User Agent Accessibility Guidelines 1.0

W3C Working Draft 23 October 2000

This version:
http://www.w3.org/TR/2000/WD-UAAG10-20001023
(plain text, gzip PostScript, gzip PDF, gzip tar file of HTML, zip archive of HTML)
Latest version:
http://www.w3.org/TR/UAAG10
Previous version:
http://www.w3.org/TR/2000/PR-UAAG10-20000310

Editors:
Ian Jacobs, W3C
Jon Gunderson, University of Illinois at Urbana-Champaign
Eric Hansen, Educational Testing Service
Authors and Contributors:
Refer to acknowledgements .

Copyright ©1999 - 2000 W3C® (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

Abstract

This document provides guidelines for designing user agents that lower barriers to Web accessibility for people with disabilities. User agents include HTML browsers and other software that retrieves and renders Web content . A user agent that conforms to these guidelines will promote accessibility through its own user interface and through other internal facilities, including its ability to communicate with other technologies (especially assistive technologies). By following these guidelines, developers will create more usable software for all Web users.

In addition to helping developers of HTML browsers, media players, etc., this document will also benefit developers of assistive technologies because it explains what types of information and control an assistive technology may expect from a conforming user agent. Technologies that are outside the scope of this document (e.g., those for braille rendering) will be essential to ensuring Web access for some users with disabilities.

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 23 October 2000 last call Working Draft of the "User Agent Accessibility Guidelines 1.0". This last call review period ends 13 November 2000. Please send comments before the review period ends to the public mailing list w3c-wai-ua@w3.org; public archives are available.

The previous formally published version of this document was a W3C Proposed Recommendation, for which the review period ended 7 April 2000. In light of Proposed Recommendation review comments, the W3C Director returned the document to the User Agent Accessibility Guidelines Working Group (UAWG) for further work. Since April, the UAWG has been revising the document to clarify its conformance requirements and make it easier to use (refer to the list of changes to the document). At their 12 October 2000 teleconference, the Working Group decided to advance the revised document to a second last call.

To help the UAWG build an implementation report (as part of advancing the document on the W3C Recommendation Track), reviewers are encouraged to evaluate software and indicate (e.g., by filling out a checklist *[UAAG10-CHECKLIST]*) which checkpoints that software satisfies.

Note: Three checkpoints in this document (checkpoint 5.1, checkpoint 5.2, and checkpoint 5.7) refer to three W3C DOM Level 2 specifications (*[DOM2CORE]* , *[DOM2HTML]* , *[DOM2STYLE]*) that are expected to become W3C Recommendations before the UAAG 1.0 becomes a Recommendation.

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite W3C Working Drafts as other than "work in progress."

This document is part of a series of accessibility documents published by the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C). WAI Accessibility Guidelines are produced as part of the WAI Technical Activity. The goals of the User Agent Accessibility Guidelines Working Group are described in the charter.

A list of current W3C Recommendations and other technical documents can be found at the W3C Web site.

Table of contents

Abstract 1
Status of this document 2
1. Introduction 5
1.1 Benefits of accessible user agent design 5
1.2 Scope of the document 6
2. The user agent accessibility guidelines 7
1. Support input and output device-independence. 9
2. Ensure user access to all content. 10
3. Allow the user to configure the user agent not to render some content that may reduce accessibility. 12
4. Ensure user control of styles. 13
5. Observe system conventions and standard interfaces. 17
6. Implement specifications that promote accessibility. 19
7. Provide navigation mechanisms. 20
8. Orient the user. 22
9. Allow configuration and customization. 24
10. Provide accessible product documentation and help. 26
3. Conformance 26
4. Glossary 32
5. References 45
6. Acknowledgments 48

An appendix to this document *[UAAG10-CHECKLIST]* lists all checkpoints for convenient reference (e.g., as a tool for developers to evaluate software for conformance).

Related resources

A separate document, entitled "Techniques for User Agent Accessibility Guidelines 1.0" *[UAAG10-TECHS]* , provides suggestions and examples of how each checkpoint might be satisfied. It also includes references to other accessibility resources (such as platform-specific software accessibility guidelines) that provide additional information on how a user agent may satisfy each checkpoint. The techniques provided in "Techniques for User Agent Accessibility Guidelines 1.0" are informative examples only, and other strategies may be used or required to satisfy the checkpoints. The Techniques document is expected to be updated more frequently than the current guidelines.

"User Agent Accessibility Guidelines 1.0" is part of a series of accessibility guidelines published by the Web Accessibility Initiative (WAI). This document explains the responsibilities of user agent developers in making the Web accessible to users with disabilities. The series also includes "Web Content Accessibility Guidelines 1.0" *[WCAG10]* , which explains the responsibilities of authors, and

"Authoring Tool Accessibility Guidelines 1.0" *[ATAG10]* , which explains the responsibilities of authoring tool developers. The WAI also makes available other resources *[WAIRESOURCES]* and educational materials to promote Web accessibility.

1. Introduction

This introduction (section 1) provides context for understanding the guidelines listed in section 2 . Section 1 explains some benefits of accessible user agent design and identifies some of the criteria that were used to establish the scope of requirements for conforming user agents. Section 3 explains how to make claims that software conforms to these guidelines and details about the applicability of the requirements for different kinds of user agents.

1.1 Benefits of accessible user agent design

People with (or without) disabilities access the Web with widely varying sets of capabilities, software, and hardware. Some users with disabilities:

- May not be able to see, hear, move, or may not be able to process some types of information easily or at all.
- May have difficulty reading or comprehending text.
- May not have or be able to use a keyboard or mouse.

This document specifies requirements that user agent developers must satisfy to lower barriers to accessibility. Many users without disabilities browse the Web with requirements similar to those of users with disabilities. For instance:

- They may have a text-only screen, a small screen, or a slow Internet connection (e.g., via a mobile phone browser). These users are likely to benefit from the same features that provide access to people with low vision or blindness.
- 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 noisy environment, etc.). These users are likely benefit from the same features that provide access to people who cannot use a mouse or keyboard due to a visual or physical disability.
- They may not understand fluently the natural language of spoken content. These users are likely to benefit from the same visual rendering of text equivalents that make spoken language accessible to people with a hearing disability.

Software that satisfies the requirements of this document will also be more flexible, manageable, extensible, and beneficial to all users. For example, a user agent architecture that allows programmatic access to content and the user interface will encourage software modularity and reuse, and will enable operation by scripting tools and automated test engines in addition to assistive technologies.

1.2 Scope of the document

This document was designed specifically to improve the accessibility of mainstream user agents with multimedia capabilities. In this context, a mainstream user agent is one designed for the general public to handle general-purpose content in ordinary operating conditions. It is expected that a conforming user agent will typically consist of a Web browser, one or more media players, and possibly other components.

A user agent that conforms to these guidelines will promote accessibility through its own user interface and through other internal facilities, including its ability to communicate with other technologies (especially assistive technologies). Technologies that are outside the scope of this document (e.g., those for braille rendering) will be essential to ensuring Web access for some users with disabilities. Note that the ability of conforming user agents to communicate well with assistive technologies will depend in part on the willingness of assistive technology developers to follow the same standards and conventions for communication.

This document does allow a certain amount of flexibility in the features a user agent must support in order to conform. For example, some user agents may conform even though they do not support certain content types such as video or audio. For more information on the "applicability " of certain checkpoints, please refer to the section on conformance .

The requirements of this document apply not just to the operation of a conforming user agent, but to its installation as well, including all subsequent software updates. A user with a disability must be able to install the software in order to use it at all. Because so many of the checkpoints in the document apply to aspects of the installation procedure, there is no specific checkpoint requiring an accessible installation. **Note:** User agent developers are **strongly** encouraged to design software that conforms in the default configuration. Users may not be able to install complementary software because the default configuration does not allow it easily (e.g., the mechanisms for retrieving and installing plug-ins are not accessible by default), because they don’t have access privileges on a public computer, etc.

Some of the requirements of this document were designed as complements to requirements made by the "Web Content Accessibility Guidelines 1.0" *[WCAG10]* . The current document also promotes the authoring of more accessible Web content by requiring conformance to specifications (refer to guideline 6). However, because Web content does not always conform to specifications, this document does include several repair checkpoints (for instance, checkpoint 2.5 and checkpoint 2.7). Other requirements in this document, while not strictly repair, also support authoring practices that may be widely deployed but that are discouraged because they cause accessibility or usability problems (e.g., some uses of HTML frames). For more information about Web content repair techniques, please refer to "Techniques For Accessibility Evaluation And Repair Tools" *[AERT]* .

Considerable effort has been made to ensure that the requirements of this document are compatible with other good software design practices. However, this document does not purport to be a complete guide to good software design. For instance, the general topic of user interface design for computer software exceeds the scope of this document, though some user interface requirements have been included because of their importance to accessibility. The "Techniques for User Agent Accessibility Guidelines 1.0" *[UAAG10-TECHS]* includes some references to general software design guidelines and platform-specific accessibility guidelines (refer to checkpoint 5.8). To ensure the accessibility of any piece of software, and in particular the user interface, people with disabilities should be involved in its design and testing.

Accessibility needs that are out of scope

Some accessibility needs not addressed, or not addressed completely by this document, include:

- Braille. This document does not address braille rendering.
- Speech and Voice. This document includes only three checkpoints related to synthesized speech (checkpoints 4.12, 4.13 and 4.14). This document includes several checkpoints related to voice input as part of general input requirements (e.g., use of standard APIs and configurability of voice input). This document does not otherwise address voice-based navigation or control. **Note:** The UAWG intends to coordinate further work on the topics of voice input and synthesized speech output with groups in W3C's Voice Browser Activity.
- Size and Color. This document includes some checkpoints to ensure that the user is able to control the size and color of visually rendered text content (checkpoints 4.1, 4.3, and 4.4). This document does not in general address control of the size and color of rendered non-text content .

2. The user agent accessibility guidelines

The ten 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 identification of some groups of users who benefit from it.
- A list of checkpoint definitions. This list may be split into groups of related checkpoints. For instance, the list might be split into one group of "checkpoints for content accessibility" and a second group of "checkpoints for user interface accessibility". Within each group, checkpoints are ordered according to their priority , e.g., Priority 1 before Priority 2.

Each checkpoint definition includes:

- The checkpoint number.
- The statement of the checkpoint. The statement of the checkpoint is one or more requirements that must be satisfied by the user agent, or "subject ," for the purposes of conformance . The "user agent" may consist of more than one software component, as explained in the section on well-formed conformance claims .
- The priority of the checkpoint.
- Informative notes about the checkpoint. These notes include examples, cross references, and commentary to help readers understand the scope of the checkpoint. **Note:** Some checkpoints in this document are more general than others, and some may overlap in scope. Therefore, a checkpoint may be identified as an "important special case" of one or more other checkpoints.
- A link to a corresponding section of "Techniques for User Agent Accessibility Guidelines 1.0" *[UAAG10-TECHS]* , where the checkpoint is examined in detail, including information about implementation and examples.

Each checkpoint has been designed to express a clear minimal requirement for the purposes of conformance . However, both this document and "Techniques for User Agent Accessibility Guidelines 1.0" *[UAAG10-TECHS]* suggest how users agent developers may surpass the minimal requirements to promote accessibility. **Note:** In some cases, though the requirement of a checkpoint may be clear, without documentation from vendors (e.g., about implemented APIs), it may be difficult to verify that the subject of a conformance claim has satisfied the requirement.

