User Agent Accessibility Guidelines 1.0

W3C Working Draft 26 January 2001

This version:
http://www.w3.org/WAI/UA/WD-UAAG10-20010126
(plain text, gzip PostScript, gzip PDF, gzip tar file of HTML, zip archive of HTML)
Latest version:
http://www.w3.org/WAI/UA/UAAG10
Previous version:
http://www.w3.org/WAI/UA/WD-UAAG10-20010116

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

Copyright ©1999 - 2001 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 (visual, hearing, physical, and cognitive). User agents include HTML browsers and other types of software that retrieve and render 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 not addressed directly by this document (e.g., technologies 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 version of the document takes into account Working Group resolutions through the 25 January 2001 teleconference to last call issues. This document includes some changes not yet agreed on by the Working Group, and does not yet address all issues raised during last call.

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 Relationship to WAI accessibility guidelines 5
1.2 Target user agents 6
1.3 Known limitations of this document 7
1.4 Relationship to general software design guidelines 8
2. The user agent accessibility guidelines 9
1. Support input and output device-independence. 10
2. Ensure user access to all content. 11
3. Allow configuration not to render some content that may reduce accessibility. 13
4. Ensure user control of styles. 15
5. Observe operating environment conventions and standard interfaces. 19
6. Implement specifications that benefit accessibility. 22
7. Provide navigation mechanisms. 23
8. Orient the user. 25
9. Allow configuration and customization. 28
10. Provide accessible product documentation and help. 29
3. Conformance 30
4. Glossary 36
5. References 51
5.1 Normative references 51
5.2 Informative references 51
6. Acknowledgments 54

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.

The Web Accessibility Initiative provides other resources and educational materials to promote Web accessibility. Resources include information about accessibility policies, links to translations of WAI materials into languages other than English, information about specialized user agents and other tools, accessibility training resources, and more.

1. Introduction

This document specifies requirements that user agent developers must satisfy to lower barriers to accessibility. This introduction (section 1) provides context for understanding the guidelines listed in section 2 . Section 1 explains the relationship of this document to other accessibility guidelines published by the Web Accessibility Initiative, which user agents are expected to conform , known limitations of this document, and the relationship of this document to other software design guidelines. 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 Relationship to WAI accessibility guidelines

"User Agent Accessibility Guidelines 1.0" (UAAG 1.0) is part of a series of accessibility guidelines published by the Web Accessibility Initiative (WAI). The documents in this series reflect an accessibility model in which Web content authors, format designers, and software developers have roles in ensuring that users with disabilities have access to the Web. These roles intersect as follows:

- Protocol (e.g., HTTP) and content format (e.g., HTML, XML, SVG, etc.) specifications allow communication on the Web. These specifications includes features that authors may use to create accessible content, and features that user agents must support through an accessible user interface. Essentially, authors must provide accessible alternatives to inaccessible content, and user agents must allow users to access these alternatives.
- Authors make use of the accessibility features of different format specifications, use markup appropriately, write in clear and simple language, organize a Web site consistently, etc. The "Web Content Accessibility Guidelines 1.0" *[WCAG10]* explains the responsibilities of authors in meeting the needs of users with disabilities. In User Agent Accessibility Guidelines 1.0, WCAG 1.0 is considered the reference for what defines accessible Web content. The "Authoring Tool Accessibility Guidelines 1.0" *[ATAG10]* explains the responsibilities of authoring tool developers. An accessible authoring tool facilitates the creation of accessible Web content and may be operated by users with disabilities.
- User agent developers design software that conforms to specifications (including implementation of their accessibility features), provides an accessible user interface, accessible documentation, and communicates with other software (notably assistive technologies).

This document explains the responsibilities of user agents in meeting the needs of users with disabilities. The requirements of this document interact with those of the "Web Content Accessibility Guidelines 1.0" *[WCAG10]* in a number of ways:

- UAAG 1.0 checkpoint 6.1 requires implementation of the accessibility features of all implemented specifications. Features are those identified as such and those that satisfy all of the requirements of WCAG 1.0 *[WCAG10]* .

- UAAG 1.0 checkpoint 10.1 requires conformance to WCAG 1.0 for user agent documentation.
- UAAG 1.0 also incorporates some terms and concepts from WCAG 1.0, a natural consequence of fact that the documents were designed to complement one another.

Formats, authors, and designers all have limitations. No format allows authors to encode all of their knowledge in a way that a user agent can recognize . A format may lack features required for accessibility. An author may not make use of the accessibility features of a format or may misuse a format (which can cause problems for user agents). A user agent designer may not implement a format specification correctly or completely. Some requirements of this document take these limitations into account.

- UAAG 1.0 includes requirements to satisfy the expectations set by WCAG 1.0 "until user agent" clauses. These clauses make additional requirements of authors in order to compensate for some limitations of deployed user agents.
- UAAG 1.0 includes several repair requirements (e.g., checkpoints checkpoint 2.6 and checkpoint 2.8) for cases where content does not conform to WCAG 1.0. Furthermore, some requirements in this document 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).
- Except for the indicated repair checkpoints, UAAG 1.0 only requires user agents to handle what may be recognized through protocols and formats. For example, user agents are not expected to recognize that the author has used "clear and simple" language to express ideas. Please see the section on checkpoint applicability for more information about what the user agent is expected to recognize.

