User Agent Accessibility Guidelines 1.0

W3C Working Draft 4-October-1999

This version:
http://www.w3.org/WAI/UA/WAI-USERAGENT-19991004
(plain text, postscript, pdf, gzip tar file of HTML, zip archive of HTML)
Latest version:
http://www.w3.org/WAI/UA/WAI-USERAGENT
Previous version:
http://www.w3.org/WAI/UA/WAI-USERAGENT-19990827

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

Copyright © 1999 W3C® (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

Abstract

This document provides guidelines to user agent developers for making their products more accessible to people with disabilities and for increasing usability for all users. The guidelines in this document apply to user agents in general, including graphical, text, and voice browsers, assistive technologies [p. 24] (screen readers, screen magnifiers, braille displays, and voice input software) that depend on them, multimedia players, and plug-ins.

This document includes an appendix that organizes all of the checkpoints [p. 7] by topic and priority. The checkpoints in the appendix link to their definitions in the current document. The topics identified in the appendix include user interface, keyboard support, images, multimedia, tables, frames, forms, and scripts. This appendix is available both as a tabular summary and as a simple list of checkpoints.

A separate document, entitled "Techniques for User Agent Accessibility Guidelines 1.0" ([UA-TECHNIQUES] [p. 32]), explains how to implement the checkpoints defined in the current document. The Techniques Document discusses each checkpoint in more detail and provides examples using the Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), Synchronized Multimedia Integration Language (SMIL), and the Mathematical Markup Language (MathML). The Techniques Document has been designed to track changes in technology 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. The series also includes Web Content Accessibility Guidelines ([WAI-WEBCONTENT] [p. 32]) and Authoring Tool Accessibility Guidelines ([WAI-AUTOOLS] [p. 32]).

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 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.

In this version of the document, the conformance distinction between "graphical desktop user agents" and "dependent user agents" was removed. However, the Working Group has not yet decided how to modify those checkpoints that were designated "for dependent user agents only."

Please send comments about this document to the public mailing list: w3c-wai-ua@w3.org.

This document has been produced as part of the Web Accessibility Initiative, 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 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

1. Introduction 4
    1.1 Principles of Accessible Design 4
2. How the Guidelines are Organized 7
    2.1 Document conventions 7
3. Priorities 7
    3.1 Conformance 8
4. User Agent Accessibility Guidelines 10
    1. Support input and output device-independence 10
    2. Ensure keyboard access to user agent functionalities 11
    3. Ensure user access to document content 12
    4. Allow the user to turn off features that may reduce accessibility . 14
    5. Ensure user control over document styles 15
    6. Observe system conventions and standard interfaces 17
    7. Support applicable W3C technologies and guidelines 17
    8. Provide navigation mechanisms 18
    9. Help orient the user 20
    10. Notify the user of document and viewport changes 22
    11. Allow the user to configure the user agent 23
    12. Provide accessible product documentation and help 23
Appendix A. -- Glossary 24
5. Acknowledgments 30
6. References 31

The appendix list of checkpoints is available as either a tabular summary of checkpoints or as a simple list of checkpoints.

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 the document is written.
- 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.).

The guidelines in this document are designed to help developers understand barriers to accessibility and encourage solutions that broaden accessibility to the Web. The associated Techniques Document ([UA-TECHNIQUES] [p. 32]) 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.

A user agent is an application that retrieves and renders Web resources, including text, graphics, sounds, video, images, and other objects. An application 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 and is therefore covered by these guidelines. User agents include graphical desktop browsers, multimedia players, text browsers, voice browsers, and dependent user agents [p. 25] such as screen readers, screen magnifiers, speech synthesizers, onscreen keyboards, and voice input software.

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. This includes the functionalities built into the tool (made available through menus, dialogs, toolbars and other user interface components) as well as those offered via the document (made available through links, form controls, applets, and other interactive elements).

User agents should provide serial access to functionalities (e.g., through cascading menus, through help systems, etc.) to provide context to users with cognitive impairments and to any users unfamiliar with the tool. For users with physical impairments (e.g., motor limitations) and for power users, user agents should provide direct access to functionalities in an input device-independent manner: through the mouse, keyboard shortcuts, voice commands, etc.

