [Priority 3]

> For example, when turned off, allow the user to refresh content manually instead (through the user interface).

Guideline 4. Ensure user control over styles

Ensure that the user has control over the colors, text size, speech rate and pitch, and other stylistic aspects of a resource and can override author styles and user agent default styles.

In order to access content, some users may require that it be rendered in a manner other than what the author intended. Users with visual impairments, including color blindness, may be insensitive to certain colors and may not be able to perceive author-specified or user agent default color combinations. Users with reduced visual acuity, including people who are older, may require larger text than user agent defaults or the text size specified by the author.

User agents must therefore allow the user to control :

- Style parameters (e.g., text size, colors, audio volume, speech pitch, video frame rate, etc.). Style information includes author-specified styles and user agent defaults .
- Aspects of the user interface. User agents must ensure access to selection and focus information and allow users to be notified of and to control author-specified changes to the browsing context that may make content inaccessible.

Note. The checkpoints in this guideline apply to all content, in including alternative representations of content.

Refer also to guideline 10.

Checkpoints for fonts and colors:

4.1 Allow the user to control font family. [Priority 1]
4.2 Allow the user to control the size of text. [Priority 1]
For example, allow the user to control font size through style sheets or the user interface. Or allow the user to magnify text.
4.3 Allow the user to control foreground color. [Priority 1]
4.4 Allow the user to control background color. [Priority 1]
4.5 Allow the user to control selection highlighting (e.g., foreground and background color). [Priority 1]
4.6 Allow the user to control focus highlighting (e.g., foreground and background color). [Priority 1]

Checkpoints for applets and animations:

4.7 Allow the user to control animation rate. [Priority 2]

Checkpoints for video.

4.8 Allow the user to control video frame rates. [Priority 1]
4.9 Allow the user to control the position of audio closed captions . [Priority 1]
4.10 Allow the user to start, stop, pause, and rewind video. [Priority 2]

Checkpoints for audio:

4.11 Allow the user to control audio playback rate. [Priority 1]
4.12 When the user agent renders audio natively, allow the user to control the audio volume. [Priority 2]
4.13 Allow the user to start, stop, pause, and rewind audio. [Priority 2]

Checkpoints for synthesized speech:

4.14 Allow the user to control synthesized speech playback rate. [Priority 1]
4.15 Allow the user to control synthesized speech volume. [Priority 1]
4.16 Allow the user to control synthesized speech pitch, gender, and other articulation characteristics. [Priority 2]

Checkpoints for the user interface:

4.17 Allow the user to select from available author, user, and user agent default style sheets. [Priority 1]

Note. Users should be able to select no style sheets (i.e., turn them off).

4.18 Allow the user to control user agent-initiated spawned viewports . [Priority 2]

For example, in HTML, allow the user to control the process of opening a document in a new target frame or a viewport created by author-supplied scripts. In SMIL 1.0, allow the user to control viewports created with show="new". Control may involve prompting the user to confirm or cancel the viewport creation. Users may also want to control the size or position of the viewport and to be able to close the viewport (e.g., with the "back" functionality).

Guideline 5. Observe operating system conventions and standard interfaces

Communicate with other software (assistive technologies, the operating system, plug-ins) through applicable interfaces and observe conventions for the user interface, documentation, installation, etc.

To promote interoperability, user agents should adopt operating system conventions and standard APIs for communication, user interface design, documentation , etc. Following operating system conventions and supporting standard APIs will promote predictability for users as well as assistive technologies that rely on information from other software.

Some operating systems have operating system-level flags and settings that are pertinent to accessibility, such as high-contrast colors and "show" sounds for people with hearing impairments. User agents should take these global settings into account for their own settings.

5.1 Provide accessible APIs to other technologies. [Priority 1]
5.2 Use accessibility resources and conventions of the operating system and supported programming languages, including those for plug-ins and virtual machine environments. [Priority 1]

For instance, if the user agent supports Java applets and provides a Java Virtual Machine to run them, the user agent should support the proper loading and operation of a Java native assistive technology. This assistive technology can provide access to the applet as defined by Java accessibility standards.

5.3 Provide programmatic read and write access to user agent functionalities and user interface controls. [Priority 1]

For example, ensure that assistive technologies have access to information about the current input configuration so that they can trigger functionalities through keyboard events, mouse events, etc. Refer also to checkpoint 5.2.