Priorities

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

[Priority 1]
: This checkpoint **must** be satisfied by user agents, otherwise one or more groups of users with disabilities will find it impossible to access the Web. Satisfying this checkpoint is a basic requirement for enabling some people to access the Web.

[Priority 2]
: This checkpoint **should** be satisfied by user agents, otherwise one or more groups of users with disabilities will find it difficult to access the Web. Satisfying this checkpoint will remove significant barriers to Web access for some people.

[Priority 3]
: This checkpoint **may** be satisfied by user agents to make it easier for one or more groups of users with disabilities to access information. Satisfying this checkpoint will improve access to the Web for some people.

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 people use a variety of devices for input and output, user agent developers must ensure redundancy in the user interface . Th user must be able to operate the user interface with a variety of input devices (mouse, keyboard, speech input, etc.) and output devices (graphical display, speech output, braille display, etc.). The user must also have access to the full benefit of Web content through each of at least three modalities -- visually-displayed text, synthesized speech, and braille. Text messages are generally accessible as text may be used by people with graphical displays, speech synthesizers, and braille displays.

People who cannot or do not use a mouse must be able to operate the user interface with the keyboard, through voice input, a head wand, touch screen, or other device. *Keyboard operation* of all functionalities offered through the user interface is one of the most important aspects of user agent accessibility on almost every platform. The keyboard is available to most users, it is widely supported, and specialized input devices may reuse the keyboard API .

To ensure that assistive technologies can both operate the user agent programmatically (e.g., through simulated keyboard events) and monitor user agent output (e.g., output text), developers are expected to use each API appropriately. Developers should not, for example, pre-rasterize text or since doing so may prevent assistive technologies from being able to render the text as speech or braille.

Checkpoints for communication with other software:

1.1 Ensure that every functionality available through the user interface is also available through every input API that is implemented by the user agent. This checkpoint does not require developers to reimplement the input methods associated with the keyboard, pointing device, voice, and other input APIs . [Priority 1]

> **Note:** This checkpoint does not require developers to implement all operating system input APIs , only to make the software accessible through those they do implement. Developers are not required to reimplement input methods of APIs , e.g., text input through a mouse API or pointer motion through a keyboard API.

1.2 Use the standard input and output APIs of the operating system. Do not bypass the standard output APIs when rendering information. [Priority 1]

> **Note:** For example, do not bypass (for reasons of speed, efficiency, etc.) standard APIs to manipulate the memory associated with rendered content , since assistive technologies may monitor rendering through the APIs . When available, developers should use APIs at a higher level of abstraction than the standard device APIs for the operating system. If these higher level APIs do not use the standard device APIs properly, developers should also use the standard

device APIs.

1.3 Implement the operating system's standard API for the keyboard and ensure that every functionality available through the user interface is available through this API. [Priority 1]

Note: This checkpoint is an important special case of checkpoint 1.1. Refer also to checkpoint 9.8.

Checkpoints for user interface accessibility:

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

Note: For example, users without a pointing device (such as some users who are blind or have physical disabilities) must be able to activate form controls and links (including the links in a client-side image map). This checkpoint is an important special case of checkpoint 1.1.

1.5 Ensure that every message (e.g., prompt , alert , notification, etc.) that is a non-text element and is part of the user agent user interface has a text equivalent . [Priority 1]

Note: For example, if the user is alerted of an event by an audio cue, a visually-rendered text equivalent in the status bar would satisfy this checkpoint. Per checkpoint 5.4, a text equivalent for each such message must be available through a standard API . Refer also to checkpoint 5.5.

Guideline 2. Ensure user access to all content.

Ensure that users have access to all content, notably author-specified equivalents such as text equivalents and auditory descriptions.

Just as people use a variety of devices for user interface input and output, they require that content be available in different modes -- auditory (synthesized speech and prerecorded audio), tactile (braille), graphical , or a mix of some of these. Authors and user agents share responsibility for ensuring redundant modes. Web content authors specify equivalents , such as text equivalents for images or video, according to the conventions of the markup language they are using (refer to the Techniques document *[UAAG10-TECHS]* for details). User agents must ensure that users have access to this content, as well as any content generated by the user agent itself. User agents should allow users to specify whether content should be rendered, equivalents for that content rendered, or both.

Ensuring access to equivalents benefits all users since some users may not have access to some content due to a technological limitation (e.g., their mobile browser cannot display graphics) or simply a configuration preference (e.g., they have a slow Internet connection and prefer not to download images).

Checkpoints for content accessibility:

2.1 Make all content available through the user interface. [Priority 1]

Note: Users must have access to the entire document object through the user interface, including recognized equivalents , attributes, style sheets, etc. This checkpoint does not require that all content be available in every viewport . A document source view is an important part of a solution for providing access to content, but is not a sufficient solution on its own for all content. Refer to guideline 5 for more information about programmatic access to content.

2.2 For a presentation that requires user input within a specified time interval, allow the user to configure the user agent to pause the presentation automatically and await user input before proceeding. [Priority 1]

Note: In this configuration, the user agent may have to pause the presentation more than once, depending on the number of times input is requested.

2.3 Provide easy access to each equivalent and each equivalency target through at least one of the following mechanisms: (1) allowing configuration to render the equivalent instead of the equivalency target; (2) allowing configuration to render the equivalent in addition to the equivalency target; (3) allowing the user to select the equivalency target and then inspect its equivalents; (4) providing a direct link to the equivalent in content, just before or after the equivalency target in document order. [Priority 1]

Note: For example, if an image in an HTML document has text equivalents , provide access to them (1) by replacing the image with the rendered equivalents, (2) by rendering the equivalents near the image, (3) by allowing the user to select the image and then inspect its equivalents, or (4) by allowing the user to follow readily available links to the equivalents.

2.4 Allow the user to specify that text transcripts , collated text transcripts , captions , and auditory descriptions be rendered at the same time as the associated audio and visual tracks. Respect author-specified synchronization cues during rendering. [Priority 1]

2.5 For non-text content that has no recognized text equivalent , allow configuration to generate repair text . If the non-text content is included by URI reference, base the repair text on the URI reference and content type of the Web resource . Otherwise, base the repair text on the name of the element that includes the non-text content . [Priority 2]

Note: For information on URI references, refer to "Uniform Resource Identifiers (URI): Generic Syntax" (*[RFC2396]* , section 4). Some markup languages (such as HTML 4 *[HTML4]* and SMIL 1.0 *[SMIL]* require the author to provide text equivalents for some content. When they don't, the user agent is required by this document to generate repair text . Refer also to checkpoint 2.6.

2.6 Allow configuration so that when the author has specified an empty text equivalent for non-text content , the user agent generates no repair text or generates repair text as required by checkpoint 2.5. [Priority 3]

Note: An empty text equivalent (e.g., `alt=""`) is considered to be a valid text equivalent in some authoring scenarios. For instance, when some non-text content has no other function than pure decoration, or an image is part of a "mosaic" of several images and doesn't make sense out of the mosaic. Please refer to the Web Content Accessibility Guidelines 1.0 *[WCAG10]* for more

information about text equivalents. Refer also to checkpoint 2.5.

2.7 Allow the user to configure the user agent not to render content marked up in a recognized but unsupported natural language . Indicate to the user in context that author-supplied content has not been rendered. [Priority 3]

Note: For example, use a text substitute or a graphical icon to indicate that content in a particular language has not been rendered. If a graphical icon is used, make the text substitute its text equivalent .

Guideline 3. Allow the user to configure the user agent not to render some content that may reduce accessibility.

Ensure that the user may turn off rendering of content (audio, video, scripts, etc.) that may reduce accessibility by obscuring content or disorienting 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, or may make a Web page too distracting to be usable by someone with a cognitive disability. Blinking can affect screen reader users, since screen readers (in conjunction with speech synthesizers or braille displays) may re-render the text every time it blinks. Distracting background images, colors, or sounds may make it impossible for users to see or hear other content.

Dynamically changing Web content may cause problems for some assistive technologies . Scripts that cause unanticipated changes (viewports that open, automatically redirected or refreshed pages, etc.) may disorient some users with cognitive disabilities.

To ensure that users have access to content, user agents must allow them to configure the user agent not to render certain content types when loading a Web resource . A user agent must allow this configurability even when it passes content (e.g., a sound file) to the operating system or to a helper application for rendering; the user agent is aware of the content type and thus can choose not to render it.

This guideline requires the user agent to allow configuration so that, when loading a Web resource , the user agent does not render portions of that resource that are of a particular type, or the user agent will render those portions in a way that does not pose accessibility problems.

Requirements for interactive control of rendered content are part of guideline 4.

Checkpoints for content accessibility:

3.1 Allow the user to configure the user agent not to render background images. In this configuration, provide an option to alert the user when a background image is available but has not been rendered. [Priority 1]

Note: This checkpoint only requires control of background images for "two-layered renderings", i.e., one rendered background image with all other

content rendered "above it". When background images are not rendered, user agents should render a solid background color (refer to checkpoint 4.4 and checkpoint 4.3). In this configuration, the user agent is not required to retrieve background images from the Web.

3.2 Allow the user to configure the user agent not to render audio, video, or animated images except on explicit request from the user. In this configuration, provide an option to render a substitute placeholder in context for each unrendered source of audio, video, or animated image. When placeholders are rendered, allow the user to activate each placeholder individually and replace it with the original author-supplied content. [Priority 1]

Note: This checkpoint requires configuration for content rendered without any user interaction (including content rendered on load or as the result of a script) as well as content rendered as the result of user interaction that is not an explicit request (e.g., when the user activates a link). Activation of a placeholder is considered an explicit user request to render the original content. When configured not to render content except on explicit user request, user agents may render the content "invisibly" or "silently" (i.e., in a manner that doesn't appear through the viewport). In this configuration, the user agent is not required to retrieve the audio, video, or animated image from the Web until requested by the user. Refer also checkpoint 4.6, checkpoint 4.10 and checkpoint 4.11.

3.3 Allow the user to configure the user agent to render animated or blinking text as motionless text. [Priority 1]

3.4 Allow the user to configure the user agent to render blinking images as motionless images. [Priority 1]

3.5 Allow the user to configure the user agent not to execute scripts or applets. In this configuration, provide an option to alert the user when scripts or applets are available. [Priority 1]

3.6 Allow configuration so that an author-specified "client-side redirect" (i.e., one initiated by the user agent, not the server) does not change content except on explicit user request . Allow the user to access the new content manually (e.g., by following a link). [Priority 2]

3.7 Allow configuration so that author-specified content refreshes do not change content except on explicit user request . Allow the user to request the new content manually (e.g., by activating a button or following a link). Continue to alert the user, according to schedule specified by the author, that a manual request will refresh the content. [Priority 2]

3.8 Allow the user to configure the user agent not to render images. [Priority 2]

Guideline 4. Ensure user control of styles.

Ensure that the user can select preferred styles (colors, size of rendered text, synthesized speech characteristics, etc.) from choices offered by the user agent. The user must be able to override author-specified styles and user agent default styles.

Providing access to content (refer to guideline 2) includes enabling users to configure its rendering. Users with low vision may require that text be rendered at a size larger than the size specified by the author or the user agent's default. Users with color blindness may need to impose or prevent certain color combinations.

For dynamic presentations such as synchronized multimedia presentations created with SMIL 1.0 *[SMIL]* , users with cognitive, hearing, visual, and physical disabilities may not be able to interact with a presentation within the time delays assumed by the author. To make the presentation accessible to these users, user agents rendering synchronized multimedia presentations or audio-only presentations must provide access to content in a time-independent manner and/or allow users to adjust the playback rate of the presentation.