1.2 Target user agents

This document was designed specifically to improve the accessibility of mainstream user agents with multimedia capabilities for users with one or more disabilities (including visual, hearing, physical, and cognitive). 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 enable access through its own user interface and through other internal facilities, including its ability to communicate with other technologies (especially assistive technologies). Technologies not addressed directly by 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 allows 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) or input modalities (such as mouse or voice). See the section on conformance for more information.

1.3 Known limitations of this document

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 does not include requirements to meet all known accessibility needs. Some known limitations of this document include the following:

- Braille. This document does not address braille rendering.
- Synthesized speech. This document includes only three checkpoints related to synthesized speech (checkpoints 4.11, 4.12 and 4.13).
- Size and color of non-text content. 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 and 4.3). This document does not in general address control of the size and color of visually rendered non-text content . **Note:** Resizing capabilities may be required for conformance to other specifications (e.g., SVG *[SVG]*).
- Input modalities. This document only includes requirements for keyboard, pointing device, and voice input modalities. 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.
- Time. This document includes requirements for control of some time parameters (including checkpoint 2.2, checkpoint 4.4, checkpoint 4.5, and checkpoint 4.11). The requirements are for time parameters that the user agent recognizes and controls. This document does not include requirements for control of server-side time parameters.
- Security. This document does not address security issues that may arise as a result of these requirements. For instance, requirements that software be able to read and write content and user interface information through APIs raise security issues. See the section on restricted functionality and conformance .
- Intellectual property. This document does not address intellectual property issues that may arise as a result of these requirements.

1.4 Relationship to general software design guidelines

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 (see checkpoint 5.13). Involving people with disabilities in the design and testing of software will generally improve the accessibility of the software.

Installation is an important aspect of both accessibility and general software usability. On platforms where a user can install a user agent, the installation (and update) procedures need to be accessible. This document does not include a checkpoint requiring that installation procedures be accessible. Since this document considers installation to be part of software usage, the different aspects of installation (user interface, documentation, operating environment conventions, etc.) are already covered by the complete set of checkpoints.

Benefits of accessible user agent design

Many users without disabilities are likely to benefit from the requirements developed to benefit users with disabilities. For example, users without disabilities:

- 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.
- 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 to benefit from the same features that provide access to people who cannot use a mouse or keyboard due to a visual, hearing, or physical disability.
- 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 is expected to 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.

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 (i.e., the "subject of the claim) 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.
- Content type labels (zero or more), which may be used in a in conformance claim.
- Informative notes about the checkpoint. These notes do **not** state requirements that must be satisfied as part of conformance; they are informative only. They are meant to clarify the scope of the checkpoint through further description, examples, cross references, and commentary. **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 is intended to express one or more minimal requirements clearly, so that someone evaluating a user agent may verify that it satisfies the requirements. Both this document and "Techniques for User Agent Accessibility Guidelines 1.0" *[UAAG10-TECHS]* suggest techniques to help user agent developers meet or surpass the minimal requirements. **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. Some checkpoints (e.g., those requiring developers to follow conventions or implement specifications defined

outside this document) are inherently more subject to interpretation than others.

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 need to ensure redundancy in the user interface . The user has to 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 also requires access to the full benefit of Web content through each of at least three modalities -- visually-displayed text, synthesized speech, and braille. Text content has the accessibility advantage of being available to people who use graphical displays, speech synthesizers, and braille displays.

People who cannot or do not use a mouse have to be able to operate the user interface with the keyboard, through voice input, a head wand, touch screen, or other device. Keyboard operation (using as few keystrokes as possible) 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, developers are expected to use each API appropriately. Developers should not in general, for example, pre-rasterize text (i.e., turn it into a bitmap for rendering

rather using the operating environment's text drawing facilities) since doing so may prevent assistive technologies from being able to render the text as speech or braille.

Checkpoints for user interface accessibility:

1.1 Ensure that the user can operate the user agent fully through keyboard input alone, pointing device input alone, and voice input alone. [Priority 1]

> **Note:** For example, ensure that through all three input modalities, the user can interact with active elements , select content, navigate viewports, configure the user agent, access documentation, install the user agent, etc. A user agent may claim conformance to this document without satisfying the pointing device and voice portions of this checkpoint. See the section on input modality labels .

1.2 Ensure that the user can interact with all active elements through keyboard input alone, pointing device input alone, and voice input alone. [Priority 1]

> **Note:** A user agent may claim conformance to this document without satisfying the pointing device and voice portions of this checkpoint. See the section on input modality labels . This checkpoint is an important special case of checkpoint 1.1.

1.3 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 . See also checkpoint 5.5.

Guideline 2. Ensure user access to all content.