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 includes some references to general software accessibility guidelines.

Ensure that the user has access to content.

Through the user interface, users must have access to document content, including alternative content [p. 26] (e.g., "alt" attribute text in HTML). To ensure access to content, users must have final control of colors, fonts, synthesized speech rate and volume, and other rendering parameters. User agents should also make content available to other software through standard application programming interfaces (APIs). This allows, for example, assistive technologies to render content according to the functional requirements of some users with disabilities.

User agents may facilitate access to content by providing a convenient user interface (e.g., scrolled viewport) and navigation mechanisms (e.g., allowing users to tab to active elements [p. 26] of a document).

Help orient the user.

Orientation - the sense of where one is within a document, series of documents, video clip, etc. - is fundamental to a successful Web experience. Graphical mechanisms provided by the author that promote orientation include:

- frames, which suggest content relationships
- graphical site maps, which describe page organization

Graphical mechanisms provided by the user agent that promote orientation include:

- proportional scroll bars, which indicate how much of a resource has been viewed
- a visually highlighted [p. 27] selection

Some users cannot use graphical orientation clues and require other mechanisms to remain oriented:

- Contextual information about document elements. For instance, users with blindness who navigate by surfing only the links of a document benefit from knowing how many links the document contains or the number of the current link. Numerical position information, in conjunction with a navigation mechanism that allows users to jump directly to a given link, can greatly facilitate browsing, especially for documents with many links.
- Contextual information about the document, viewports, multimedia objects, and other resources (e.g., how many viewports, whether the document or video clip has finished loading, how much of an audio clip has played, etc.) For instance, information about the number of frames and their names is very helpful, as well as links to navigate among them.
- Summary information about specific elements (e.g., the dimensions of a table).
- Alerts when events occur during browsing (e.g., a window opens, a script is executed, etc.).

The user agent should also 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.

Follow system standards and conventions.

Following system standards and conventions promotes accessibility in a number of ways:

- Observing system conventions in user interface design, software installation, and software documentation improves usability.
- Using standard system interfaces makes it possible for assistive technologies [p. 24] 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 information available to other software. Scripting tools and automated test engines also benefit from having access to user agent information through standard interfaces.

Implementing interoperable specifications (such as those produced by W3C) promotes accessibility for several reasons:

- W3C technologies 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.

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.
- Guideline navigation links. Three links allow navigation to the next guideline (right arrow icon), the previous guideline (left arrow icon), or the current guideline's position in the table of contents (up arrow icon).
- 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 relevant entry in the "Checkpoint Map" of the Techniques Document (refer to [UA-TECHNIQUES] [p. 32]). The Checkpoint Map lists each checkpoint and links to the sections of the Techniques Document 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.

2.1 Document conventions

The following editorial conventions are used throughout this document:

- Element names are in uppercase letters.
- Attribute names are quoted in lowercase letters.
- Links to definitions are highlighted through the use of style sheets.

3. 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 remove significant barriers to accessing Web documents.

[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.

3.1 Conformance

The terms "must", "should", and "may" (and related terms) are used in this document in accordance with RFC 2119 ([RFC2119] [p. 31]).

User agents must satisfy natively [p. 27] all the applicable [p. 24] checkpoints for a chosen conformance level [p. 8] .

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/WAI/UA/WAI-USERAGENT-19991004
- The conformance level satisfied: "A", "Double-A", or "Triple-A".

Example of Form 1:

This product conforms to W3C's "User Agent Accessibility Guidelines 1.0", available at http://www.w3.org/WAI/UA/WAI-USERAGENT-19991004, 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.*

4. User Agent Accessibility Guidelines

Guideline 1. Support input and output device-independence

Ensure that the user can interact with the user agent (and the document it renders) through standard input and output APIs supported by the operating system.

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 devices supported by the underlying system.

The best way to make this possible is to design software that follows 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 keyboard can still use special devices or an on-screen keyboard to operate the user agent.

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 the most general output media, 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 (and to develop new ones), including pointing devices, keyboards, braille devices, head wands, microphones, touch screens, speech synthesizers, and more.