User agents must also allow users to configure the style of the user interface elements, such as styles for selection and content focus (e.g., to ensure adequate color contrast).

For people with visual disabilities or certain types of learning disabilities, it is important that the point of regard remain as stable as possible. Unexpected changes may cause users to lose track of how many viewports are open, which is the current viewport, etc. For changes to the content or viewport that the user does not initiate, allow the user to request to be alerted when these changes occur (e.g., when a viewport opens, a script is executed, etc.).

For more information about configuration of style preferences, refer to guideline 9.

Note: The checkpoints in this guideline apply to all content, including equivalents .

Checkpoints for fonts and colors (content accessibility):

4.1 Allow the user to configure and control the reference size of rendered text with an option to override author-specified and user agent default sizes of rendered text. Make available the range of system font sizes. [Priority 1]

> **Note:** The reference size of rendered text corresponds to the default value of the CSS2 'font-size' property, which is 'medium' (refer to CSS2 *[CSS2]* , section 15.2.4). The default reference size of rendered text may vary among user agents. User agents may offer different mechanisms to allow the user to control the size of rendered text, for example by allowing the user to change the font size or by allowing the user to zoom or magnify content (refer, for example to the Scalable Vector Graphics specification *[SVG]*).

4.2 Allow the user to configure the font family of all text, with an option to override author-specified, and user agent default, font families. Allow the user to select from among the range of system font families. [Priority 1]

> **Note:** For example, allow the user to specify that all text must be rendered in a particular sans-serif font family.

4.3 Allow the user to configure the foreground color of all text, with an option to override author-specified, and user agent default, foreground colors. Allow the user to select from among the range of system colors. [Priority 1]

4.4 Allow the user to configure the background color of all text, with an option to override author-specified and user agent default background colors. Allow the user to select from among the range of system colors. [Priority 1]

Checkpoints for multimedia presentations , audio-only presentations , and visual-only presentations (content accessibility):

4.5 Allow the user to slow the presentation rate of audio, video and animations that are not recognized as style. For a visual track, provide at least one setting between 40% and 60% of the original speed. For a prerecorded audio track including audio-only presentations , provide at least one setting between 75% - 80% of the original speed. When the user agent allows the user to slow the visual track of a synchronized multimedia presentation to between 100% and 80% of its original speed, synchronize the visual and audio tracks. Below 80%, the user agent is not required to render the audio track . [Priority 1]

Refer also to checkpoint 2.4.

4.6 Allow the user to stop, pause, resume, fast advance, and fast reverse audio, video, and animations that last three or more seconds at their default playback rate and that are not recognized as style. [Priority 1]

Note: This checkpoint applies to content that is rendered automatically or on request from the user. Enable control of each independent source recognized as distinct. Respect synchronization cues per checkpoint 2.4. Refer also to checkpoint 3.2.

4.7 For graphical viewports, allow the user to position text transcripts , collated text transcripts , and captions in the viewport. Allow the user to choose from among the same range of positions available to the author (e.g., the range of positions allowed by the markup or style language). [Priority 1]

4.8 Allow the user to slow the presentation rate of audio, video and animations not covered by checkpoint 4.5. The same speed percentage requirements of checkpoint 4.5 apply. [Priority 2]

Note: User agents automatically satisfy this checkpoint if they satisfy checkpoint 4.5 for every audio, video, and animation.

4.9 Allow the user to stop, pause, resume, fast advance, and fast reverse audio, video, and animations not covered by checkpoint 4.6. [Priority 2]

Note: User agents automatically satisfy this checkpoint if they satisfy checkpoint 4.6 for every audio, video, and animation.

Checkpoints for audio volume control (content accessibility):

4.10 Allow the user to configure and control the global audio volume. The user must be able to choose zero volume (i.e., silent). [Priority 1]

Note: User agents should allow global control of volume through available system-level controls.

4.11 Allow the user to control independently the volumes of distinct audio sources synchronized to play simultaneously. [Priority 1]

Note: Refer also to checkpoint 4.13.

Checkpoints for synthesized speech (content accessibility):

4.12 Allow the user to configure and control synthesized speech playback rate according to the full range offered by the speech synthesizer. The lower bound for this range must be at most 120 words per minute. The upper bound for this range must be at least 400 words per minute. The user must be able to increase or decrease the playback rate in increments of 5% of the current playback rate. [Priority 1]

4.13 Allow the user to control the synthesized speech volume independently of other sources of audio. [Priority 1]

Note: Refer also to checkpoint 4.11.

4.14 Allow the user to configure synthesized voice gender, pitch, pitch range, stress, richness, and control of spelling, punctuation, and number processing according to the full range of values offered by the speech synthesizer. [Priority 2]

Note: This list of voice characteristic properties is based on the list in section 19.8 of Cascading Style Sheets Level 2 *[CSS2]* . Ranges of values for these properties may vary among speech synthesizers.

Checkpoints for user interface accessibility:

4.15 For user agents that support style sheets, allow the user to select from (and apply) available author and user style sheets or to ignore them. [Priority 1]

Note: By definition, the user agent's default style sheet is always present, but may be overridden by author or user styles.

4.16 Allow the user to configure how the selection is highlighted (e.g., foreground and background color, voice pitch, etc.). For graphical viewports, offer at least three rendering options, including colors and fonts. Allow the user to select from among the range of system colors and fonts. [Priority 1]

Note: For information for control of speech output and using those parameters for highlighting, refer to checkpoint 4.14.

4.17 Allow the user to configure how the content focus is highlighted (e.g., foreground and background color, voice pitch, etc.). For graphical viewports, offer at least three rendering options, including colors and fonts. For graphical viewports, allow the user to select from among the range of system colors and fonts. The default focus highlight mechanism must be different from the default selection highlight mechanism. [Priority 1]

Note: For information for control of speech output and using those parameters for highlighting, refer to checkpoint 4.14.

4.18 Allow the user to configure whether the current focus moves automatically to a viewport that opens without an explicit request from the user. [Priority 2]

4.19 Ensure that when a viewport's selection or content focus changes, it is in the viewport after the change. [Priority 2]

Note: For example, if users navigating links move to a portion of the document outside a graphical viewport, the viewport should scroll to include the new location of the focus. Or, for users of audio viewports, allow configuration to render the selection or focus immediately after the change.

4.20 Allow the user to configure the user agent to only open viewports on explicit user request . In this configuration, instead of opening a viewport automatically, alert the user and allow the user to open it manually. Allow the user to close viewports. [Priority 2]

> **Note:** User creation of a new viewport (e.g., empty or with a new resource loaded) through the user agent's user interface constitutes an explicit user request. Refer also to checkpoint 4.18 and checkpoint 5.5.

4.21 For graphical user interfaces, allow the user to configure the user agent so that the viewport with the current focus remains "on top" of all other viewports. In this configuration, when a viewport opens without explicit user request , alert the user. [Priority 2]

Guideline 5. Observe system conventions and standard interfaces.

Communicate with other software (e.g., assistive technologies, the operating system, plug-ins) through applicable interfaces. Observe system and programming language conventions for the user agent user interface, documentation, installation, etc.

Part of user agent accessibility involves communication within the user's "accessibility environment." This includes:

- exchanging information about content and user agent user interface controls with other user agents, especially with assistive technologies .
- using standard communication channels for this exchange.
- ensuring the exchange takes place in a timely manner. Otherwise, assistive technology rendering or response may lag behind that of the "source" user agent, which can disorient the user. Timely exchange is also necessary for proper synchronization of alternative renderings.
- following system conventions for user agent user interface design, documentation , and installation.
- incorporating system-level user preferences into the user agent. For instance, some operating systems include settings that allow users to request high-contrast colors (for users with low vision) or graphical rendering of audio cues (for users with hearing disabilities).

Using interoperable APIs and following system conventions increases predictability for users and for developers of assistive technologies . Platform guidelines explain what users will expect from the look and feel of the user interface, keyboard conventions, documentation, etc. Platform guidelines also include information about accessibility features that the user agent should adopt rather than reimplement.

Checkpoints for communication with other software:

5.1 Provide programmatic read access to HTML and XML content by conforming to the W3C Document Object Model (DOM) Level 2 Core and HTML Specifications and exporting the interfaces they define. [Priority 1]

Note: These specifications are defined the "Document Object Model (DOM) Level 2 Core Specification" *[DOM2CORE]* and the "Document Object Model (DOM) Level 2 HTML Specification" *[DOM2HTML]* . Please refer to those specifications for information about which versions of HTML and XML the specifications cover and for the definition of a "read-only" DOM. For content other than HTML and XML, refer to checkpoint 5.3.

5.2 If the user can modify HTML and XML content through the user interface , provide the same functionality programmatically by conforming to the W3C Document Object Model (DOM) Level 2 Core and HTML Specifications and exporting the interfaces they define. [Priority 1]

Note: For example, if the user interface allows users to complete HTML forms, this must also be possible through the DOM APIs . These specifications are defined the "Document Object Model (DOM) Level 2 Core Specification" *[DOM2CORE]* and the "Document Object Model (DOM) Level 2 HTML Specification" *[DOM2HTML]* . Please refer to those documents for information about which versions of HTML and XML the specifications cover. For markup languages other than HTML and XML, refer to checkpoint 5.3.

5.3 For markup languages other than HTML and XML, provide programmatic access to content using standard APIs (e.g., platform-independent APIs and standard APIs for the operating system). [Priority 1]

Note: This checkpoint addresses content not covered by checkpoints checkpoint 5.1 and checkpoint 5.2.

5.4 Provide programmatic read and write access to user agent user interface controls using standard APIs (e.g., platform-independent APIs such as the W3C DOM; standard APIs defined for a specific operating system; and conventions for programming languages, plug-ins, virtual machine environments, etc.) [Priority 1]

Note: For example, provide access to information about the user agent's current input configuration so that assistive technologies can trigger functionalities through keyboard events, mouse events, etc.

5.5 Using standard APIs , provide programmatic alert of changes to content and user interface controls (including selection , content focus , and user interface focus). [Priority 1]

Note: Use the standard APIs required by guideline 5.

5.6 Ensure that programmatic exchanges proceed in a timely manner. [Priority 2]

Note: For example, the programmatic exchange of information required by other checkpoints in this document must be efficient enough to prevent information loss, a risk when changes to content or user interface occur more quickly than the communication of those changes. The techniques for this checkpoint explain how developers can reduce communication delays, e.g., to ensure that assistive technologies have timely access to the document object model and other information needed for accessibility.

5.7 For user agents that implement Cascading Style Sheets (CSS), provide programmatic access to CSS style sheets by conforming to the W3C Document Object Model (DOM) Level 2 Style Specification and exporting the interfaces it defines. [Priority 3]

Note: As of the publication of this document, Cascading Style Sheets (CSS) are defined by CSS Level 1 *[CSS1]* and CSS Level 2 *[CSS2]* . The DOM style specification is defined by "Document Object Model (DOM) Level 2 Style Specification" *[DOM2STYLE]* . Please refer to that specification for information about which versions of CSS the DOM style specification covers.

Checkpoints for user interface accessibility:

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

Note: Operating system conventions that benefit accessibility are those described in this document and in platform-specific accessibility guidelines. Some of these conventions (e.g., sticky keys, mouse keys, show sounds, etc.) are discussed in the Techniques document *[UAAG10-TECHS]* . Refer also to checkpoint 9.2.

Guideline 6. Implement specifications that promote accessibility.