Ensure that users have access to all content, notably recognized 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 should be 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. As part of meeting this requirement, provide a view (e.g., a document source view) of the text portions of content. This is only required for formats defined by specifications that the user agent implements . [Priority 1]

> **Note:** In general, user agents render content through the user interface according to the instructions of implemented specifications (see checkpoint 6.2). However, some users may not be able to access this content, even when the user agent renders it according to specification (e.g., for scripts or style sheets). Therefore, this checkpoint includes the requirement that user agents make available raw text content (including attributes, style sheets, etc.) through the user interface. Although this view (which may be a document source view) is an important part of a solution for providing access to content, it is not a sufficient solution on its own for all content. This checkpoint does not require that all content be available through every viewport . See guideline 5 for more information about programmatic access to content.

2.2 For content that requires user input within a time interval controlled by the user agent, allow configuration to make the time interval "infinite" (i.e., pause automatically at the beginning of each time interval where user input is required, and resume automatically after the user has explicitly completed input). [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. In SMIL 1.0 *[SMIL]* , for example, the "`begin`", "`end`", and "`dur`" attributes synchronize presentation components. This checkpoint does not apply when the user agent cannot recognize the time interval in the presentation format, or when the user agent cannot control the timing (e.g., because it is controlled by the server).

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 tracks and visual tracks . [Priority 1]

> Content type labels : Video , Audio .

2.5 Respect synchronization cues during rendering. [Priority 1]

> Content type labels : Video , Audio .

2.6 Allow configuration to generate repair text when the user agent recognizes that the author has failed to provide a required equivalent . If the content missing an equivalent is included by URI reference, base the repair text on the URI reference and content type. Otherwise, base the repair text on element type information. [Priority 2]

> **Note:** 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 . See also checkpoint 2.7.

2.7 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 2.6. [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. See also checkpoint 2.6.

2.8 Allow the user to configure the user agent not to render content in unsupported natural languages . Indicate to the user in context that author-supplied content has not been rendered. [Priority 3]

> **Note:** For example, use a text substitute or accessible graphical icon to indicate that content in a particular language has not been rendered. This checkpoint does not require the user agent to allow different configurations for different natural languages.

Guideline 3. Allow configuration 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 text 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 environment 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]

Content type labels : Image .

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 (see 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 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]

Content type labels : Image , Video , Audio .

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, the user agent is not required to retrieve the audio, video, or animated image from the Web until requested by the user. See also checkpoint 4.5, checkpoint 4.9 and checkpoint 4.10.

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

Content type labels : VisualText .

Note: This checkpoint does not apply for blinking and animation effects that are caused by mechanisms that the user agent cannot recognize .

3.4 Allow the user to configure the user agent not to execute any excecutable content (e.g., scripts and applets). In this configuration, provide an option to alert the user when executable content is available (but has not been executed). [Priority 1]
3.5 Allow configuration so that client-side content refreshes (i.e., those initiated by the user agent, not the server) do not change content except on explicit user request . Allow the user to request the new content on demand (e.g., by following a link or confirming a prompt). Alert the user, according to the schedule specified by the author, whenever fresh content is available (to be obtained on explicit user request). [Priority 1]
3.6 Allow configuration so that a "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 on demand (e.g., by following a link or confirming a prompt). The user agent is not required to provide these functionalities for client-side redirects that occur instantaneously (i.e., when there is no delay before the new content is retrieved). [Priority 2]
3.7 Allow the user to configure the user agent not to render images. In this configuration, provide an option to render a placeholder in context for each unrendered image. When placeholders are rendered, allow the user to activate each placeholder individually and replace it with the original author-supplied content. [Priority 2]

Content type labels : Image .

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 (see 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. Users need to be alerted to changes to content or viewports that the users did not initiate (e.g., when a viewport opens automatically).

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

Checkpoints for visually rendered text (content accessibility):

4.1 Allow the user to configure globally and control the reference size of rendered text, with an option to override reference sizes specified by the author or user agent defaults. Allow the user to choose from among the full range of font sizes supported by the operating environment . [Priority 1]

Content type labels : VisualText .

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 globally the font family of all rendered text, with an option to override font families specified by the author or user agent defaults. Allow the user to choose from among the full range of font families supported by the operating environment . [Priority 1]

Content type labels : VisualText .

Note: For example, allow the user to specify that all text must be rendered in a particular sans-serif font family. For text that cannot be rendered properly using the user's selected font family, the user agent may select an alternative font family.

4.3 Allow the user to configure globally the foreground and background color of all rendered text, with an option to override foreground and background colors specified by the author or user agent defaults. Allow the user to choose from among the full range of colors supported by the operating environment . [Priority 1]

Content type labels : ColorText .

Note: User configuration of foreground and background colors may result in the inability to distinguish ordinary text from selected text, focused text, etc. See checkpoint 8.2 for more information about highlight styles.