5.4 Implement selection and focus mechanisms and make the selection and focus available to users and through APIs. [Priority 1]

Refer also to checkpoint 7.1 and checkpoint 5.3.

5.5 Provide programmatic notification of changes to content and user interface controls (including selection and focus). [Priority 1]

Refer also to checkpoint 5.2.

5.6 Conform to W3C Document Object Model specifications and export interfaces defined by those specifications. [Priority 1]

For example, refer to [DOM1] and [DOM2] . User agents should export these interfaces using available operating system conventions. **Note.** The DOM Level 1 specification states that "DOM applications may provide additional interfaces and objects not found in this specification and still be considered DOM compliant."

5.7 Provide programmatic exchange of information in a timely manner. [Priority 2]

This is important for synchronous alternative renderings and simulation of events.

5.8 Follow operating system conventions and accessibility settings. In particular, follow conventions for user interface design, default keyboard configuration, product installation, and documentation . [Priority 2]

Refer also to checkpoint 10.5.

Guideline 6. Implement open specifications and their accessibility features

In particular, implement W3C specifications when they are appropriate for a task and follow accessibility guidelines for those specifications.

W3C specifications promote interoperability, which improves accessibility through predictability and openness. The current guidelines also recommend implementing W3C specifications (e.g., [HTML32] , [HTML40] , [CSS1] , [CSS2] , [MATHML] , [SMIL] , etc.) for the following reasons:

- W3C specifications include "built-in" accessibility features.
- W3C specifications undergo early review to ensure that accessibility issues are considered during the design phase.
- W3C specifications are developed in an open, industry consensus process.

6.1 Implement the accessibility features of supported specifications (markup languages, style sheet languages, metadata languages, graphics formats, etc.). [Priority 1]

Note. The Techniques Document ([UA-TECHNIQUES]) discusses accessibility features of W3C specifications.

6.2 Conform to W3C specifications when they are appropriate for a task. [Priority 2]

For instance, for markup, implement [HTML40] or [XML] . For style sheets, implement [CSS1] , [CSS2] , or XSL. For mathematics, implement [MATHML] . For synchronized multimedia, implement [SMIL] . For access to the structure of HTML or XML documents, implement [DOM1] . For an event model, implement [DOM2] . Refer also to checkpoint 5.6.

Guideline 7. Provide navigation mechanisms

Provide navigation mechanisms that meet the needs of different users: serial navigation for context, direct navigation for speed, search functions, structured navigation, etc.

Navigation mechanisms help all users find the information they seek. User agents should provide a variety of mechanisms - from simple scrolling through content to search mechanisms to serial ("tabbing") navigation - to meet the diverse needs of users, in particular users of devices that render content serially (e.g., synthesized speech output or single-line refreshable braille displays). So that users of serial devices are not required to view an entire page or presentation to find information, user agents should provide more direct navigation mechanisms.

Authors are encouraged to include navigation mechanisms (e.g., image maps or navigation bars) designed for their content, but user agents should provide generic mechanisms, some of which are described here:

- Sequential access (e.g., line scrolling, page scrolling, tabbing access through active elements, etc.) means advancing through rendered in well-defined steps (line by line, screen by screen, link by link, etc.) forward and backward. Sequential access provides context, but can be slow. This navigation technique benefits users who cannot scan a page for context and any user unfamiliar with it. Sequential access may be based on element type (e.g., links only), content structure (e.g., navigate from header to header), or other criteria. Structured navigation mechanisms allow users to move rapidly through highly structured content such as books or instructional material.
- Direct access (go to a particular link or paragraph, search for instances of this string, etc.) is faster than sequential access, but context may be lost. Direct access benefits users with some motor impairments and power users familiar

with a document. Searching on text is one important variant of direct access, but other types of direct access are possible (e.g., navigation to a link based on its position in content). Selecting text or structured content with the pointing device is another form of direct access.

User agents should allow users to configure navigation mechanisms (e.g., to allow navigation of links only, or links and headers, or tables and forms, etc.). Refer also to guideline 10..

Note. For all search and navigation functions, the user agent should follow operating system conventions for using selection and focus mechanisms. For instance, the selection should be used to identify the results of a text search, the focus should identify active elements during sequential navigation of active elements, etc.