Support the accessibility features of all implemented specifications. Implement W3C Recommendations when available and appropriate for a task.

Developers should implement open specifications. Conformance to open specifications promotes interoperability and accessibility by making it easier to design assistive technologies (also discussed in guideline 5).

While developers should implement the accessibility features of any specification, this document promotes W3C specifications 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. This review includes review from stakeholders in accessibility.
- W3C specifications are developed in a consensus process (refer to the process defined by the W3C Process Document *[W3CPROCESS]*). W3C encourages the public to review and comment on specifications at all times during their development, from Working Draft to Candidate Recommendation (for implementation experience) to Proposed Recommendation. For information about how specifications become W3C Recommendations, refer to The W3C Recommendation track (*[W3CPROCESS]* , section 6.2). W3C Recommendations (and other technical reports) are published at the W3C Web

site.

Checkpoints for content accessibility:

6.1 Implement the accessibility features of all implemented specifications (markup languages, style sheet languages, metadata languages, graphics formats, etc.). The accessibility features of a specification are those identified as such and those that satisfy *all* of the requirements of the "Web Content Accessibility Guidelines 1.0" *[WCAG10]* . [Priority 1]

Note: This checkpoint applies to both W3C-developed and non-W3C specifications. The Techniques document *[UAAG10-TECHS]* provides information about the accessibility features of some specifications, including W3C specifications.

6.2 Use and conform to W3C Recommendations when they are available and appropriate for a task. [Priority 2]

Note: For instance, for markup, conform to HTML 4 *[HTML4]* , XHTML 1.0 *[XHTML10]* , or XML 1.0 *[XML]* . For style sheets, conform to CSS (*[CSS1]* , *[CSS2]*). For mathematics, conform to MathML *[MATHML]* . For synchronized multimedia, implement SMIL 1.0 *[SMIL]* . For information about programmatic access to HTML and XML content, refer to guideline 5. User agents may conform to other specifications in addition to those required by this checkpoint. For reasons of backward compatibility, user agents should continue to implement deprecated features of specifications. Information about deprecated language features is generally part of the language's specification.

Guideline 7. Provide navigation mechanisms.

Provide access to content through a variety of navigation mechanisms: sequential navigation, direct navigation, searches, structured navigation, etc.

Users should be able to navigate to important pieces of content within a configurable view, identify the type of object they have navigated to, interact with that object easily (if it is an active element), and recall the surrounding context (to orient themselves). Providing a variety of navigation mechanisms helps users with disabilities (and all users) access content more efficiently. Content navigation is particularly important to users who access content serially (e.g., as synthesized speech or braille).

Sequential navigation (e.g., line scrolling, page scrolling, sequential navigation through active elements, etc.) means advancing (or rewinding) through rendered content in well-defined steps (line by line, screen by screen, link by link, etc.). Sequential navigation can provide context, but can be time-consuming. Sequential navigation is important to users who cannot scan a page visually for context and also benefits users unfamiliar with a page. Sequential access may be based on element type (e.g., links only), content structure (e.g., navigation from heading to heading), or other criteria.

Direct navigation (go to a particular link or paragraph, search for instances of a string, etc.) is faster than sequential navigation, but generally requires familiarity with the content. Direct navigation is important to users with some physical disabilities (who may have little or no manual dexterity and/or increased tendency to push unwanted buttons or keys), to users with visual disabilities, and also benefits "power users." Selecting text or structured content with the pointing device is another form of direct navigation. Searching on text is one important variant of direct navigation.

Structured navigation mechanisms offer both context and speed. User agents should allow users to navigate to content known to be structurally important: blocks of content, headers and sections, tables, forms and form elements, active elements, navigation mechanisms, containers, etc. For information about programmatic access to document structure, refer to guideline 5.

User agents should allow users to configure navigation mechanisms (e.g., to allow navigation of links only, or links and headings, or tables and forms, etc.).

Checkpoints for user interface accessibility:

7.1 Allow the user to navigate among all 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 accessed or viewed one at a time (e.g., by a text browser or speech synthesizer), provide a list of links to other frames. Navigation among all viewports implies at least allowing the user to cycle through all viewports. Navigating into a viewport makes it the current viewport .

7.2 Associate a point of regard with each state in a viewport's browsing history and when the user returns to a state in the history, restore the associated point of regard. [Priority 1]

Note: For example, when the user navigates from one viewport to another (per checkpoint 7.1) and back, restore the point of regard .

7.3 Allow the user to navigate all active elements . If the author has not specified a navigation order, allow at least forward sequential navigation of elements, in document order. [Priority 1]

Note: Navigation may include non-active elements in addition to active elements. This checkpoint is an important special case of checkpoint 7.6.

7.4 Allow the user to choose to navigate only active elements . If the author has not specified a navigation order, allow at least forward and reverse sequential navigation of active elements, in document order. [Priority 2]

7.5 Allow the user to search forward through text content that has been rendered . The search must encompass all text within the viewport, both inside and outside the point of regard . Allow the user to start a search from any selected or focused location in content. When there is a match, allow the user to search for the next instance of the text from the location of the match. When there is a match, move the point of regard so that the matched text is in the viewport. Alert the user when there is no match. Provide a case-insensitive search option when applicable to the natural language of text. [Priority 2]

Note: The default search starting point should be the beginning of content. Use

operating system conventions for indicating the result of a search (e.g., selection or content focus).

7.6 Allow the user to navigate efficiently to and among important structural elements identified by the author. Allow forward and backward sequential navigation to important structural elements. [Priority 2]

Note: This specification intentionally does not identify the set of "important elements" that must be navigable; refer to the Techniques document *[UAAG10-TECHS]* for information about identifying important elements. Structured navigation of headings, tables, forms, lists, etc., is most effective in conjunction with a configurable view (refer to configuration requirements of checkpoint 8.4 and checkpoint 7.7). User agents should follow operating system conventions for indicating navigation progress (e.g., selection or content focus).

7.7 Allow the user to configure and control the set of important elements required by checkpoint 7.6 and checkpoint 8.4. Allow the user to include and exclude element types in the set of elements. [Priority 3]

Note: For example, allow the user to navigate only paragraphs, or only headings and paragraphs, etc. Refer also to checkpoint 5.4.

Guideline 8. Orient the user.

Provide information that will help the user understand browsing context.

All users require clues to help them understand their "location" when browsing: where they are, how they got there, where they can go, what's nearby, etc. Some mechanisms that provide such clues include:

- information about browsing context such as proportional scroll bars, a visually highlighted selection, a history mechanism, the title of the current document or frame, etc. These clues must be available in a device-independent manner;
- information about elements, such as the dimensions of a table, the length of an audio clip, the structure of a form, whether following a link will involve a fee, etc.
- information about relationships among elements, such as between table cells and related table headers.

Orientation mechanisms such as these are especially important to users who view content serially, (e.g., when rendered as speech or braille). For instance, these users cannot "scan" a graphically displayed table with their eyes for information about a table cell's headers, neighboring cells, etc. User agents must provide other means for users to understand table cell relationships, frame relationships (what relationship does the graphical layout convey?), form context (have I filled out the form completely?), link information (have I already visited this link?), etc.

Checkpoints for content accessibility:

8.1 Make available to the user the author-specified purpose of each table and the author-specified relationships among the table cells and headers. [Priority 1]

Note: Depending on the table, some techniques may be more efficient than

others for conveying data relationships. For many tables, user agents rendering in two dimensions may satisfy this checkpoint by rendering a table as a grid and by ensuring that users can find headers associated with cells. However, for large tables or small viewports, allowing the user to query cells for information about related headers may improve access. Refer also to checkpoint 5.3. This checkpoint is an important special case of checkpoint 2.1.

8.2 Render recently visited links in a distinct style and allow the user to configure this style. For graphical viewports, offer at least three rendering options, including colors and fonts. Allow the user to select from among the range of system colors and fonts. [Priority 2]

Note: Do not use color as the only distinguishing factor between visited and unvisited links as some users may not perceive colors and some devices may not render them. This checkpoint is an important special case of checkpoint 8.5.

8.3 Render in a distinct style those links that have been marked up to indicate that following them will involve a fee and allow the user to configure this style. For graphical viewports, offer at least three rendering options, including colors and fonts. Allow the user to select from among the range of system colors and fonts. [Priority 2]

Note: This checkpoint is an important special case of checkpoint 8.5.

8.4 Make available to the user an "outline" view of content , composed of labels for important structural elements (e.g., heading text, table titles, form titles, etc.). For discussion about what constitutes the set of important structural elements, please refer to checkpoint 7.6. [Priority 2]

Note: This checkpoint is meant to allow the user to simplify the view of content by hiding some content selectively. For example, for each frame in a frameset, provide a table of contents composed of headings (e.g., the H1 - H6 elements in HTML) where each entry in the table of contents links to the heading in the document. This checkpoint does not require that the outline view be navigable, but this is recommended; refer to checkpoint 7.6. For those elements that do not have associated text titles or labels, the user agent should generate a brief text label (e.g., from content, the element type, etc.). Refer also to checkpoint 7.7.

8.5 To help the user decide whether to traverse a link, make available the following information about it: link content, link title, whether the link is internal to the local resource, whether the user has traversed the link recently, whether traversing it may involve a fee, and information about the type, size, and natural language of linked Web resources. The user agent is not required to compute or make available information that requires retrieval of linked Web resources . [Priority 3]

Checkpoints for user interface accessibility:

8.6 Implement selection , content focus , and user interface focus mechanisms. Implement them according to system conventions (per checkpoint 5.8). [Priority 1]

Note: This checkpoints refers to the *logical* selection and focus; rendering requirements are addressed by checkpoint 8.7, checkpoint 4.17, and checkpoint 4.16. Refer also to checkpoint 7.1.

8.7 Provide a mechanism for highlighting the current viewport , selection , and content focus . [Priority 1]

Note: This includes highlighting and identifying frames. This checkpoint is an

important special case of checkpoint 1.1. Refer also to checkpoints checkpoint 4.16, checkpoint 5.8, and checkpoint 8.5.

8.8 Provide a mechanism for highlighting and identifying active elements . [Priority 2]

Note: On most systems, the focus is used to identify and highlight active elements.

8.9 Allow configuration so the user is prompted to confirm any form submission not caused by explicit user request to activate a form submit control. [Priority 2]

Note: For example, do not submit a form automatically when a menu option is selected, when all fields of a form have been filled out, or when a mouseover event occurs. The user agent may satisfy this checkpoint by prompting the user to confirm *all* form submissions.

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

Note: The user agent may calculate the relative position according to content focus position, selection position, or viewport position, depending on how the user has been browsing. The user agent may indicate the proportion of content viewed in a number of ways, including as a percentage, as a relative size in bytes, etc. For two-dimensional renderings, relative position includes both vertical and horizontal positions.

Guideline 9. Allow configuration and customization.

Allow users to configure the user agent so that frequently performed tasks are made convenient, and allow users to save their preferences.

Web users have a wide range of capabilities and must be able to configure the user agent according to their preferences for styles, graphical user interface configuration, keyboard configuration, etc.

Checkpoints for user interface accessibility:

9.1 Provide information to the user about current user preferences for input configurations (e.g., keyboard or voice bindings). [Priority 1]

9.2 Avoid default input configurations that interfere with operating system accessibility conventions. [Priority 1]

Note: In particular, default configurations should not interfere with operating conventions for keyboard accessibility. Information about operating system accessibility conventions is available in the Techniques document *[UAAG10-TECHS]* . Refer also to checkpoint 5.8.