Checkpoints for multimedia presentations and other presentations that change continuously over time (content accessibility):

4.4 Allow the user to slow the presentation rate of audio, video and animations. 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% and 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 . The user agent is not required to satisfy this checkpoint for audio, video and animations whose recognized role is to create a purely stylistic effect. [Priority 1]

Content type labels : Animation , Video , Audio .

Note: Purely stylistic effects include background sounds, decorative animated images, and effects caused by style sheets. The style exception of this checkpoint is based on the assumption that authors have satisfied the requirements of the "Web Content Accessibility Guidelines 1.0" *[WCAG10]* not to convey information through style alone (e.g., through color alone or style sheets alone). See also checkpoint 4.7.

See also checkpoint 2.5.

4.5 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. The user agent is not required to satisfy this checkpoint for audio, video and animations whose recognized role is to create a purely stylistic effect. [Priority 1]

Content type labels : Animation , Video , Audio .

Note: See checkpoint 4.4 for more information about the exception for purely stylistic effects. 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 2.5. See also checkpoint 3.2.

4.6 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]

4.7 Allow the user to slow the presentation rate of audio, video and animations not covered by checkpoint 4.4. The same speed percentage requirements of checkpoint 4.4 apply. [Priority 2]

Content type labels : Animation , Video , Audio .

Note: User agents automatically satisfy this checkpoint if they satisfy checkpoint 4.4 for every audio, video, and animation.

4.8 Allow the user to stop, pause, resume, fast advance, and fast reverse audio, video, and animations not covered by checkpoint 4.5. [Priority 2]

Content type labels : Animation , Video , Audio .

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

Checkpoints for audio volume control (content accessibility):

4.9 Allow the user to configure globally and control the volume of all audio, with an option to override audio volumes specified by the author or user agent defaults. The user must be able to choose zero volume (i.e., silent). [Priority 1]

Content type labels : Audio .

Note: User agents should allow configuration and control of volume through available operating environment-level controls.

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

Content type labels : Audio .

Note: Sounds that play at different times are distinguishable and therefore independent control of their volumes is not part of this checkpoint (volume control per checkpoint 4.9 suffices). The user agent may satisfy this checkpoint by allowing the user to control independently the volumes of all distinct audio sources. The user control required by this checkpoint includes the ability to override author-specified volumes for the relevant sources of audio. See also checkpoint 4.12.

Checkpoints for synthesized speech (content accessibility):

4.11 Allow the user to configure and control synthesized speech playback rate according to the full range offered by the speech synthesizer. [Priority 1]

Content type labels : Speech .

Note: The range of playback rates offered by the speech synthesizer may depend on the natural language.

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

Content type labels : Speech .

Note: The user control required by this checkpoint includes the ability to override author-specified speech volume. See also checkpoint 4.10.

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

Content type labels : Speech .

Note: Many speech synthesizers allow users to choose from among preset options that control different voice parameters (gender, pitch range, stress, richness, etc.) as a group. When using these synthesizers, allow the user to choose from among the full range of preset options (e.g., "adult male voice", "female child voice", "robot voice", etc.). Ranges of values for these characteristics may vary among speech synthesizers. For information about these synthesized speech characteristics, please refer to descriptions in section 19.8 of Cascading Style Sheets Level 2 *[CSS2]* .

Checkpoints for user interface accessibility:

4.14 For user agents that support style sheets, allow the user to choose 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.

4.15 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]

4.16 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.

4.17 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 with which it overlaps. [Priority 2]

4.18 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 on demand (e.g., by following a link or confirming a prompt). Allow the user to close viewports. If a viewport (e.g., a frame set) contains other viewports, these requirements only apply to the outermost container viewport. [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. See also checkpoint 4.15 (for control over changes of focus when a viewport opens) and checkpoint 5.5.

4.19 Allow configuration so the user is prompted to confirm any viewport that closes without explicit user request. [Priority 3]

Guideline 5. Observe operating environment conventions and standard interfaces.

Communicate with other software (e.g., assistive technologies, the operating environment, plug-ins). Observe operating environment 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 operating environment conventions for user agent user interface design, documentation , and installation.
- incorporating operating environment-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 operating environment 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 following modules of the W3C Document Object Model DOM Level 2 Core Specification *[DOM2CORE]* and exporting the interfaces they define: (1) the Core module for HTML; (2) the Core and XML modules for XML. [Priority 1]

> **Note:** Please refer to the "Document Object Model (DOM) Level 2 Core Specification" *[DOM2CORE]* for information about HTML and XML versions covered.

5.2 If the user can modify HTML and XML content through the user interface , provide the same functionality programmatically by conforming to the following modules of the W3C Document Object Model DOM Level 2 Core Specification *[DOM2CORE]* and exporting the interfaces they define: (1) the Core module for HTML; (2) the Core and XML modules for XML. [Priority 1]

> **Note:** For example, if the user interface allows users to complete HTML forms, this must also be possible through the required DOM APIs . Please refer to the "Document Object Model (DOM) Level 2 Core Specification" *[DOM2CORE]* for information about HTML and XML versions covered.

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 environment). [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:** For instance, when user interaction in one frame causes automatic changes to content in another, provide programmatic alert through standard APIs. Use the standard APIs required by the checkpoints of guideline 5.