7.1 Allow the user to navigate viewports (including frames). [Priority 1]

Note. For example, when all frames of a frameset are displayed side-by-side, allow the user to navigate among them with the keyboard. Or, when frames are displayed individually (e.g., by a text browser or speech synthesizer), provide a list of links to individual frames. Navigating into a viewport makes it the current viewport.

7.2 For user agents that offer a browsing history mechanism, when the user returns to a previous view, restore the point of regard in the viewport . [Priority 1]

For example, when users navigate "back" and "forth" among views, for each view they should find the viewport position where they left it.

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

Note. Navigation techniques include keyboard navigation from cell to cell (e.g., using the arrow keys) and page up/down scrolling. Refer also to checkpoint 1.1 and checkpoint 5.3.

7.4 Allow the user to navigate all active elements . [Priority 1]

Navigation mechanisms may range from sequential (e.g., serial navigation by tabbing) to direct (e.g., by entering link text) to searching on active elements only (e.g., based on form control text, associated labels, or form control names).

7.5 Allow the user to navigate just among all active elements . [Priority 2]

Refer also to checkpoint 7.4.

7.6 Allow the user to search for rendered text content, including alternative text content. [Priority 2]

7.7 Allow the user to navigate according to structure. [Priority 2]

For example, allow the user to navigate familiar elements of a document: paragraphs, tables, headers, lists, etc.

7.8 Allow the user to configure structured navigation. [Priority 3]

For example, allow the user to navigate only paragraphs, or only headers and paragraphs, etc.

Guideline 8. Help orient the user

Provide information about resource structure, viewport structure, element summaries, etc. that will assist the user understand their browsing context.

All users require clues to help them understand their "location" when browsing. Graphical user agents provide clues such as proportional scroll bars to indicate how much content has been viewed. A highlighted selection or focus (either visually or aurally) distinguishes the selected or focused content from other content. User agent history allows users to track and undo their browsing path. HTML 4.0 ([HTML40] , section 11.2.3) allows authors to create table headers and footers (with THEAD and TBODY) so that user agents can scroll table content while keeping table head and foot visible on the screen.

Orientation mechanisms such as these are especially important to users who view content through serial means such speech or braille (current tactile technology is limited in the amount of information that can be displayed). Users of graphically displayed tables can scan a table quickly to understand the position (and related header information) of a particular cell. For users of serial output, user agents should provide this context on demand. Similarly, users need to know about:

- relationships between frames (e.g., how changes in one frame affect another). Refer also to checkpoint 9.1.
- link context. Users who browse by navigating links only require information about links that will allow them to decide whether to follow the link.
- form context. Users need to know when they’ve provided all necessary information in a form before submitting it.

For people with visual impairments, blindness, or certain types of learning disabilities, it is important that the point of regard remain as stable as possible. 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.

User agents must make orientation information available in an output device-independent manner. Refer also to guideline 1.

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

Note. This includes highlighting and identifying frames. Refer also to checkpoint 9.1..

8.2 Convey the author-specified purpose of each table and the relationships among the table cells and headers. [Priority 1]

For example, provide information about table headers, how headers relate to cells, table caption and summary information, cell position information, table dimensions, etc. **Note.** This checkpoint is an important special case of checkpoint 2.1.

8.3 Provide an outline of a resource view built from its structural elements (e.g., frames, headers, lists, forms, tables, etc.) [Priority 2]

For example, for each frame in a frameset, provide a table of contents composed of headers where each entry in the table of contents links to the header in the document.

8.4 Indicate whether a focused link has been marked up to indicate that following it will involve a fee. [Priority 2]

Note. [MICROPAYMENT] describes how authors may mark up micropayment information in an interoperable manner. This information may be provided through the standard user interface provided the interface is accessible. Thus, any prompt asking the user to confirm payment must be accessible.

8.5 Provide information to help the user decide whether to follow a focused link. [Priority 2]

Note. Useful information includes: whether the link has already been visited, whether it designates an internal anchor, the type of the target resource, the length of an audio or video clip that will be started, and the expected natural language of target resource.

8.6 Allow the user to configure the outline view. [Priority 3]

For example, allow the user to control the level of detail of the outline. Refer also to checkpoint 8.3. Refer also to checkpoint 5.3.

8.7 Allow the user to configure what information about links to present. [Priority 3]

Note. Using color as the only distinguishing factor between visited and unvisited links does not suffice since color may not be perceivable by all users or rendered by all devices. Refer also to checkpoint 8.5.