Please refer also to guideline 6, which discusses the importance to accessibility of following operating system conventions.

1.1 Ensure that all functionalities offered through the user interface may be operated through standard input device APIs supported by the operating system. [Priority 1]
    This will allow users to operate the user agent in an input device independent manner.
1.2 Ensure that the user can interact with all active elements [p. 26] of a document in a device independent [p. 25] manner. [Priority 1]
    For example, ensure that the user can activate links of a client-side image map in a device-independent manner (e.g., by making them available as text links).
1.3 Ensure that the user can install the user agent software in a device independent [p. 25] manner. [Priority 1]
1.4 Ensure that the user can configure [p. 25] the user agent in a device independent [p. 25] manner. [Priority 1]

1.5 Ensure that the user can access user agent documentation in a device independent [p. 25] manner. [Priority 1]
1.6 Ensure that all messages to the user (e.g., warnings, errors, etc.) are available through standard output device APIs supported by the operating system. [Priority 1]

For instance, ensure that information about how much of a document or video clip has been viewed is available through output device APIs. Proportional navigation bars may provide this information visually, but the information must be available to users relying on synthesized speech or braille output.

Guideline 2. Ensure keyboard access to user agent functionalities

Ensure that the user has access to user agent functionalities through the keyboard since keyboard access is widely supported.

Ensuring access to user agent functionality through the system's standard keyboard API (where available) is important to accessibility since keyboard access is available to many users and is widely supported. Even when a user doesn't use a physical keyboard, it is still possible to simulate keyboard events with software. This guideline is important for ensuring compatibility between graphical desktop browsers and dependent user agents [p. 25] .

Checkpoints in this section do not apply to user agents (e.g., kiosks) that do not natively support keyboard input.

One area where configurability is important involves keyboard access to user agent functionality. Some users require single-key access, others require that keys activated in combination be physically close together, while others require that they be spaced physically far apart.

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. Refer also to checkpoint 9.12.

2.1 By default and without additional customization, ensure that all functionalities offered by the user agent may be operated through the standard keyboard API supported by the operating system. [Priority 1]

Note. Functionalities include being able to show, hide, resize and move graphical viewports [p. 28] created by the user agent.

2.2 Provide documentation on default keyboard commands and include with user agent documentation and/or user help system. [Priority 1]

Refer also to guideline 12.

2.3 Provide information to the user about the current keyboard configuration. [Priority 1]

Note. For example, users should be able to find information about complex key combinations. Refer also to guideline 12.

2.4 Allow the user to configure [p. 25] the keystrokes used to activate user agent functionalities. Users should be able to configure single key activation of functionalities. [Priority 2]
2.5 Allow the user to turn on and off author-specified keyboard configurations. [Priority 2]

For example, in HTML, the author may specify tabbing order with the "tabindex" attribute and keyboard bindings with the "accesskey" attribute.

2.6 Use platform conventions to indicate which keys activate which user agent functionalities. [Priority 2]

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

2.7 Avoid default keyboard configurations that interfere with system conventions. [Priority 2]

For example, the default configuration should not include "Alt-F4" or "Control-Alt-Delete" on 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 6.

2.8 Provide a default keyboard configuration for frequently performed operations. [Priority 3]

Guideline 3. Ensure user access to document content

Ensure that users have access to primary as well as author-supplied alternative representations of content (descriptions of images, closed captions for video or audio, etc.)

Otherwise, some users cannot perceive the primary content due to a disability or a technological limitation (e.g., browser configured [p. 25] not to display images).

The primary or alternative content may be rendered to the user through character, graphic, audio, speech, or braille devices. The differing characteristics of these devices mean that user requirements and capabilities will vary.

For example, browsers that output synthesized speech must convey sub-structures of text content like paragraphs, sentences, words and the spelling of individual words to the user as certain words or phrases may not be properly converted from their original text to their correct speech pronunciation. By conveying sub-structures of text, user agents allow users to replay and even spell words and phrases until the user understands the content and context. Speech-based user agents must use auditory nuances - including pitch, articulation model, volume, and orientation - to convey meaning the way fonts, spacing, and borders do in graphical media. For example, a user agent might speak the word "link" before a link, "header" before the text content of a header or "item 1.4" before a list item to convey context. Speech-based user agents must also heed author-specified markup that indicates changes in natural languages, and should render appropriately for supported languages.