5.6 Implement standard accessibility APIs (e.g., of the operating environment). Where these APIs do not enable the user agent to satisfy the requirements of this document, use the standard input and output APIs of the operating environment. [Priority 1]

Note: Accessibility APIs enable assistive technologies to monitor input and output events. As part of satisfying this checkpoint, the user agent needs to ensure that text content is available as text through these APIs (and not, for example, as a series of strokes drawn on the screen).

5.7 Implement the operating environment's standard APIs for the keyboard . [Priority 1]

Note: An operating environment may define more than one standard API for the keyboard. For instance, for Japanese and Chinese, input may be processed in two stages, with an API for each. This checkpoint is an important special case of checkpoint 1.1. See also checkpoint 9.4.

5.8 For an API implemented to satisfy requirements of this document, support the character encodings required for that API. [Priority 1]

Note: Support for character encodings is important so that text is not "broken" when communicated to assistive technologies. For example, the DOM Level 2 Core Specification *[DOM2CORE]* , section 1.1.5 requires that the `DOMString` type be encoded using UTF-16. This checkpoint is an important special case of the other API requirements of this document.

5.9 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.

5.10 For user agents that implement Cascading Style Sheets (CSS), provide programmatic access to those style sheets by conforming to the CSS module of the W3C Document Object Model (DOM) Level 2 Style Specification *[DOM2STYLE]* and exporting the interfaces it defines. [Priority 2]

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

Checkpoints for user interface accessibility:

5.11 Follow operating environment conventions that benefit accessibility when implementing the selection , content focus , and user interface focus . [Priority 1]

Note: This checkpoint is an important special case of checkpoint 5.13. See also checkpoint 7.1.

5.12 Ensure that default input configurations do not interfere with operating environment accessibility conventions. [Priority 1]

Note: In particular, default configurations should not interfere with operating conventions for keyboard accessibility. Information about operating environment accessibility conventions is available in the Techniques document *[UAAG10-TECHS]* . See also checkpoint 9.4.

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

Note: Operating environment 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]* . See also checkpoint 5.12.

5.14 Follow operating environment conventions to indicate the input configuration . [Priority 2]

Note: For example, in some operating environments, 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 5.13. See also checkpoint 9.4.

Guideline 6. Implement specifications that benefit 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 benefits 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 recommends conformance to 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 these specifications (public Working Drafts, Candidate Recommendations, and Proposed Recommendations). 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 either (1) W3C Recommendations when they are available and appropriate for a task, or (2) non-W3C specifications that enable the creation of content that conforms to the Web Content Accessibility Guidelines 1.0 *[WCAG10]* at any conformance level. [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]* . A specification is considered "available" if it is published (e.g., as a W3C Recommendation) in time for integration into a user agent's development cycle.

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, see 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:** The set of navigable elements required by this checkpoint must include active elements and may include non-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]

> **Note:** The set of navigable elements required by this checkpoint must include active elements and must not include non-active elements.

7.5 Allow the user to search within rendered text content for a sequence of characters from the document character set . Allow the user to start a forward search (in document order) from any selected or focused location in content. When there is a match (1) move the viewport so that the matched text content is within it, and (2) allow the user to search for the next instance of the text from the location of the match. Alert the user when there is no match. If the search wraps back to the

beginning of content, alert the user prior to wrapping. Provide a case-insensitive search option for text in scripts (i.e., writing systems) where case is significant. [Priority 2]

> **Note:** The default search starting point should be the beginning of content. Use operating environments conventions for indicating the result of a search (e.g., selection or content focus). A wrapping search is one that restarts automatically at the beginning of content once the end of content has been reached.

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

> **Note:** This specification intentionally does not identify which "important elements" must be navigable as this will vary according to markup language. What constitutes "efficient navigation" may depend on a number of factors as well, including the "shape" of content (e.g., serial navigation of long lists is not efficient) and desired granularity (e.g., among tables, then among the cells of a given table). Refer to the Techniques document *[UAAG10-TECHS]* for information about identifying and navigating important elements. See also checkpoint 7.7.

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. See also 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 purpose of each table and the relationships among the table cells and headers. [Priority 1]

> **Note:** This checkpoint refers only to table information that the user can recognize . 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. See also checkpoint 5.3. This checkpoint is an important special case of checkpoint 2.1.

8.2 Ensure that all of the default highlight styles for the selection , content focus , active elements , recently visited links, and fee links (1) do not rely on color alone, and (2) differ from each other, and not by color alone. [Priority 1]

> **Note:** For instance, by default a graphical user agent may present the selection using color and a dotted outline, the focus using a solid outline, active elements as underlined in blue, recently visited links as dotted underlined in purple, and fee links using a special icon or flag to draw the user's attention.