8.8 Provide a mechanism for highlighting and identifying (through a standard interface where available) active elements. [Priority 3]

Note. User agents may satisfy this checkpoint by implementing the appropriate style sheet mechanisms, such as link highlighting.

8.9 Maintain consistent user agent behavior and default configurations between software releases. Consistency is less important than accessibility and adoption of operating system conventions. [Priority 3]

In particular, make changes conservatively to the layout of user interface controls, behavior of existing functionalities, and default keyboard configuration.

Guideline 9. Notify the user of content and viewport changes

Alert users, in an output device-independent fashion, of changes to content or the viewport.

Changes to content or browsing context (How many viewports are open? Which is the current one?) may disorient users with visual impairments or certain types of learning disabilities. User agents should provide information about changes caused by scripts, or allow users to turn off scripts entirely (refer to checkpoint 3.7).

User agents must ensure that notifications are available in an output device-independent manner. Refer also to guideline 1.

9.1 Provide information about user agent-initiated content and viewport changes directly to the user and through APIs. [Priority 1]

For example, inform the users when a script causes a popup menu to appear.

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

9.3 Prompt the user to confirm any form submission triggered indirectly, that is by any means other than the user activating an explicit form submit control. [Priority 2]

For example, do not submit a form automatically when a menu option is selected, when all fields of a form have been filled out, on a mouseover event, etc.

9.4 Allow the user to configure notification preferences for common types of content and viewport changes. [Priority 3]

For example, allow the user to choose to be notified (or not) that a script has been executed, that a new viewport has been opened, that a pulldown menu has been opened, that a new frame has received focus, etc.

9.5 When loading content (e.g., document, video clip, audio clip, etc.) indicate what portion of the content has loaded and whether loading has stalled. [Priority 3]

9.6 Indicate the relative position of the viewport in content (e.g., the percentage of an audio or video clip that has been played, the percentage of a Web page that has been viewed, etc.). [Priority 3]

Note. Depending on how the user has been browsing, the percentage may be calculated according to focus position, selection position, or viewport position.

Guideline 10. Allow the user to configure the user agent

Allow users to configure rendering, mouse, keyboard, the user interface, etc. to facilitate daily use of the software.

Web users have a wide range of functional capabilities and so they must be able to configure the user agent to meet their particular requirements.

Refer also to checkpoint 8.9.

10.1 Provide information about the current user-specified input configuration (e.g., keyboard or voice bindings specified through the user agent's user interface). [Priority 1]

10.2 Provide information about the current author-specified input configuration (e.g., keyboard bindings specified in content such as by "accesskey" in [HTML40]). [Priority 2]

10.3 Allow the user to change and control the input configuration . Users should be able to activate a functionality with a single-stroke (e.g., single-key, single voice command, etc.). [Priority 2]

Users should not be required to activate functionalities by navigating through the graphical user interface (e.g., by moving a mouse to activate a button or by pressing the "down arrow" key repeatedly in order to reach the desired activation mechanism. Input configurations should allow quick and direct access that does not rely on graphical output. For self-voicing browsers, allow the user

to modify what voice commands activate functionalities. Similarly, allow the user to modify the graphical user interface for quick access to commonly used functionalities (e.g., through buttons).

10.4 Use operating system conventions to indicate the input configuration . [Priority 2]

For example, on some operating systems, if a functionality is available from a menu, the letter of the key that will activate that functionality is underlined.

10.5 Avoid default input configurations that interfere with operating system conventions. [Priority 2]

For example, the default configuration should not include "Alt-F4" or "Control-Alt-Delete" on operating systems where that combination has special meaning to the operating system. In particular, default configurations should not interfere with the mobility access keyboard modifiers reserved for the operating system. Refer also to guideline 5.

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

Users must be able to select from among available profiles or no profile (i.e., the user agent default settings).

10.7 Provide default input configurations for frequently performed operations. [Priority 3]

Make it easy to use the most frequently requested commands. In particular, provide convenient mappings to functionalities that promote accessibility such as navigation of links.

10.8 Allow the user to configure the graphical arrangement of user interface controls. [Priority 3]

Guideline 11. Provide accessible product documentation and help

Ensure that the user can learn about software features, notably those that relate to accessibility.

Documentation Documentation includes anything that explains how to install, get help for, use, or configure the product. Users must have access to installation information, either in electronic form (CD-ROM, diskette, over the Web), by fax, or by telephone.