9.3 Provide information to the user about current author-specified input configurations (e.g., keyboard bindings specified in HTML documents with the "accesskey" attribute). [Priority 2]

9.4 Allow the user to change the default input configuration as follows: Allow the user to override any binding that is part of the user agent default input configuration (checkpoint 9.8). The user agent is not required to allow the user to override

standard bindings for the operating system (e.g., for access to help). For any binding in the default keyboard configuration, allow the user to override it with a binding of a single key alone or with modifier keys. [Priority 2]

> **Note:** This checkpoint applies to all supported input methods: keyboard, voice, graphical user interface, etc. The override requirement only applies to bindings for the same input method (i.e., the user must be able to override a keyboard binding with another keyboard binding). Refer also to checkpoint 9.5, checkpoint 9.9, checkpoint 9.8, and checkpoint 10.3.

9.5 Allow the user to assign a single-key binding to at least a majority of the functionalities available in the default keyboard configuration (refer to checkpoint 9.8). [Priority 2]

> **Note:** In some modes of interaction (e.g., when the user is entering text), the number of available single keys will be significantly reduced. The number of available single keys will also be determined by the keyboard device capabilities. This checkpoint is an important special case of checkpoint 9.4. Refer also to checkpoint 1.3, checkpoint 9.9, checkpoint 9.8, and checkpoint 10.3.

9.6 Follow operating system conventions to indicate the input configuration . [Priority 2]

> **Note:** For example, on some operating systems, developers may specify which command sequence will activate a functionality so that the standard user interface components display that binding. For example, if a functionality is available from a menu, the letter of the activating key will be underlined in the menu. This checkpoint is an important special case of checkpoint 5.8.

9.7 For the configuration requirements of this document, allow the user to save user preferences in at least one user profile . Allow users to select from among available profiles or no profile (i.e., the user agent default settings). [Priority 2]

> **Note:** The configuration requirements of the checkpoints in this document involve user preferences for styles, presentation rates, input configurations , navigation, viewport behavior, and user agent alerts.

9.8 Ensure that the default input configuration includes bindings for the following functionalities required by other checkpoints in this document: move focus to next active element; move focus to previous active element; activate focused link; search for text; search again for same text; next history state (forward); previous history state (back); increase size of rendered text; decrease size of rendered text; increase global volume; decrease global volume; (each of) stop, pause, resume, fast advance, and fast reverse selected audio, video, and animation. If the user agent supports the following functionalities, the default input configuration must also include bindings for them: enter URI for new resource; add to favorites (i.e., bookmarked resources); view favorites; stop loading resource; reload resource; refresh rendering; forward one viewport; back one viewport; next line; previous line. [Priority 2]

9.9 For graphical user interfaces, allow the user to configure the position of controls on tool bars of the user agent user interface , to select or remove controls for the user interface from a predefined set, and to restore the default user interface. [Priority 3]

Note: This checkpoint is an important special case of checkpoint 9.4.

Guideline 10. Provide accessible product documentation and help.

Ensure that the user can learn about software features from documentation, and in particular, features that relate to accessibility.

Documentation includes anything that explains how to install, get help for, use, or configure the product. At least one version of the documentation must conform to the Web Content Accessibility Guidelines 1.0 *[WCAG10]* .

Features that support accessibility must be clearly documented so that users with disabilities can learn to operate the user agent efficiently. Documentation of keyboard accessibility is particularly important to users with visual disabilities and some types of physical disabilities. Without this documentation, a user with a disability (or multiple disabilities) may not think that a particular task can be performed. Or the user may try to use a much less efficient technique to perform a task, such as using a mouse, or using an assistive technology's mouse emulation through key strokes.

Refer also to checkpoint 5.8.

Checkpoints for accessible documentation:

10.1 Ensure that at least one version of the product documentation conforms to at least Level Double-A of the Web Content Accessibility Guidelines 1.0 *[WCAG10]* . [Priority 1]
10.2 Document all user agent features that promote accessibility. [Priority 1]
Note: For example, review the documentation or help system to ensure that it includes information about the accessibility requirements of WAI Guidelines.
10.3 Document the default input configuration (e.g., default keyboard bindings). [Priority 1]
10.4 In a dedicated section of the documentation , describe all features of the user agent that promote accessibility. [Priority 2]
Note: This is a more specific requirement than checkpoint 10.2.
10.5 In each software release, document all changes that affect accessibility. [Priority 2]
Note: Features that affect accessibility are listed in this document and in platform-specific accessibility guidelines.

3. Conformance

This normative section explains how to make a valid claim that a user agent conforms to this document. The terms "must", "should", and "may" (and related terms) are used in this document in accordance with RFC 2119 *[RFC2119]* . Throughout this section the term "subject" refers to one or more software components (generally a browser plus additional software) treated as a unit for the

purposes of a conformance claim.

Anyone may make a claim (e.g., vendors about their own products, third parties about those products, journalists about products, etc.). Claims may be published anywhere (e.g., on the Web or in product documentation).

Claimants are solely responsible for their claims and the use of the conformance icons . If the subject of the claim changes after the date of the claim, the claimant is responsible for updating the claim. Claimants are encouraged to conform to the most recent "User Agent Accessibility Guidelines 1.0" available.

Note: Conformance to the requirements of this document is expected to be a strong indicator of accessibility, but it is neither a necessary nor sufficient condition for ensuring the accessibility of software. Some software may not conform to this document but still be accessible to some users with disabilities. Conversely, software may conform to this document but still be inaccessible to some users with disabilities. Please refer to the section on the scope of requirements for conforming user agents .

3.1 Content type labels

Some subjects may not support every content type (which in this context has a meaning similar to the term "media type" in RFC 2046 *[RFC2046]*). This document has been designed primarily to promote the accessibility of subjects that support the rendering of most content types available on the Web today, including text, images, animations, video, and audio. However, this document allows some flexibility in conformance claims, so that claims may be made, for example, for subjects that don't support audio, or don't support synthesized speech. To facilitate comparison of claims (and to shorten the claims), this document requires that each claim include one or more labels to identify the content type (or types) supported by the subject. Each label refers to a set of checkpoints. Each checkpoint in a given set includes one or more requirements related to the content type in question.

This is the list of valid content type labels and corresponding sets of checkpoints.

Text
: Requirements of the following checkpoints related to text or text rendering: 3.3, 4.1, and 4.2.

Color
: Requirements of the following checkpoints related to color: 3.4, 4.3 and 4.4.

Image
: Requirements of the following checkpoints related to images: 3.1 and 3.8. When this label is used, the user agent must implement at least one image format.

Animation
: Requirements of the following checkpoints related to animated images: 3.2, 3.4, 4.5, 4.6, 4.8, and 4.9. When this label is used, the user agent must implement at least one animation format.

Video
: Requirements of the following checkpoints related to video: 2.4, 3.2, 4.5, 4.6, 4.8, and 4.9 When this label is used, the user agent must implement at least one video format.

Audio
: Requirements of the following checkpoints related to audio except for synthesized speech: 2.4, 3.2, 4.5, 4.6, 4.8, 4.9, 4.10, and 4.11. When this label is used, the user agent must implement at least one audio format.

Speech
: Requirements of the following checkpoints related to speech: 4.12, 4.13 and 4.14. When this label is used, the user agent must support synthesized speech.

Basic
: This is an abbreviation for all the requirements of the following labels: Text, Color, Image, and Animation.

Multimedia
: This is an abbreviation for all the requirements of the following labels: Text, Color, Image, Animation, Video, and Audio.

All
: This is an abbreviation for all the requirements of the following labels: Text, Color, Image, Animation, Video, Audio, and Speech.

Note: Some of the labels above require implementation of at least one format (e.g., for images). This document does not require implementation of specific formats, (e.g., PNG *[PNG]* versus SVG *[SVG]* for images). However, please refer to the requirements of checkpoint 6.2.

3.2 Use of operating system features as part of conformance

Developers are encouraged to adopt operating system conventions and features that benefit accessibility to meet the requirements of this document. However, if these features are not accessible, developers must provide an alternative accessible solution.

Developers may, but are not required to, provide access to adopted operating system features through the user agent's user interface. For example, if the user agent relies on the operating system's audio control features to meet some requirements of this document, the user agent is not required to include those controls in its own user interface.

3.3 Checkpoint applicability

This section of the document explains when the subject of the claim is or is not required to satisfy a checkpoint for the purposes of conformance. For a given claim, a checkpoint applies:

- If it is one of the checkpoints in the set identified by a content type label that appears in the claim.
- If it is not in any of the sets of checkpoints identified by a valid content type label

. In other words, all other checkpoints than those in the identified sets form a "core" of checkpoints that must be satisfied by the subject, in addition to at least one set identified by a content type label .

- Unless any one of the following conditions is met:
  - The checkpoint makes requirements for graphical user interfaces or graphical viewports and the subject only has an audio or tactile user interface or viewports.
  - The checkpoint refers to a role of content (e.g., transcript, caption, text equivalent, etc.) that the subject does not recognize . For instance, HTML user agents can recognize "alt", OBJECT content, or NOFRAMES content as providing equivalents for other content since these are specified by the markup language. HTML user agents are not expected to recognize that a text description embedded without indicative markup in a nearby paragraph is a text equivalent for the image.
  - The checkpoint requires control of content properties (e.g., video or animation rate) that the subject cannot control (e.g., the format does not allow it) or does not recognize (e.g., because the property is controlled by a script in a manner that the subject cannot recognize).

Some checkpoints include more than one requirement. The subject is required to satisfy all applicable requirements of a checkpoint.

The subject **must** implement at least one API for the keyboard in order to conform to this document. Per checkpoint 1.3, the subject **must** implement the standard system API for the keyboard . Consequently, checkpoints that refer to the keyboard always apply.

Each checkpoint requirement must be satisfied by making information or functionalities available through the user interface of the subject unless the checkpoint explicitly states that the requirement must be met by making information available through an application programming interface (API); communication checkpoints are labeled "checkpoints for communication with other software."

3.4 Conformance levels

A conformance claim must indicate what conformance level is met:

- **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 (e.g., "Double-A" rather than "AA") so they may be understood when rendered as speech.

3.5 Well-formed conformance claims

A well-formed claim must include the following information:

About the guidelines:

- The guidelines title/version: "User Agent Accessibility Guidelines 1.0".
- The URI of the guidelines: http://www.w3.org/TR/2000/WD-UAAG10-20001023.
- The conformance level satisfied: "A", "Double-A", or "Triple-A".
- One or more valid content type labels .
- The checkpoints of the chosen conformance level considered not applicable . Claimants may use the checklist *[UAAG10-CHECKLIST]* for this purpose.

The subject of the claim may consist of one or more software components (e.g., a browser plus a multimedia player plus a plug-in). For each component, the claim must include the following:

- The product name and version information (version number, minor release number, and any required patches or updates). The claim must also include the vendor name if it is required to identify the product.
- The operating system name and version number.

Properties of the claim:

- The date of the claim.