8.3 Provide a mechanism for highlighting all active elements , recently visited links, and fee links , and allow the user to configure the highlight styles. The highlight mechanism must not rely on color alone. For graphical viewports, if the highlight mechanism involves colors, fonts, or text decorations , allow the user to choose from among the full range of colors, fonts, or text decorations supported by the operating environment . [Priority 2]

> **Note:** Examples of highlight mechanisms include foreground and background color variations, font variations, underlining, distinctive voice pitches, rectangular boxes, etc.

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.). [Priority 2]

> **Note:** This checkpoint is meant to provide the user with a simplified view of content (e.g, a table of contents). What constitutes a label is defined by a markup language specification. For example, in HTML, a heading (`H1-H6`) is a label for the section that follows it, a `CAPTION` is a label for a table, the "`title`" attribute is a label for its element, etc. A label is not required to be text only. For important elements that do not have associated labels, user agents may generate labels for the outline view. For information about what constitutes the set of important structural elements, please see the Note following checkpoint 7.6. By making the outline view navigable, it is possible to satisfy this checkpoint and checkpoint 7.6 together: Allow users to navigate among the important elements of the outline view, and to navigate from a position in the outline view to the corresponding position in a full view of content. See also 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 Provide a mechanism for highlighting the selection and content focus , and allow the user to configure the highlight styles. The highlight mechanism must not rely on color alone. For graphical viewports, if the highlight mechanism involves colors or text decorations , allow the user to choose from among the full range of colors or text decorations supported by the operating environment . [Priority 1]

> **Note:** Examples of highlight mechanisms include foreground and background color variations, underlining, distinctive voice pitches, rectangular boxes, etc. Because the selection and focus change frequently, user agents should not highlight them using mechanisms (e.g., font size variations) that cause content to reflow as this may disorient the user. See also checkpoint 5.11.

8.7 Provide a mechanism for highlighting the current viewport . For graphical viewports, the default highlight mechanism must not rely on color alone. [Priority 1]

> **Note:** This includes highlighting and identifying frames. This checkpoint is an important special case of checkpoint 1.1. See also to checkpoint checkpoint 5.13.

8.8 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.

8.9 Allow configuration so the user is prompted to confirm any payment resulting from activation of a fee link . [Priority 2]

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 . [Priority 1]

> **Note:** To satisfy this checkpoint, the user agent may make available binding information in a centralized fashion (e.g., a list of bindings) or a distributed fashion (e.g., by listing keyboard shortcuts in user interface menus).

9.2 Provide a centralized view of the current author-specified input configuration bindings. [Priority 2]

> **Note:** For example, for HTML documents, provide a view of keyboard bindings specified by the author through the `"accesskey"` attribute. The intent of this checkpoint is to centralize information about author-specified bindings so that the user does not have to read the entire content first to find out what bindings are available. The user agent may satisfy this checkpoint by providing different views for different input modalities (keyboard, pointing device, voice, etc.).

9.3 Allow the user to override any binding that is part of the user agent default input configuration . Allow the user to override any binding in the default keyboard configuration with a binding of a single key and (possibly zero) modifier keys. Allow the user to assign a single key binding (with zero modifier keys) to at least a majority of the functionalities available in the default keyboard configuration. The user agent is not required to allow the user to override standard bindings for the operating environment (e.g., for access to help). [Priority 2]

> **Note:** The override requirement only applies to bindings for the same input modality (i.e., the user must be able to override a keyboard binding with another keyboard binding). This checkpoint does not require single-key bindings for character input, only for the activation of user agent functionalities. See also checkpoint 9.4, checkpoint 9.6, and checkpoint 10.3.

9.4 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]

Note: This checkpoint does not make any requirements about the ease of use of default input configurations, though clearly the default configuration should include single-key bindings and allow easy operation. Ease of use is ensured by the configuration requirements of checkpoint 9.3.

9.5 For the configuration requirements of this document, allow the user to save user preferences in at least one user profile . Allow users to choose 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.

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

Note: This checkpoint is a special case of checkpoint 9.3.

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.

See also checkpoint 5.13.

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 benefit accessibility. [Priority 1]

Note: For example, review the documentation or help system to ensure that it includes information about the functions and capabilities of the user agent that are required by WAI Guidelines.

10.3 Document the default input configuration (e.g., the default keyboard bindings). [Priority 1]
10.4 In a dedicated section of the documentation , describe all features of the user agent that benefit 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 defines what it means to conform to this document and explains how to make a valid conformance claim . Here is a sample claim (expressed in HTML):

On 26 January 2001, 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/WAI/UA/WD-UAAG10-20010126, level Double-A. Unsupported content types: Video, Speech. Unsupported input modalities: Voice. (see section 3.1 of the UAAG 1.0). The list of checkpoints that do not apply is available online.

The terms "must", "should", and "may" (and related terms) are used in this document in accordance with RFC 2119 *[RFC2119]* . This section uses the expression "subject of a claim" to refer to a user agent about which someone wishes to claim some level of conformance to this document. The subject of a claim may be one or more software components (e.g., a browser plus additional software).

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 known limitations of this document .