Some people cannot use printed documents. Vendors should provide accessible electronic documentation for users with visual impairments, learning disabilities, or movement impairments. Alternative hardcopy formats may also benefit some users.

Since users who are not disabled are generally unaware of software features designed specifically for accessibility, those features should be clearly documented. This will allow users with disabilities to learn about the software more easily.

Refer also to checkpoint 5.8.

11.1 Provide a version of the product documentation that conforms to the Web Content Accessibility Guidelines. [Priority 1]

User agents may provide documentation in many formats, but one must be accessible as per [WAI-WEBCONTENT] . Alternative content, navigation mechanisms, and illustrations will all help make the documentation accessible.

11.2 Document all user agent features that promote accessibility. [Priority 1]

For example, review the documentation or help system to ensure that it discusses the functionalities addressed by the checkpoints of this document.

11.3 Document the default input configuration (e.g., default keyboard bindings). [Priority 1]

For example, documentation of what user agent features may be activated with a single keystoke, voice command, or button activation is an important part of the user interface to users visual impairments, some types of movement impairments, or multiple disabilities. Without this documentation,these users may not realize they can accomplish a particular task with a single gesture and so might unnecessarily avoid that feature of the software. Or they might waste time and energy using a very inefficient technique to perform a task.

11.4 In a dedicated section, document all features of the user agent that promote accessibility. [Priority 2]

---

3. Appendix: Terms and Definitions

Active element

Active elements constitute a user interface for the document. They have associated behaviors that may be ***activated*** (or "triggered") either through user interaction or through scripts. Which elements are active depends on the document language and whether the features are supported by the user agent. In HTML documents, for example, active elements include links, image maps, form controls, element instances with a value for the "longdesc" attribute, and element instances with associated scripts (event handlers) explicitly associated with them (e.g., through the various "on" attributes).

An active element's behavior may be triggered through any number of mechanisms, including the mouse, keyboard, an API, etc. The effect of activation depends on the element. For instance, when a link is activated, the user agent generally retrieves the linked resource. 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.

Most systems use the focus to designate the active element the user wishes to trigger.

Applicable checkpoint

If a user agent offers a functionality, it must ensure that all users have access to that functionality or an equivalent alternative. Thus, if the user agent supports keyboard input, it must support accessible keyboard input. If the user agent supports images, it must ensure access to each image or an alternative equivalent supplied by the author. If a user agent supports style sheets, it must implement the accessibility features of the style sheet language. If the user agent supports frames, it must ensure access to frame alternatives supplied by the author.

Not all user agents support every content type, markup language feature, input or output device interface, etc. When a content type, feature, or device interface is not supported, checkpoints with requirements related to it do not apply to the user agent. Thus, if a user agent supports style sheets at all, all checkpoints related to style sheet accessibility apply. If a user agent does not support style sheets at all, the checkpoints do not apply.

The applicability of checkpoints related to markup language features is measured similarly. If a user agent supports tables, it must support the accessibility features of the language related to tables (or images, or frames, or video, or links, etc.). The Techniques Document includes information about the accessibility features of W3C languages such as HTML, CSS, and SMIL.

The following summarizes criteria for applicability. A checkpoint applies to a user agent unless:

- The checkpoint definition states explicitly that it only applies to a different class of user agent.
- The checkpoint includes requirements about a content type (script, image,

video, sound, applets, etc.) that the user agent does not recognize at all.
- The checkpoint includes requirements about a content type that the user agent recognizes but does not support natively .
- The checkpoint refers to the properties of an embedded object (e.g., video or animation rate) that may not be controlled or accessed by the user agent.
- The checkpoint includes requirements about an unsupported markup language or other technology (e.g., style sheets, mathematical markup language, synchronized multimedia, metadata description language, etc.)
- The checkpoint refers to an unsupported input or output device interface. Note that if the interface is supported at all, it must be supported accessibly.

Assistive Technology

Software or hardware that has been specifically designed to assist people with disabilities in carrying out daily activities. Assistive technology includes wheelchairs, reading machines, devices for grasping, alternative computer keyboards or pointing devices, etc. In the area of Web Accessibility, common software-based assistive technologies include assistive technologies, which rely on other user agents for input and/or output. These include:

- screen magnifiers, which are used by people with visual impairment to enlarge and change colors on the screen to improve readability of text and images.
- screen readers, which are used by people who are blind or with reading disabilities to read textual information through speech or braille displays.
- alternative keyboards, which are used by people with movement impairments to simulate the keyboard.
- alternative pointing devices, which are used by people with movement impairments to simulate mouse pointing and button activations.