In dynamic presentations such as synchronized multimedia presentations created with SMIL (refer to [SMIL [p. 31]]), content may change 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 example, in SMIL 1.0 authors may designate visual regions that are linked to different resources at different times; to the user, it appears as though a single link designates different resources dependent on when you select the link. In fact, there is a sequence of links over time that occupy the same place on the visual layout. To make the links accessible SMIL players might present time-dependent link sequences as a static list of links, allow the user to navigate the sequence of time-dependent links, allow the user to control the timing of changes, etc.

General checkpoints:

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

> 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.

3.2 *For dependent user agents only. [p. 25]* Ensure that the user has access to the content of an element selected by the user. [Priority 1]

> For instance, allow the user to identify a table cell with the selection and provide the user with cell content and (optionally) associated header information. Refer also to checkpoint 8.1 and checkpoint 6.2.

3.3 Render content according to natural language identification. For unsupported natural languages, notify the user of language changes when configured [p. 25] to do so. [Priority 1]

> Natural language may be identified by markup (e.g., the "lang" attribute in HTML or "xml:lang" in XML) or HTTP headers. Refer also to checkpoint 6.2.

3.4 Provide time-independent access to time-dependent active elements [p. 26] or allow the user to control the timing of changes. [Priority 1]

3.5 When no alternative text representation has been specified, indicate what type of object is present. [Priority 2]

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

Checkpoints for continuous equivalent tracks (closed captions, auditory descriptions, etc.):

3.7 Allow the user to specify that continuous equivalent tracks [p. 25] (e.g., closed captions, auditory descriptions, video of sign language, etc.) be rendered at the same time as audio and video tracks. [Priority 1]

3.8 If a technology allows for more than one continuous equivalent tracks [p. 25] (e.g., closed captions, auditory descriptions, video of sign language, etc.), allow the user to choose from among the tracks. [Priority 1]

Checkpoints for audio:

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

Guideline 4. Allow the user to turn off features that may reduce accessibility

Ensure that the user may turn off features that may obscure content or disorient the user.

Some rendering behavior may make the user agent unusable or may obscure information. For instance, people with photosensitive epilepsy must be able to turn off flashing within certain ranges, otherwise the flashing may trigger a seizure. Users who require specific color contrasts or who have low vision need to be able to turn off background images if those images interfere with their ability to read text. Dynamically changing web content or opening windows can be a problem for people using some types of dependent user agents [p. 25] and may be disorienting to users with cognitive disabilities. Flashing also affects screenreader users, since screenreaders (in conjunction with speech synthesizers or braille displays) may repeat the text every time it flashes.

User agents are only expected to provide this control for content or user interface controls that they recognizes. [p. 28] Please also refer to guideline 5 and guideline 11.

4.1 Allow the user to turn on and off rendering of images. [Priority 1]
4.2 Allow the user to turn on and off rendering of background images. [Priority 1]
4.3 Allow the user to turn on and off rendering of video. [Priority 1]
4.4 Allow the user to turn on and off rendering of sound. [Priority 1]
4.5 Allow the user to turn on and off rendering of continuous equivalent tracks [p. 25] (e.g., closed captions, auditory descriptions, video of sign language, etc.) [Priority 1]
4.6 Allow the user to turn on and off animated or blinking text. [Priority 1]
4.7 Allow the user to turn on and off animations and blinking images. [Priority 1]
4.8 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.
4.9 Allow the user to turn on and off support for user style sheets. [Priority 1]
4.10 Allow the user to turn on and off support for author style sheets. [Priority 1]
4.11 Allow the user to turn on and off support for spawned windows. [Priority 1]

4.12 Allow the user to choose between a frameset or its alternative supplied by the author. [Priority 2]

Note. For example, in HTML, allow the user to choose between a frameset or a NOFRAMES alternative. The ability to control frames is important for users with screen readers and users with some cognitive impairments.