There is no restriction on the format used to make the claim, except that at least one representation of the claim must conform to the Web Content Accessibility Guidelines 1.0 *[WCAG10]* and should conform to at least Level Double-A. For instance, the claim may be marked up using HTML, or expressed in the Resource Description Framework (RDF) *[RDF10]* Here is an example of a claim expressed in HTML:

```
<p>On 23 October 2000, Project X (version 2.3) running on
MyOperatingSystem (version 4.2) conforms to <abbr title="the World Wide Web
Consortium">W3C</abbr>'s "User Agent Accessibility Guidelines 1.0",
http://www.w3.org/TR/2000/WD-UAAG10-20001023, level Double-A. This
product satisfies the "Multimedia" checkpoints (as described in section 3.4 of the
UAAG 1.0). The <a href="http://example.com/checkpoints"> list of checkpoints
that do not apply</a> is available online.</p>
```

3.6 Validity of a claim

A conformance claim is valid for a given conformance level if:

1. The claim is well-formed , and
2. The subject of the claim *as a whole* satisfies all the applicable checkpoints for that level. **Note:** A subject may consist of more than one software component, and taken together they must satisfy applicable checkpoints. This includes assistive technologies that are part of a claim. Some components may not have

to satisfy some requirements as long as the subject *as a whole* satisfies them. For instance, a particular component of the subject may not have to conform to the DOM APIs required by guideline 5 as long as the subject as a whole makes *all content* available through those APIs.

Claimants (or relevant assuring parties) are responsible for the validity of a claim. As of the publication of this document, W3C does not act as an assuring party, but it may do so in the future, or establish recommendations for assuring parties.

Claimants are expected to modify or retract a claim if it may be demonstrated that the claim is not valid. Please note that it is not currently possible to validate claims completely automatically.

3.7 Conformance icons

As part of a conformance claim, people may use a conformance icon on a Web site, on product packaging, in documentation, etc. Each conformance icon (chosen according to the appropriate conformance level) must link to the W3C explanation of the icon. The appearance of a conformance icon (or, "conformance logo") does not imply that W3C has reviewed or validated the claim. An icon must be accompanied by a well-formed claim .

Note: *In the event this document becomes a W3C Recommendation, additional information about the icons and how to use them will be available at the W3C Web site.*

4. Glossary

Active element

An active element is an element with behaviors that may be ***activated*** (or "triggered") either through the user interface or through an API (e.g., by using scripts). Some element instances may be active at times but not at others (e.g., they may be "deactivated" through scripts, or they may only active for a period of time determined by the author). Which elements are active depends on the document language and whether the features are supported by the user agent. In HTML 4 *[HTML4]* documents, for example, active elements include links, image maps, form controls, element instances with a value for the "longdesc" attribute, and element instances with scripts (event handlers) explicitly associated with them (e.g., through the various "on" attributes). Most systems use the content focus to navigate active elements and identify which is to be activated. 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 Web resource . When a form control is activated, it may change state (e.g., check boxes) or may take user input (e.g., a text entry field). Refer also to the definition of event handler .

Alert

In this document, "to alert" means to make the user aware of some event, without requiring acknowledgement. For example, the user agent may alert the user that new content is available on the server by displaying a text message in the user agent's status bar. Refer to checkpoint 1.5 for requirements about alerts.

Application Programming Interface (API), ***standard input/output/device API***

An application programming interface (API) defines how communication may take place between applications.

As part of encouraging interoperability, this document recommends using standard APIs where possible, although this document does not define in all cases how those APIs are standardized (i.e., whether they are defined by specifications such as W3C Recommendations, defined by an operating system vendor, de facto standards, etc.). Implementing APIs that are independent of a particular operating system (e.g., the W3C DOM Level 2 specifications) may reduce implementation costs for multi-platform user agents and promote the development of multi-platform assistive technologies. Implementing standard APIs defined for a particular operating system may reduce implementation costs for assistive technology developers who wish to interoperate with more than one piece of software running on that operating system.

A "device API" defines how communication may take place with an input or output device such as a keyboard, mouse, video card, etc. A "standard device API" is one that is considered standard for that particular device on a given operating or windowing system.

In this document, an "input/output API" defines how applications or devices communicate with a user agent. As used in this document, input and output APIs include, but are not limited to, device APIs (and in general, they define a more abstract communication interface than device APIs). A "standard input/output API" is one that is expected to be implemented by software running on a particular operating system. Standard input/output APIs may vary from system to system. For example, on desktop computers today, the standard input APIs are for the mouse and keyboard. For touch screen devices or mobile devices, standard input APIs may include stylus, buttons, voice, etc. The graphical display and sound card are considered standard ouput devices for a graphical desktop computer environment, and each has a standard API.

Assistive technology

In the context of this document, an assistive technology is a user agent that:

1. relies on services (such as retrieving Web resources , parsing markup, etc.) provided by one or more other "host" user agents. Assistive technologies communicate data and messages with host user agents by using and monitoring APIs .
2. provides services beyond those offered by the host user agents to meet the requirements of a users with disabilities. Additional services include alternative renderings (e.g., as synthesized speech or magnified content), alternative input methods (e.g., voice), additional navigation or orientation mechanisms, content transformations (e.g., to make tables more accessible), etc.

For example, screen reader software is an assistive technology because it relies on browsers or other software to enable Web access, particularly for people with visual and learning disabilities.

Examples of assistive technologies that are important in the context of this document include the following:

- screen magnifiers, which are used by people with visual disabilities to enlarge and change colors on the screen to improve the visual readability of rendered text and images.
- screen readers, which are used by people who are blind or have reading disabilities to read textual information through synthesized speech or braille displays.
- speech recognition software, which may be used by people who have some physical disabilities.
- alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard.
- alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations.

Beyond this document, assistive technologies consist of software or hardware that has been specifically designed to assist people with disabilities in carrying out daily activities, e.g., wheelchairs, reading machines, devices for grasping, text telephones, vibrating pagers, etc.

Attribute

This document uses the term "attribute" in the XML sense: an element may have a set of attribute specifications (refer to the XML 1.0 specification *[XML]* section 3).

Audio, Audio object

An audio object is output from an audio viewport .

Audio-only presentation

An audio-only presentation is a presentation consisting exclusively of one or more audio tracks presented concurrently or in series. Examples of an audio-only presentation include a musical performance, a radio-style news broadcast, and a book reading.

Audio track

An audio track is an audio object that is intended as a whole or partial presentation . An audio track can, but does not necessarily correspond to a single audio channel (left or right audio channel).

Auditory description

An auditory description is either a prerecorded human voice or a synthesized voice (recorded or generated dynamically) describing the key visual elements of a movie or animation. 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.

Author styles

Authors styles are style property values that come from a document, its associated style sheets, or are generated by the server.

Captions

Captions (or sometimes "closed captions") are text transcripts that are synchronized with other audio or visual tracks. Captions convey information about spoken words and non-spoken sounds such as sound effects. They benefit people who are deaf or hard-of-hearing, and anyone who cannot hear the audio (e.g., someone in a noisy environment). Captions are generally rendered graphically above, below, or superimposed over video. **Note:** Other terms that include the word "caption" may have different meanings in this document. For instance, a "table caption" is a title for the table, often positioned graphically above or below the table. In this document, the intended meaning of "caption" will be clear from context.

Collated text transcript

A collated text transcript is a text equivalent of a movie or animation. More specifically, it is the combination of the text transcript of the audio track and the text equivalent of the visual track. For example, a collated text transcript typically includes segments of spoken dialogue interspersed with text descriptions of the key visual elements of a presentation (actions, body language, graphics, and scene changes). Refer also to the definitions of text transcript and auditory description . Collated text transcripts are essential for individuals who are deaf-blind.

Configure and ***Control***

In the context of this document, both the terms "control" and "configure" share in common the idea of governance such as a user may exercise over interface layout, user agent behavior, rendering style, and other parameters required by this document. Generally, the difference in the terms centers on the idea of *persistence*. When a user makes a change by "controlling" a setting, that change usually does not persist beyond that user session. On the other hand, when a user "configures" a setting, that setting typically persists into later user sessions. Furthermore, the term "control" typically means that the change can be made easily (such as through a keyboard shortcut) and that the results of the change occur immediately, whereas the term "configure" typically means that making the change requires more time and effort (such as making the change via a series of menus leading to a dialog box, via style sheets or scripts, etc.) and that the results of the change may not take effect immediately (e.g., due to time spent reinitializing the system, initiating a new session, rebooting the system). In order to be able to configure and control the user agent, the user must be able to "read" as well as "write" values for these parameters. Configuration settings may be stored in a profile . The range and granularity of the changes that can be controlled or configured by the user may depend on system or hardware limitations.

Both configuration and control may apply at different "levels": across Web resources (i.e., at the user agent level, or inherited from the system), to the entirety of a Web resource, or to components of a Web resource (e.g., on a per-element basis). For example, users may configure the user agent to apply the same font family across Web resources, so that all text is displayed by default using that font family. Or, the user may wish to configure the rendering of a particular element type, which may be done through style sheets. Or, the user may wish to control the text size dynamically (zooming in and out) for a given document, without affecting the Web resource-level configuration. Or, the user may wish to control the text size dynamically for a given element, e.g., by navigating to the element and zooming in on it.

Note: In this document, the noun "control" means "user interface component" or "form component".

Content

In this specification, the term "content" is used in three ways:

1. Content refers to the document object as a whole or in parts.
2. Content refers to the content of an HTML or XML element, in the sense employed by the XML 1.0 specification (*[XML]* , section 3.1): "The text between the start-tag and end-tag is called the element's content." Context should indicate that the term content is being used in this sense.
3. Content is used in the context of the phrases non-text content and text content .

Device-independence

Device-independence refers to the ability to make use of software with any supported input or output device. User agents should follow operating system

conventions and use standard system APIs for input and output.

Document Object, ***Document Object Model***

In general usage, the term "document object" refers to the user agent's representation of data (e.g., a document). This data generally comes from the document source , but may also be generated (from style sheets, scripts, transformations, etc.), produced as a result of preferences set within the user agent, added as the result of a repair performed automatically by the user agent, etc. Some data that is part of the document object is routinely rendered (e.g., in HTML, what appears between the start and end tags of elements and the values of attributes such as "alt", "title", and "summary"). Other parts of the document object are generally processed by the user agent without user awareness, such as DTD-defined names of element types and attributes, and other attribute values such as "href", "id", etc. These guidelines require that users have access to both types of data through the user interface.

A "document object model" is the abstraction that governs the construction of the user agent's document object. The document object model employed by different user agents may vary in implementation and sometimes in scope. This specification requires that user agents implement the APIs defined in the "Document Object Model (DOM) Level 2 Specification" (*[DOM2CORE]* , *[DOM2HTML]* , *[DOM2STYLE]*) for access to HTML, XML, and CSS content. These DOM APIs allow authors to access and modify the content via a scripting language (e.g., JavaScript) in a consistent manner across different scripting languages. As a standard interface, the DOM APIs make it easier not just for authors, but for assistive technology developers to extract information and render it in ways most suited to the needs of particular users. The relevant W3C DOM Recommendations are listed in the references .

Document source, ***Document source view***

In this document, the term "document source" refers to the data that the user agent receives as the direct result of a request for a Web resource (e.g., as the result of an HTTP/1.1 *[RFC2616]* "GET", as the result of opening a local resource, etc.). A "document source view" generally renders the document source as text written in the markup language(s) used to build it. The document source is generally a subset of the document object (e.g., since the document object may include repair content).

Documentation

Documentation refers to **all** information provided by the vendor about a product, including all product manuals, installation instructions, the help system, and tutorials.

Element

This document uses the term "element" both in the XML sense (an element is a syntactic construct as described in the XML 1.0 specification *[XML]* , section 3) and more generally to mean a type of content (such as video or sound) or a logical construct (such as a header or list).