Configure

To set user preferences. This may be done through the user agent's user interface, through configuration files, by scripts, etc.

Continuous Equivalent Track

A continuous equivalent track presents an equivalent alternative to another track (generally audio or video) and is synchronized with that track. Continuous equivalent tracks convey information about spoken words and non-spoken sounds such as sound effects. A continuous text track presents ***closed captions***. Captions are generally rendered visually by being superimposed over a video track, which benefits people who are deaf and hard-of-hearing, and anyone who cannot hear the audio (e.g., when in a crowded room). A ***collated text transcript*** combines (collates) captions with text descriptions of video information (descriptions of the actions, body language, graphics, and scene changes of the video track). These text equivalents make presentations accessible to people who are deaf-blind and to people who cannot play movies, animations, etc.

One example of a non-text continuous equivalent track is an ***auditory description*** of the key visual elements of a presentation. The description is either a prerecorded human voice or a synthesized voice (recorded or

generated on the fly). The auditory description is synchronized with the audio track of the presentation, usually during natural pauses in the audio track. Auditory descriptions include information about actions, body language, graphics, and scene changes.

A video track that shows sign language is another example of a continuous equivalent track.

Control

User control of the user agent - interface, behavior, styles, etc. - means that the user can choose preferred behavior from a set of options. For instance, control of colors means that the user can choose from available colors, within the limits offered by the operating system or user agent.

The term "control" also means "user interface component" or "form component" in this document. Which meaning is intended should be apparent from context.

Device Independence

The ability to make use of software via any input or output device API provided by the operating system and used by the user agent. User agents should follow operating system conventions and use standard APIs for device input and output.

Documentation

Documentation includes **all** product documentation, notably installation instructions, the help system, and all product manuals.

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. An element's ***rendered content*** is that which a user agent renders for the element. This may be what lies between the element's start and end tags, the value of an attribute (c.f. the "alt", "title", and "longdesc" attributes in HTML), or external data (e.g., the IMG element in HTML). Rendering is not limited to graphical displays alone, but also includes audio (speech and sound) and tactile displays (braille and haptic displays).

Since rendered content is not always accessible, authors must specify ***alternative equivalents for content*** that user agents must make available to users or software that require it (in place of and/or in addition to the "primary" content). Alternative representations may take a variety of forms including alternative text, closed captions, and auditory descriptions . The Techniques Document ([UA-TECHNIQUES]) describes the different mechanisms authors use to supply alternative representations of content. Please also consult the Web Content Accessibility Guidelines ([WAI-WEBCONTENT] and ([WAI-WEBCONTENT-TECHS] .

Events and scripting

When certain events occur (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] 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.

Focus

The user focus designates an active element in a document. A viewport has at most one focus. When several viewports co-exist, each may have a focus, but only one is active, called the ***current focus***. The current focus is generally presented (e.g., highlighted) in a way that makes it stand out.

Highlight

Any mechanism used to emphasize selected or focused content. Visual highlight mechanisms include dotted boxes, underlining, and reverse video. Synthesized speech highlight mechanisms may include altering voice pitch or volume.

Input Configuration

Every user agent functionality available to the user is mapped to some user interface mechanism, including menus, buttons, keyboard shortcuts, voice commands. The default input configuration is the mapping the user finds after installation of the software. The documentation should tell users what functionalities are available and the user interface should remind users of the current mapping to the user interface and allow them to figure out quickly how to use the appropriate software features.

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 viewport has at most one insertion point. When several viewports 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).

Native support

A user agent supports a feature natively if it does not require another piece of software (e.g., plug-in or external program) for support. Native support does not preclude more extensive support for accessibility by assistive technologies , so user agents must still make information available through APIs.

Natural Language

Spoken, written, or signed human languages such as French, Japanese, American Sign Language, and braille. The natural language of content may be indicated in markup (e.g., by the "lang" attribute in HTML ([HTML40] , section 8.1) or by HTTP headers.

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 [CSS2] .) A reference to "styles" in this document means a set of style-related properties.

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***.