4.13 Allow the user to turn on and off author-specified page forwards that occur after a time delay and without user intervention. [Priority 3]

For example, when turned off, offer a static link to the target resource instead.

4.14 Allow the user to turn on and off automatic page refresh. [Priority 3]

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

Guideline 5. Ensure user control over document styles

Ensure that the user has control over the colors, fonts, and other stylistic aspects of a document and can override author styles and user agent default styles.

In order to access a document, 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 fonts than user agent defaults or those specified by the author. Users who are blind may require audio or tactile rendering. Users who are deaf may require closed captions [p. 25] 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: 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.

Otherwise, users who are blind, have visual impairments, or have some types of learning disabilities, or any user who cannot or has chosen not to view the primary representation of information, will not know the content of the information on the page.

The following checkpoints state which styles the user must be able to control. The checkpoints also require user agents to allow users to override author styles [p. 28] and user agent defaults [p. 27] . The checkpoints apply to alternative representations of content as well as primary content.

Refer also to guideline 11.

Checkpoints for fonts and colors:

5.1 Allow the user to control font family. [Priority 1]
5.2 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.
5.3 Allow the user to control foreground color. [Priority 1]
5.4 Allow the user to control background color. [Priority 1]
5.5 Allow the user to control selection highlighting [p. 27] (e.g., foreground and background color). [Priority 1]
5.6 Allow the user to control focus highlighting [p. 27] (e.g., foreground and background color). [Priority 1]

Checkpoints for applets and animations:

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

Checkpoints for video.

5.8 Allow the user to control video frame rates. [Priority 1]
5.9 Allow the user to control the position of audio closed captions [p. 25] . [Priority 1]
5.10 Allow the user to start, stop, pause, and rewind video. [Priority 2]

Checkpoints for audio:

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

Checkpoints for synthesized speech:

5.14 Allow the user to control synthesized speech playback rate. [Priority 1]
5.15 Allow the user to control synthesized speech volume. [Priority 1]
5.16 Allow the user to control synthesized speech pitch, gender and other articulation characteristics. [Priority 2]

Checkpoints for changes to the user interface:

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

Guideline 6. Observe system conventions and standard interfaces

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

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.

6.1 Use and provide accessible interfaces to other technologies. [Priority 1]
To promote interoperability, open standards and W3C specifications should be used wherever possible.
6.2 Provide programmatic read and write access to user agent functionalities and user interface controls (including selection and focus) by using operating system and development language accessibility resources and conventions. [Priority 1]
6.3 Notify dependent user agents of changes to the document and user interface controls (including selection and focus) by using operating system and development language accessibility resources and conventions. [Priority 1]
6.4 Comply with W3C Document Object Model specifications and export interfaces defined by those specifications. [Priority 1]
For example, refer to [DOM, Level 1 [p. 31]]. User agents should export these interfaces using available 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."
6.5 Provide programmatic exchange of information in a timely manner. [Priority 2]
This is important for synchronous alternative renderings and simulation of events.
6.6 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 2.7.

Guideline 7. Support applicable W3C technologies and guidelines

User agents should support applicable W3C specifications and in particular the accessibility features defined for the specification.

The current guidelines recommend support for W3C specifications (e.g., HTML, CSS, MathML, SMIL, etc.) for several 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.

7.1 Implement the accessibility features defined for supported specifications. [Priority 1]

Note. The Techniques Document ([UA-TECHNIQUES] [p. 32]) discusses accessibility features of W3C specifications.

7.2 Support appropriate W3C Recommendations. [Priority 2]

For instance, for document markup, support HTML and XML; for style sheets, support CSS; for mathematics, support MathML; for multimedia, support SMIL, etc.

Refer also to checkpoint 6.4.

Guideline 8. Provide navigation mechanisms

Provide a variety of navigation mechanisms to allow convenient access to content, notably for serial devices such as synthesized speech output or single-line refreshable braille displays.