Equivalent (for content)

In the context of this document, an equivalency relationship between two pieces of content means that one piece -- the "equivalent" -- is able to serve

essentially the same function for a person with a disability (at least insofar as is feasible, given the nature of the disability and the state of technology) as the other piece -- the "***equivalency target***" -- does for a person without any disability. For example, the text "The Full Moon" might convey the same information as an image of a full moon when presented to users. If the image is part of a link and understanding the image is crucial to guessing the link target, then the equivalent must also give users an idea of the link target. Thus, an equivalent is provided to fulfill the same function as the equivalency target.

Equivalents include text equivalents (e.g., text equivalents for images; text transcripts for audio tracks; collated text transcripts for multimedia presentations and animations) and non-text equivalents (e.g., a prerecorded auditory description of a visual track of a movie, or a sign language video rendition of a written text, etc.). Please refer to the definitions of text content and non-text content for more information.

Each markup language defines its own mechanisms for specifying equivalents. For instance, in HTML 4 *[HTML4]* or SMIL 1.0 *[SMIL]* , authors may use the "alt" attribute to specify a text equivalent for some elements. In HTML 4, authors may provide equivalents (or portions of equivalents) in attribute values (e.g., the "summary" attribute for the TABLE element), in element content (e.g., OBJECT for external content it specifies, NOFRAMES for frame equivalents, and NOSCRIPT for script equivalents), and in prose. Please consult the Web Content Accessibility Guidelines 1.0 *[WCAG10]* and its associated Techniques document *[WCAG10-TECHS]* for more information about equivalents.

Events and scripting, event handler

User agents often perform a task when an event occurs that is due to user interaction (e.g., document loading, mouse motion or a key press), a request from the operating system, etc. Some markup languages allow authors to specify that a script, called an ***event handler***, be executed when the event occurs. **Note:** The combination of HTML, style sheets, the Document Object Model (DOM) and scripting is commonly referred to as "Dynamic HTML" or DHTML. However, as there is no W3C specification that formally defines DHTML, this document only refers to event handlers and scripts.

Explicit user request

In several checkpoints in this document, the term "explicit user request" is used to mean any user interaction recognized with certainty to be for a specific purpose. For instance, when the user selects "New viewport" in the user agent's user interface, this is an explicit user request for a new viewport. On the other hand, it is not an explicit request when the user activates a link and that link has been marked up by the author to open a new viewport (since the user may not know that a new viewport will open). Nor is it an explicit user request even if the link text states "will open a new viewport". Some other examples of explicit user requests include "yes" responses to prompts from the user agent, configuration through the user agent's user interface, activation of known form submit controls, and link activation (which should not be assumed to mean more than "get this linked resource", even if the link text or title or role indicates more). Some examples of behaviors that happen without explicit user request include

changes due to scripts. **Note:** Users make mistakes. For example, a user may submit a form inadvertently by activating a known form submit control. In this document, this type of mistake is still considered an explicit user request.

Focus, ***content focus***, ***user interface focus***, ***current focus***

The notion of focus refers to two identifying mechanisms of user agents:

1. The "content focus" designates an active element in a document. A viewport has at most one content focus.
2. The "user interface focus" designates a control of the user interface that will respond to user input (e.g., a radio button, text box, menu, etc.).

In this document, the term "focus" by itself encompasses both types of focus. Where one is meant specifically in this document, it is identified.

When several viewports coexist, each may have a content and user interface focus. At all times, only one content focus **or** one user interface focus is active, called the current focus. The current focus responds to user input and may be toggled between content focus and user interface focus through the keyboard, pointing device, etc. Both the content and user interface focus may be highlighted . Refer also to the definition of point of regard .

Graphical

In this document, the term "graphical" refers to information (text, colors, graphics, images, animations, etc.) rendered for visual consumption.

Highlight

In this document, "to highlight" means to emphasize through the user interface. For example, user agents highlight which content is selected or focused and which viewport is the current viewport. Graphical highlight mechanisms include dotted boxes, underlining, and reverse video. Synthesized speech highlight mechanisms include alterations of voice pitch and volume.

Input configuration

An input configuration is the mapping of user agent functionalities to some user interface trigger mechanisms (e.g., menus, buttons, keyboard keys, voice commands, etc.). The default input configuration is the mapping the user finds after installation of the software; it must be part of the user agent documentation (per checkpoint 10.3]).

Multimedia Presentation

For the purposes of this document, a multimedia presentation is a presentation that is not a visual-only presentation , audio-only presentation , or tactile-only presentation . In a "classic" multimedia presentation (e.g., a movie that has sound track or an animation with accompanying audio), at least one visual track is closely synchronized with at least one audio track .

Natural language

Natural language is spoken, written, or signed human language such as French, Japanese, and American Sign Language. On the Web, the natural language of content may be specified by markup or HTTP headers. Some examples include the "lang" attribute in HTML 4 (*[HTML4]* section 8.1), the "xml:lang" attribute in XML 1.0 (*[XML]* , section 2.12), the HTML 4 "hreflang" attribute for links in HTML 4 (*[HTML4]* , section 12.1.5), the HTTP Content-Language header

(*[RFC2616]* , section 14.12) and the Accept-Language request header *([RFC2616]* , section 14.4).

Point of regard

The point of regard is a position in rendered content that the user is presumed to be viewing. The dimensions of the point of regard may vary. For example, it may be a point (e.g., a moment in an audio rendering or a cursor in a graphical rendering), or a range of text (e.g., focused text), or a two-dimensional area (e.g., content rendered through a two-dimensional graphical viewport). The point of regard is almost always within a viewport (though the dimensions of the point of regard could exceed those of the viewport). The point of regard may also refer to a particular moment in time for content that changes over time (e.g., an audio-only presentation). User agents may use the focus , selection , or other means to designate the point of regard. A user agent should not change the point of regard unexpectedly as this may disorient the user.

Presentation

In this document, the term presentation refers to a collection of information, consisting of one or more Web resources , intended to be rendered simultaneously, and identified by a single URI. In general, a presentation has an inherent time component (i.e., it's not just a static "Web page" (refer to the definition of "Web page" in "Web Characterization Terminology and Definitions Sheet" *[WEBCHAR]*).

Profile

A profile is a named and persistent representation of user preferences that may be used to configure a user agent. Preferences include input configurations, style preferences, etc. On systems with distinct user accounts, profiles enable users to reconfigure software quickly when they log on, and they may be shared by several users. Platform-independent profiles are useful for those who use the same user agent on different platforms.

Prompt

In this document, "to prompt" means to require input from the user. The user agent should allow users to configure how they wish to be prompted. For instance, for a user agent functionality X, configurations might include: always do X without prompting me, never do X without prompting me, don't ever do X but tell me when you could have done X but didn't, don't ever do X and don't tell me, etc.

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, on paper, through speakers, on a braille display, on a mobile device, etc. Style information (e.g., fonts, colors, voice inflection, etc.) may come from the elements themselves (e.g., certain font and phrase elements 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. Generally in this document, the term "property" has the meaning defined in CSS 2 (*[CSS2]* , section 3). 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 installed is called the property's ***default value***.

Recognize
A user agent is said to recognize a piece of information when the user agent developer has designed it to handle that information. A user agent recognizes those features of markup or style languages that it implements and the behavior of the user interface controls that it provides. User agents may not understand everything the author has encoded in content, such as the semantics of XML elements unknown to the user agent, whether the link text and link title accurately describe the linked resource, whether a sentence (that has not been specially marked up) is a text equivalent for an image, or whether a script is calculating a factorial. A user agent does not recognize everything that a script does, even though it may implement the scripting language. However, it will recognize some information encoded in scripts, such as code to open a viewport or retrieve a resource from the Web. The Techniques document *[UAAG10-TECHS]* lists some markup known to affect accessibility that should be recognized by user agents.

Rendered content
The rendered content is that part of content rendered in a given viewport (whether visual, auditory, or tactile).

Repair content, ***repair text***
In this document, the term "repair content" refers to content generated by the user agent in order to correct an error condition or as the result of a user preference. "Repair text" means repair content consisting only of text . This document does not require user agents to include repair content in the document object .

Some error conditions that may lead to the generation of repair content include:

- Erroneous or incomplete content (e.g., ill-formed markup, invalid markup, missing text equivalents , etc.);
- Missing resources for handling or rendering content (e.g., the user agent lacks a font family to display some characters, the user agent doesn't implement a particular scripting language, etc.);

Some user preferences may change content, such as when the user has turned off support for images and a placeholder icon to appears in place of each image that has not been loaded.

For more information about repair techniques for Web content and software, refer to "Techniques For Accessibility Evaluation And Repair Tools" *[AERT]* .

Selection, ***current selection***
The selection generally identifies a range of content (e.g., 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 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 selection (though the selection may be rendered graphically as discontinuous text fragments). When several viewports coexist, each may have a selection, but only one is active, called the current selection.

On the screen, the selection may be highlighted using colors, fonts, graphics, magnification, etc. The selection may also be rendered as inflected speech, for example.

Support, ***implement***, ***conform***

In this document, the terms "support", "implement", and "conform" all refer to what a developer has designed a user agent to do, but they represent different degrees of specificity. A user agent "supports" general classes of objects, such as "images" or "Japanese". A user agent "implements" a specification (e.g., the PNG and SVG image format specifications, a particular scripting language, etc.) or an API (e.g., the DOM API) when it has been programmed to follow all or part of a specification. A user agent "conforms to" a specification when it implements the specification *and* satisfies its conformance criteria. This document includes some explicit conformance requirements (e.g., to a particular level of the "Web Content Accessibility Guidelines 1.0" *[WCAG10]*).

Synchronize

In this document, "to synchronize" refer to the time-coordination of two or more presentation components (e.g., in a multimedia presentation, a visual track with captions). For Web content developers, the requirement to synchronize means to provide the data that will permit sensible time-coordinated rendering by a user agent. For example, Web content developer can ensure that the segments of caption text are neither too long nor too short, and that they map to segments of the visual track that are appropriate in length. For user agent developers, the requirement to synchronize means to present the content in a sensible time-coordinated fashion under a wide range of circumstances including technology constraints (e.g., small text-only displays), user limitations (slow reading speeds, large font sizes, high need for review or repeat functions), and content that is sub-optimal in terms of accessibility.

Text content, ***non-text content***, ***text element***, ***non-text element***, ***text equivalent***, ***non-text equivalent***

In this document, the term "text element" means content that, when rendered, is understandable in *each* of three modes to three reference groups:

1. visually-displayed text, for users who are deaf and adept in reading visually-displayed text;
2. synthesized speech, for users who are blind and adept in use of synthesized speech;
3. braille, for users who are deaf-blind and adept at reading braille.

In these definitions, a text element is said to be "understandable" when it fulfills its communication function to representatives of the three reference groups. Furthermore, these definitions make assumptions such as the availability of appropriate hardware and software, that content represents a general mix of purposes (information, education, entertainment, commerce), that the individuals in the groups are able to understand the natural language of the content, that the individuals in the groups are not required to have specialized skills (e.g., computer science degree), etc.

A text element may contain markup for style (e.g., font size or color), structure (e.g., heading levels), and other semantics. However, the essential function of the text element should be retained even if style information happens to be lost in rendering. In this document, the term "text content" refers to content that is composed of one or more text elements. A "non-text element" is an element that *fails* to be understandable when rendered in *any* of three modes to their respective reference disability audiences. Thus, text elements have essential accessibility advantages often associated with text while non-text elements are those that lack one or more such advantages.

In this document, the term "non-text content" refers to content that is composed of one or more non-text elements. Per checkpoint 1.1 of "Web Content Accessibility Guidelines 1.0" *[WCAG10]* , authors must provide a text equivalent for every author-supplied non-text element. Similarly, user agent developers must provide a text equivalent for every non-text element offered by the user agent to the user (refer to checkpoint 1.5).