Recognize

A user agent is said to recognize markup, content types, or rendering effects when it can identify (through built-in mechanisms, DTDs, style sheets, headers, etc) the information. For instance, HTML 3.2 user agents may not recognize the new elements or attributes of HTML 4.0. Similarly, a user agent may recognize blinking content specified by elements or attributes, but may not recognize that an applet is blinking. The Techniques Document ([UA-TECHNIQUES]) discusses some content that affects accessibility and should be recognized as such.

Selection

The user selection generally specifies a range of content (text, images, etc.) in a document. The selection may be structured (based on the document tree) or unstructured (e.g., text-based). Content may be selected through user interaction, scripts, etc. 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 viewport has at most one user selection. When several viewports co-exist, each may have a user selection, but only one is active, called the ***current user selection***.

The user selection is usually presented in a way the stands out (e.g., highlighted). On the screen, the selection may be highlighted using colors, fonts, graphics, or other mechanisms. Highlighted text is often used by assistive technologies to indicate through speech or braille output what the user wants to read. Most screen readers are sensitive to highlight colors. Assistive technologies may

provide alternative presentation of the selection through speech, enlargement, or refreshable braille display.

Text transcript

A text transcript is a text equivalent of audio information that includes spoken words and non-spoken sounds such as sound effects. Refer also to continuous equivalent track .

Spawned Viewport

Viewports that are created by the user agent process. This refers to viewports that display content and does not include, for example, messages or prompts to the user.

Standard Device APIs

Operating systems are designed to be used by default with devices such as pointing devices, keyboards, voice input, etc. The operating system (or windowing system) provides standard APIs for these devices that should be used by user agents and other software for input and output to those devices. For example, for desktop computers today, user agents are expected to use the mouse and keyboard APIs for input. For touch screen devices or mobile devices, standard input APIs may include stylus, buttons, voice, etc. The display and sound card are considered standard ouput devices for a graphical desktop computer environment and each has a standard API. **Note.** What is considered "standard" for a particular environment will change over time.

User-initiated and User Agent-initiated

User-initiated actions result from user input to the user agent. User Agent-initiated actions result from scripts, operating system conditions, or built-in user agent behavior.

User Agent

A user agent is an application that retrieves and renders Web resources, including text, graphics, sounds, video, images, and other objects. An user agent may require additional software to handle some types of content. For instance, a browser may run a separate program or plug-in to render sound or video. The additional software is also considered a user agent. User agents include graphical desktop browsers, multimedia players, text browsers, voice browsers, and assistive technologies such as screen readers, screen magnifiers, speech synthesizers, onscreen keyboards, and voice input software.

Views, Viewports, and Point of Regard

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 panner, 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 content in a variety of ways; each rendering is called a ***view***. For instance, a user agent may allow users to view an entire document or just a list of the document’s headers. These are two different views of the document.

The view is *how* source information is rendered and the viewport is *where* it is rendered. Both the current focus and the current user selection must be in the

same viewport, called the ***current viewport***. The current viewport is generally highlighted when several viewports co-exist.
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 graphical browser viewport generally features scrollbars or some other paging mechanism that allows the user to bring the rendered content into the viewport.
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.

---

4. Acknowledgments

Many thanks to the following people who have contributed through review and comment: Paul Adelson, James Allan, Denis Anson, Kitch Barnicle, Harvey Bingham, Olivier Borius, Judy Brewer, Bryan Campbell, Kevin Carey, Wendy Chisholm, David Clark, Chetz Colwell, Wilson Craig, Nir Dagan, Daniel Dardailler, B. K. Delong, Neal Ewers, Geoff Freed, John Gardner, Al Gilman, Larry Goldberg, Glen Gordon, John Grotting, Markku Hakkinen, Earle Harrison, Chris Hasser, Kathy Hewitt, Philipp Hoschka, Masayasu Ishikawa, Phill Jenkins, Jan Kärrman (for help with html2ps), Leonard Kasday, George Kerscher, Marja-Riitta Koivunen, Josh Krieger, Catherine Laws, Greg Lowney, Scott Luebking, William Loughborough, Napoleon Maou, Charles McCathieNevile, Karen Moses, Masafumi Nakane, Mark Novak, Charles Oppermann, Mike Paciello, David Pawson, Michael Pederson, Helen Petrie, David Poehlman, Michael Pieper, Jan Richards, Hans Riesebos, Joe Roeder, Lakespur L. Roca, Gregory Rosmaita, Lloyd Rutledge, Liam Quinn, T.V. Raman, Robert Savellis, Rich Schwerdtfeger, Constantine Stephanidis, Jim Thatcher, Jutta Treviranus, Claus Thogersen, Steve Tyler, Gregg Vanderheiden, Jaap van Lelieveld, Jon S. von Tetzchner, Willie Walker, Ben Weiss, Evan Wies, Chris Wilson, Henk Wittingen, and Tom Wlodkowski,

5. References

For the latest version of any W3C specification, please consult the list of W3C Technical Reports.