Navigation mechanisms help all users find important information in a document quickly. User agents should provide a variety of mechanisms - from simple scrolling through content to search mechanisms to tabbing navigation - to facilitate access, notably for users of devices that render documents serially (e.g., synthesized speech output or single-line refreshable braille displays).

While authors may provide some navigation mechanisms in documents (e.g., image maps or navigation bars), user agents must also provide tools. Each navigation mechanism has its advantages and disadvantages:

- 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 contextual information as one advances but can be slow. Sequential access may be based on element type (e.g., links only), document structure (e.g., navigate from header to header), or other criteria. Sequential access is very important to users reading an unfamiliar document since it allows access to all content. Structured navigation mechanisms allow users to move rapidly through highly structured documents 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 is lost. Searching on

text is one important variant of direct access, but other types of direct access are possible (e.g., go to the fifth link on the page). Selecting text or structured content with the pointing device is another form of direct access. Direct access is very important to users familiar with a document since it allows faster access to content.

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 11..

In addition to navigation mechanisms, user agents must provide contextual information that does not depend on a particular output device. Graphical user agents often use proportional scrollbars to indicate how much of a document has been viewed. User agents must make the same information available to users who cannot make use of the visual clues.

Note. For all search and navigation functions, the user agent should follow system conventions for using selection [p. 28] and focus [p. 26] mechanisms. For instance, the selection should be used to identify the results of a text search, the focus should identify active elements [p. 26] during sequential navigation of active elements, etc.

8.1 Allow the user to navigate viewports [p. 28] (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.

8.2 For user agents that offer a browsing history mechanism, when the user returns to a previous view, restore the point of regard [p. 29] in the viewport [p. 28] . [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.

8.3 *For dependent user agents only. [p. 25]* Allow the user to navigate just among table cells of a table (notably left and right within a row and up and down within a column). [Priority 1]

Refer also to checkpoint 9.4 and checkpoint 6.2.

8.4 Allow the user to navigate just among all active elements [p. 26] in the document. [Priority 2]

Navigation mechanisms may range from sequential (e.g., tabbing navigation) 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).

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

8.6 Allow the user to navigate the document structure. [Priority 2]

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

8.7 Allow the user to configure [p. 25] structured navigation. [Priority 3]
For example, allow the user to navigate only paragraphs, or only headers and paragraphs, etc.

Guideline 9. Help orient the user

Provide information about document structure, viewport structure, element summaries, and other information that will assist the user in understanding their browsing context.

Document information:

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 difficulty 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. 29] (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, or how one frame is updated when changes take place in a second. Refer also to checkpoint 10.1.

For people with visual impairments, blindness, or certain types of learning disabilities, it is important that the point of regard [p. 29] 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.

Viewport, selection, and focus information:

9.1 Provide a mechanism for highlighting [p. 27] 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 10.1..
9.2 *For dependent user agents only. [p. 25]* Provide the user with information about the number of viewports [p. 28] . [Priority 2]
Refer also to checkpoint 6.2.

Document information:

9.3 *For dependent user agents only. [p. 25]* Allow the user to view a document outline constructed from its structural elements (e.g., from header and list elements). [Priority 2]
The user should be able to control the level of detail of the outline. Refer also to checkpoint 6.2.
9.4 Describe a selected element's context within a document (e.g., numerical or relative position). [Priority 2]
For example: tenth link of fifty links; document header 3.4; list one of two, item 4.5; third table, three rows and four columns; current cell in third row, fourth

column; etc.

Link information:

9.5 For a selected link, indicate whether following the link will involve a fee. [Priority 2]

Note. 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. Refer to [MICROPAYMENT] [p. 31] for information about encoding micropayment information in markup in an interoperable manner.

9.6 For a selected link, provide information to help the user decide whether to follow the 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.

9.7 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 9.6.

9.8 Provide a mechanism for highlighting [p. 27] and identifying (through a standard interface where available) active elements of a document. [Priority 3]

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

Table information:

9.9 *For dependent user agents only. [p. 25]* Provide access to header information for a selected table cell. [Priority 1]

Refer also to checkpoint 6.2.