3.1 Conformance model

There are two ways to conform to this document: unconditionally or conditionally. A user agent conforms unconditionally to this document if it satisfies all of the requirements of all of the checkpoints. Some checkpoints include more than one requirement.

A user agent conforms conditionally if it satisfies the set of requirements that results from following these steps:

1. Choose a conformance level , which establishes a set of requirements.
2. Remove the requirements associated with any unsupported content type labels . In order to conform conditionally, a user agent must satisfy the requirements associated with at least one content type label.
3. Remove the requirements associated with any unsupported input modality label . In order to conform conditionally (or unconditionally) a user agent must be fully operable through the keyboard, and must satisfy the input device requirements of this document for the keyboard.
4. Remove the requirements of any checkpoints that do not apply .

Since these steps produce very different sets of checkpoints for different user agents, a valid conformance claim must indicate which requirements the subject of the claim does not satisfy.

Note: The checklist *[UAAG10-CHECKLIST]* may be used when evaluating a user agent for conformance.

Conformance levels

Each conformance level defines a set of requirements, based on priority .

- **Conformance Level "A"**: the requirements of all Priority 1 checkpoints.
- **Conformance Level "Double-A"**: the requirements of all Priority 1 and 2 checkpoints.
- **Conformance Level "Triple-A"**: the requirements of all Priority 1, 2, and 3 checkpoints.

Note: Conformance levels are spelled out in text (e.g., "Double-A" rather than "AA") so they may be understood when rendered as speech.

Content type labels

Each content type label defines a set of requirements based on support for images, video, animations, visually displayed text (in color), and synthesized speech.

VisualText
: This content type label refers to all of the requirements related to the visual rendering of text for the following checkpoints: 3.3, 4.1, and 4.2.

ColorText
: This content type label refers to all of the requirements related to text foreground and background color for the following checkpoint: 8.3.

Image
: This content type label refers to all of the requirements related to images (including animated images) for the following checkpoints: 3.2, 3.1 and 3.7. To conform, the user agent must implement at least one image format.

Animation
: This content type label refers to all of the requirements related to animations for the following checkpoints: 4.4, 4.5, 4.7, and 4.8. To conform, the user agent

must implement at least one animation format.

Video
: This content type label refers to all of the requirements related to video for the following checkpoints: 2.4, 2.5, 3.2, 4.4, 4.5, 4.7, and 4.8. To conform, the user agent must implement at least one video format.

Audio
: This content type label refers to all of the requirements related to audio for the following checkpoints: 2.4, 2.5, 3.2, 4.4, 4.5, 4.7, 4.8, 4.9, and 4.10. To conform, the user agent must implement at least one audio format.

Speech
: This content type label refers to all of the requirements related to synthesized speech for the following checkpoints: 4.11, 4.12 and 4.13. To conform, the user agent must support synthesized 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 see the requirements of checkpoint 6.2.

Input modality labels

Each input modality label defines a set of requirements based on support for pointing device and voice input.

Pointer
: This input modality label refers to all of the input device requirements of this document, applied to pointing device input.

Voice
: This input modality label refers to all of the input device requirements of this document, applied to voice input.

Note: Developers are encouraged to design user agents that are at least partially operable through all three input modalities.

3.2 Checkpoint applicability

A checkpoint (or portion of a checkpoint) applies unless any one of the following conditions is met:

1. The checkpoint makes requirements for graphical user interfaces or graphical viewports and the subject of the claim only has audio or tactile user interfaces or viewports.
2. The checkpoint refers to a role of content (e.g., transcript, captions, text equivalent , fee link , synchronization cue, client-side redirect, purpose of a table, etc.) that the subject of the claim cannot recognize because of how the content has been encoded in a particular format. 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.

3. The checkpoint requires control of a content property that the subject cannot recognize because of how the content has been encoded in a particular format. Some examples of this include:
   - captioning information that is "burned" into a video presentation and cannot be recognized as captions in the presentation format;
   - streamed content that cannot be fast advanced or reversed,
   - information or relationships encoded in scripts in a manner that cannot be recognized. For instance, the requirements of checkpoint 3.3 would not apply for animation effects caused by scripts. Input configuration bindings (refer to guideline 9) created through scripts in a manner that the user agent cannot recognize do not apply.

3.3 Well-formed conformance claims

A claim is well-formed if meets the following conditions.

Condition 1: The claim must include the following information:

1. The date of the claim.
2. The guidelines title/version: "User Agent Accessibility Guidelines 1.0".
3. The URI of the guidelines: http://www.w3.org/WAI/UA/WD-UAAG10-20010126.
4. The conformance level satisfied: "A", "Double-A", or "Triple-A".
5. Information about the subject. 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 environment (e.g., operating system) name and version number.

Condition 2: The claim must include the following information if the user agent conforms conditionally :

1. Content type labels . Each content type label is an assertion that the user agent does **not** satisfy the requirements associated with the label. A well-formed conformance claim must not include all of the content type labels (because the user agent must support at least one of the content types).
2. Input modality labels . Each input modality label is an assertion that the user agent does **not** satisfy the requirements associated with the label.
3. A list of checkpoints that the claim asserts do **not** apply . A well-formed claim should include rationale for why a checkpoint doesn't apply.