[CSS1]
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. This CSS1 Recommendation is http://www.w3.org/TR/1999/REC-CSS1-19990111.

[CSS2]
"CSS, level 2 Recommendation", B. Bos, H. Wium Lie, C. Lilley, and I. Jacobs, eds., 12 May 1998. This CSS2 Recommendation is http://www.w3.org/TR/1998/REC-CSS2-19980512.

[CSS-ACCESS]
"Accessibility Features of CSS", I. Jacobs, J. Brewer, The latest version of this W3C Note is available at http://www.w3.org/TR/CSS-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 1 October 1998 DOM Level 1 Recommendation is http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001

[DOM2]
"Document Object Model (DOM) Level 2 Specification", L. Wood, A. Le Hors, V. Apparao, L. Cable, M. Champion, J. Kesselman, P. Le Hégaret, T. Pixley, J. Robie, P. Sharpe, C. Wilson, eds. The DOM2 specification is a Working Draft at the time of publication.

[HTML40]
"HTML 4.0 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds. The 24 April 1998 HTML 4.0 Recommendation is http://www.w3.org/TR/1998/REC-html40-19980424

[HTML32]
"HTML 3.2 Recommendation", D. Raggett, ed. The HTML 3.2 Recommendation is http://www.w3.org/TR/REC-html32

[MATHML]
"Mathematical Markup Language", P. Ion and R. Miner, eds. The 7 April 1998 MathML 1.0 Recommendation is http://www.w3.org/TR/1998/REC-MathML-19980407

[MICROPAYMENT]
"Common Markup for micropayment per-fee-links", T. Michel, ed. The latest version of this W3C Working Draft is available at http://www.w3.org/TR/Micropayment-Markup.

[RFC2119]
"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997.

[SMIL]
"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", P.

Hoschka, editor. The 15 June 1998 SMIL 1.0 Recommendation is http://www.w3.org/TR/1998/REC-smil-19980615

[SMIL-ACCESS]

"Accessibility Features of SMIL", M-R. Koivunen, I. Jacobs. The latest version of this W3C Note is available at http://www.w3.org/TR/SMIL-access.

[UA-CHECKLIST]

An appendix to this document lists all of the checkpoints, sorted by priority. The checklist is available in either tabular form (at http://www.w3.org/TR/1999/WD-WAI-USERAGENT-19991105/full-checklist) or list form (at http://www.w3.org/TR/1999/WD-WAI-USERAGENT-19991105/checkpoint-list).

[UA-TECHNIQUES]

"Techniques for User Agent Accessibility Guidelines 1.0", J. Gunderson, I. Jacobs, eds. This document explains how to implement the checkpoints defined in "User Agent Accessibility Guidelines 1.0". The draft of the Techniques Document available at the time of this document's publication is http://www.w3.org/TR/1999/WD-WAI-USERAGENT-TECHS-19991105. The latest draft of the techniques is available at http://www.w3.org/TR/WAI-USERAGENT-TECHS/

[WAI-AUTOOLS]

"Authoring Tool Accessibility Guidelines", J. Treviranus, J. Richards, I. Jacobs, C. McCathieNevile, eds. The latest Working Draft of these guidelines for designing accessible authoring tools is available at http://www.w3.org/TR/WD-WAI-AUTOOLS/

[WAI-WEBCONTENT]

"Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. The 5 May 1999 Recommendation is http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505

[WAI-WEBCONTENT-TECHS]

"Techniques for Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. The latest version of this document is available at http://www.w3.org/TR/WAI-WEBCONTENT-TECHS

[XML]

"Extensible Markup Language (XML) 1.0.", T. Bray, J. Paoli, C.M. Sperberg-McQueen, eds. The 10 February 1998 XML 1.0 Recommendation is http://www.w3.org/TR/1998/REC-xml-19980210