9.10 *For dependent user agents only. [p. 25]* Indicate the row and column dimensions of a selected table. [Priority 3]

Note. User agents should consider multidimensional tables, headers and footers, and multiple header levels. Refer also to checkpoint 6.2.

Form control information:

9.11 Provide information about form structure and navigation (e.g., groups of controls, control labels, navigation order, and keyboard configuration). [Priority 2]

For instance, provide information about controls with explicitly associated labels (the "for" attribute of LABEL in HTML), about which keys activate the form controls (the "accesskey" attribute in HTML), about the serial navigation order of the form controls (the"tabindex" attribute in HTML), and about control groups (the FIELDSET and OPTGROUP elements in HTML). Refer also to checkpoint 1.2 and checkpoint 8.4.

Consistency:

9.12 Maintain consistent user agent behavior and default configurations between software releases. Consistency is less important than accessibility and adoption of 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 10. Notify the user of document and viewport changes

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. 14] entirely). This is also important for security reasons; users should be able to decide whether to allow scripts to execute on their machines.

10.1 Provide information about document and viewport changes (to users and through programming interfaces). [Priority 1]

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

10.2 Ensure that when the selection or focus changes, it is in the viewport [p. 28] after the change. [Priority 2]

10.3 Allow the user to selectively turn on and off notification of common types of document and viewport changes. [Priority 3]

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

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

10.5 Indicate the relative position of the viewport in a resource (e.g., the percentage of the document that has been viewed, the percentage of an audio clip that has been played, etc.). [Priority 3]

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

10.6 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.

Guideline 11. Allow the user to configure the user agent

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

Web users have a wide range of functional capabilities and so they must be able to configure [p. 25] the user agent to meet their particular requirements.

Refer also to guideline 2. Refer also to checkpoint 9.12.

11.1 Allow the user to configure [p. 25] 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).

11.2 Allow the user to configure [p. 25] the graphical arrangement of user interface controls. [Priority 3]

Guideline 12. Provide accessible product documentation and help

Ensure that the user can 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.

It is important for developers to document user agent features that enhance accessibility to ensure since users with disabilities may not learn about those features from non-disabled users (who are not generally familiar with such features).

Documentation includes **all** product documentation, notably installation instructions, the help system, and all product manuals. Refer also to guideline 2 and checkpoint 6.6.

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

Refer to [WAI-WEBCONTENT [p. 32]].

12.2 Ensure that all user agent functionalities that promote accessibility are documented. [Priority 1]

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

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

Appendix A. -- Glossary

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 [p. 28] xat all.
- The checkpoint includes requirements about a content type that the user agent recognizes [p. 28] but does not support natively [p. 27] .
- 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, etc. In the area of Web Accessibility, common software-based assistive technologies include dependent user agents [p. 25] such as screen readers, screen magnifiers, speech synthesizers, onscreen keyboards, and voice input software. Hardware assistive

technologies include alternative keyboards and pointing devices.

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.

Dependent User Agent

Dependent user agents rely on other user agents for input and/or output. Dependent user agents 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.

Device Independence

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

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 representations of 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] [p. 32]) describes the different mechanisms authors use to supply alternative representations of content. Please also consult the Web Content Accessibility Guidelines ([WAI-WEBCONTENT] [p. 32]).

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. 31] 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. 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, elements with a value for the "longdesc" attribute, and elements with associated scripts (event handlers) explicitly associated with them (e.g., through the various "on" attributes). 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 element. For instance, when a link is activated, the user agent generally retrieves the linked resource, which may be another Web page, program, etc. When a form control is activated, it may change state (e.g., check boxes) or may take user input (e.g., a text field). Activating an element with a script assigned for that particular activation mechanism (e.g., mouse down event, key press event, etc.) causes the script to be executed.

A 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 [p. 27]) 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.

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 dependent user agents [p. 25] , so user agents must still make information available through programming interfaces.

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] [p. 31] , 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 [p. 31] .) 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] [p. 32]) 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 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 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 [p. 27]). 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. 25] 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 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 [p. 25] .

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 source information 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 [p. 27] 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.

---

5. 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, 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,

6. References

[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-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 latest draft of the techniques is available at http://www.w3.org/WAI/UA/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