Note that the terms "text element" and "non-text element" are defined by the characteristics of their output (e.g., rendering) rather than those of their input (e.g., information sources) or their internals (e.g., format). For example, in principle, a text element can be generated or encoded in any fashion as long as it has the proper output characteristics. In general, text elements are composed of text (i.e., a sequence of characters). Both text elements and non-text elements should be understood as "pre-rendering" content in contrast to the "post-rendering" content that they produce.

A "text equivalent" is a text element that, when rendered, serves essentially the same function as some other content (i.e., an equivalency target) does for a person without any disability. Similarly, a "non-text equivalent" is a non-text element that, when rendered, serves essentially the same function as the equivalency target does for a person without any disability. Please refer also to the definition of equivalent .

Text transcript

A text transcript is a text equivalent of audio information (e.g., an audio-only presentation or the audio track of a movie or animation). It provides text for both spoken words and non-spoken sounds such as sound effects. Text transcripts make audio information accessible to people who have hearing disabilities and to people who cannot play the audio. Text transcripts are usually pre-written but may be generated on the fly (e.g., by speech-to-text converters).

Refer also to the definitions of captions and collated text transcripts .

Tactile object

A tactile object is output from a tactile viewport . Tactile objects include text (rendered as braille) and graphics (rendered as raised-line drawings).

Tactile-only presentation

A tactile-only presentation is a presentation consisting exclusively of one or more tactile tracks presented concurrently or in series.

Tactile track

A tactile track is a tactile object that is intended as a whole or partial presentation . This does not necessarily correspond to a single physical or logical track on the storage or delivery media.

User agent

In this document, the term "user agent" is used in two ways:

1. Any software that retrieves and renders Web content for users. This may include Web browsers, media players, plug-ins, and other programs -- including assistive technologies -- that help in retrieving and rendering Web content.
2. The subject of a conformance claim to this document. This is the most common use of the term in this document and is the usage in the checkpoints.

Text

In this document, the term "text" used by itself refers to a sequence of characters from a markup language's document character set (e.g., Unicode or ISO 10646). Refer to the "Character Model for the World Wide Web " *[CHARMOD]* for more information about text and characters. **Note:** This document makes use of other terms that include the word "text" that have highly specialized meanings: collated text transcript , non-text content , text content , non-text element , text element , text equivalent , and text transcript .

User agent default styles

User agent default styles are style property values applied in the absence of any author or user styles. Some markup languages specify a default rendering for documents in that markup language. Other specifications may not specify default styles. For example, XML 1.0 *[XML]* does not specify default styles for XML documents. HTML 4 *[HTML4]* does not specify default styles for HTML documents, but the CSS 2 *[CSS2]* specification suggests a sample default style sheet for HTML 4 based on current practice.

User interface

For the purposes of this document, user interface includes both:

1. the "***user agent user interface***", i.e., the controls and mechanisms offered by the user agent for user interaction, such as menus, buttons, keyboard access, etc.
2. the "content user interface", i.e., the active elements that are part of content, such as form controls, links, applets, etc.

The document distinguishes them only where required for clarity.

User styles

User styles are style property values that come from user interface settings, user style sheets, or other user interactions.

Visual object

A visual object is output from a visual viewport . Visual objects include graphics, text, and visual portions of movies and animations.

Visual-only presentation

A visual-only presentation is a presentation consisting exclusively of one or more visual tracks presented concurrently or in series.

Visual track

A visual track is a visual object that is intended as a whole or partial presentation . A visual track does not necessarily correspond to a single physical or software object. A visual track can be text-based or graphic, static or animated.

Views, viewports, and current viewport

User agents may handle different types of content : markup language, sound, video, etc. The user views rendered content through a ***viewport***, which may be a window, a frame, a piece of paper, a speaker, a virtual magnifying glass, etc. A viewport may contain another viewport (e.g., nested frames). Viewports do not include user interface controls such as prompts, menus, alerts, etc.

The viewport that contains both the current focus and the current selection is called the ***current viewport***. The current viewport is generally highlighted when several viewports coexist. A user agent must provide mechanisms for accessing all content that can be presented by each viewport (e.g., scrolling mechanisms, advance and rewind, 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.

Web resource

The term "Web resource" is used in this document in accordance with Web Characterization Terminology and Definitions Sheet *[WEBCHAR]* to mean anything that can be identified by a Uniform Resource Identifier (URI) as defined in RFC 2396 *[RFC2396]* .

5. References

For the **latest version** of any W3C specification please consult the list of W3C Technical Reports at http://www.w3.org/TR. Some documents listed below may have been superseded since the publication of this document.

[AERT]
"Techniques For Accessibility Evaluation And Repair Tools", C. Ridpath, W. Chisholm, eds., 26 April 2000. This W3C Working Draft is http://www.w3.org/TR/2000/WD-AERT-20000426.

[ATAG10]
"Authoring Tool Accessibility Guidelines 1.0", J. Treviranus, C. McCathieNevile, I. Jacobs, and J. Richards, eds., 3 February 2000. This W3C Recommendation is http://www.w3.org/TR/2000/REC-ATAG10-20000203.

[CHARMOD]
"Character Model for the World Wide Web", M. Dürst, 29 November 1999. This W3C Working Draft is http://www.w3.org/TR/1999/WD-charmod-19991129/

[CSS1]
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. This W3C 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 W3C Recommendation is http://www.w3.org/TR/1998/REC-CSS2-19980512.

[DOM2CORE]
"Document Object Model (DOM) Level 2 Core Specification", M. Davis, A. Le Hors, P. Le Hégaret, J. Robie, L. Wood, eds., 27 September 2000. This W3C Proposed Recommendation is http://www.w3.org/TR/2000/PR-DOM-Level-2-Core-20000927.

[DOM2HTML]
"Document Object Model (DOM) Level 2 HTML Specification", A. Le Hors, P. Le Hégaret, eds., 27 September 2000. This W3C Proposed Recommendation is http://www.w3.org/TR/2000/PR-DOM-Level-2-HTML-20000927

[DOM2STYLE]
"Document Object Model (DOM) Level 2 Style Specification", V. Apparao, P. Le Hégaret, C. Wilson, eds., 27 September 2000. This W3C Proposed Recommendation is http://www.w3.org/TR/2000/PR-DOM-Level-2-Style-20000927.

[HTML4]
"HTML 4.01 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds., 24 December 1999. This W3C Recommendation is http://www.w3.org/TR/1999/REC-html401-19991224.

[MATHML]
"Mathematical Markup Language", P. Ion and R. Miner, eds., 7 April 1998. This W3C Recommendation is http://www.w3.org/TR/1998/REC-MathML-19980407.

[PNG]
"PNG (Portable Network Graphics) Specification 1.0", T. Boutell, ed., 1 October 1996. This W3C Recommendation is http://www.w3.org/TR/REC-png.

[RDF10]
"Resource Description Framework (RDF) Model and Syntax Specification", O. Lassila, R. Swick, eds., 22 February 1999. This W3C Recommendation is http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[RFC2046]
"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed and N. Borenstein, November 1996.

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

[RFC2396]
"Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, R. Fielding, L. Masinter, August 1998.

[RFC2616]
"Hypertext Transfer Protocol -- HTTP/1.1, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999.

[SMIL]
"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", P. Hoschka, ed., 15 June 1998. This W3C Recommendation is http://www.w3.org/TR/1998/REC-smil-19980615.

[SVG]
"Scalable Vector Graphics (SVG) 1.0 Specification", J. Ferraiolo, ed., 2 August 2000. This W3C Candidate Recommendation is http://www.w3.org/TR/2000/CR-SVG-20000802/.

[UAAG10-CHECKLIST]
An appendix to this document lists all of the checkpoints, sorted by priority. The checklist is available in either tabular form or list form.

[UAAG10-TECHS]
"Techniques for User Agent Accessibility Guidelines 1.0", J. Gunderson, I. Jacobs, eds. The latest draft of the techniques document is available at http://www.w3.org/TR/UAAG10-TECHS/.

[W3CPROCESS]
World Wide Web Consortium Process Document, I. Jacobs ed. The 11 November 1999 version of the Process Document is http://www.w3.org/Consortium/Process/Process-19991111/.

[WAIRESOURCES]
A page of *Web Accessibility Initiative Resources* including links to educational materials, other WAI accessibility guidelines, information about accessibility policies, translations of WAI materials, lists of specialized user agents, and more.

[WCAG10]
"Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds., 5 May 1999. This W3C Recommendation is

http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505.

[WCAG10-TECHS]

"Techniques for Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. This W3C Note is http://www.w3.org/TR/1999/WAI-WEBCONTENT-TECHS-19990505.

[WEBCHAR]

"Web Characterization Terminology and Definitions Sheet", B. Lavoie, H. F. Nielsen, eds., 24 May 1999. This is a W3C Working Draft that defines some terms to establish a common understanding about key Web concepts. This W3C Working Draft is http://www.w3.org/1999/05/WCA-terms/01.

[XHTML10]

"XHTML[tm] 1.0: The Extensible HyperText Markup Language", S. Pemberton, et al., 26 January 2000. This W3C Recommendation is http://www.w3.org/TR/2000/REC-xhtml1-20000126.

[XML]

"Extensible Markup Language (XML) 1.0", T. Bray, J. Paoli, C.M. Sperberg-McQueen, eds., 10 February 1998. This W3C Recommendation is http://www.w3.org/TR/1998/REC-xml-19980210.

6. Acknowledgments

The active participants of the User Agent Accessibility Guidelines Working Group who authored this document were: James Allan, Denis Anson (College Misericordia), Kitch Barnicle, Harvey Bingham, Dick Brown (Microsoft), Al Gilman, Jon Gunderson (Chair of the Working Group, University of Illinois, Urbana-Champaign), Eric Hansen (Educational Testing Service), Ian Jacobs (Team Contact, W3C), Marja-Riitta Koivunen, Tim Lacy (Microsoft), Charles McCathieNevile (W3C), Mark Novak, David Poehlman, Mickey Quenzer (isSound), Gregory Rosmaita (Visually Impaired Computer Users Group of New York City), Madeleine Rothberg, and Rich Schwerdtfeger.

Many thanks to the following people who have contributed through review and past participation in the Working Group: Paul Adelson, Olivier Borius, Judy Brewer, Bryan Campbell, Kevin Carey, Tantek Çelik, Wendy Chisholm, David Clark, Chetz Colwell, Wilson Craig, Nir Dagan, Daniel Dardailler, B. K. Delong, Neal Ewers, Geoff Freed, John Gardner, Larry Goldberg, Glen Gordon, John Grotting, Markku Hakkinen, Earle Harrison, Chris Hasser, Kathy Hewitt, Philipp Hoschka, Masayasu Ishikawa, Phill Jenkins, Earl Johnson, Jan Kärrman (for help with html2ps), Leonard Kasday, George Kerscher, Peter Korn, Josh Krieger, Catherine Laws, Greg Lowney, Susan Lesch, Scott Luebking, William Loughborough, Napoleon Maou, Peter Meijer, Karen Moses, Masafumi Nakane, Charles Oppermann, Mike Paciello, David Pawson, Michael Pederson, Helen Petrie, Michael Pieper, Jan Richards, Hans Riesebos, Joe Roeder, Lakespur L. Roca, Lloyd Rutledge, Liam Quinn, T.V. Raman, Robert Savellis, 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.