Condition 3: If the claim is on the Web, it must conform to the "Web Content Accessibility Guidelines 1.0" *[WCAG10]* , level A.

There is no restriction on the format used to make a well-formed claim. For instance, the claim may be marked up using HTML (see sample claim), or expressed in the Resource Description Framework (RDF) *[RDF10]* .

3.4 Validity of a claim

A conformance claim is valid if the following conditions are met:

1. The claim is well-formed .
2. It is verified that the user agent satisfies all requirements not exempted from the claim by the allowable mechanisms (i.e., conformance levels , content type labels , input modality labels , and applicability).

It is not currently possible to validate a claim entirely automatically.

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

Note: The subject of the claim may consist of more than one software component, and taken together they must satisfy all all requirements that are not excluded through the claim. This includes assistive technologies and operating environment features 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 of the claim as a whole makes *all content* available through those APIs.

Note: Ideally, the standard (or, default) user agent installation procedure should provide and install all components that are part of a conformance claim. This is because, the more software components the user must install in order to construct a conforming user agent, the higher the risk of failure. Failure may be due to inaccessible mechanisms for downloading and installing plug-ins, or lack of installation access privileges for a computer in a public space.

Use of operating environment features as part of conformance

To satisfy the requirements of this document, developers are encouraged to adopt operating environment conventions and features that benefit accessibility. When an operating environment feature (e.g., the operating system’s audio control feature) is adopted to satisfy the requirements of this document, it is part of the subject of the claim .

Developers may provide access through the user agent's user interface to operating environment features adopted to satisfy the requirements of this document. For example, if the user agent adopts the operating system's audio control feature to satisfy checkpoint 4.9, the user agent may (but is not required to) include those controls in its own user interface.

Restricted functionality and conformance

There may be scenarios where a content provider wishes to limit the user's full access to content. For instance, a content provider may wish to limit access to content through an API (e.g., to protect intellectual property rights, or for security reasons), or to provide a "read-only" view (allowing no user interaction). A valid conformance claim remains valid even when the functionality of a conforming user agent is restricted in a particular setting. The validity of a conformance claim will be seriously jeopardized if a user agent does not meet the requirements of this document for general-purpose content.

Note: The User Agent Accessibility Guidelines Working Group recognizes that further work is necessary in the area of digital rights management as it relates to accessibility.

3.5 Responsibility for claims

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 paper product documentation).

Claimants (or relevant assuring parties) are solely responsible for the validity of their claims, keeping claims up to date, and proper use of the conformance icons . As of the publication of this document, W3C does not act as an assuring party, but it may do so in the future, or it may 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. Claimants are encouraged to claim conformance to the most recent User Agent Accessibility Guidelines Recommendation available.

3.6 Conformance icons

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

Draft Note: *In the event this document becomes a W3C Recommendation this document will link to the W3C Web site for additional information about the icons and how to use them.*

4. Glossary

Active element

An active element is a piece of content with associated behaviors, that the user may trigger (or, "***activate***") either through the user interface or through an API .

Content always determines what constitutes an active element. For instance, the HTML 4 *[HTML4]* specification defines a number of active elements: 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). The role of an element as an active element is subject to applicability

The state of the user's interaction with that content may limit which elements are active. For instance, an element may be "deactivated" by a script as the result of the user's interaction with the content. Or, an element may only be active during a given time period (e.g., during part of a SMIL 1.0 *[SMIL]* presentation). Or, the user may be viewing content in "read-only" mode, which may deactivate some elements.

The user may interact with content without necessarily activating active elements. For example, selecting an element's text and copying it to the clipboard is clearly user interaction but does not make that element an active element. (The element may *also* be an active element, but only by virtue of how the author has encoded it, not by virtue of the selection functionality provided by the user agent.)

The consequence of triggering an active element 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). See also the definition of event handler .

Most operating environments use the content focus to indicate which active element will be triggered on user demand.

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. See checkpoint 1.3 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 environment vendor, de facto standards, etc.). Implementing APIs that are

independent of a particular operating environment (as are 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 environment may reduce implementation costs for assistive technology developers who wish to interoperate with more than one piece of software running on that operating environment.

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. Input and output APIs also include more abstract communication interfaces than those specified by device APIs. A "standard input/output API" is one that is expected to be implemented by software running on a particular operating environment. 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-only presentation

An audio-only presentation is content 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 object is content rendered as sound through an audio viewport . An audio track is an audio object that is intended as a whole or partial presentation. An audio track may, but is not required to, 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, or from its associated style sheets, or that are generated by the server.

Captions

Captions (sometimes called "closed captions") are text transcripts that are synchronized with other audio tracks 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.

Character encoding

A "character encoding" is a mapping from a character set definition to the actual code units used to represent the data. Please refer to the Unicode specification *[UNICODE]* for more information about character encodings. Refer to "Character Model for the World Wide Web" *[CHARMOD]* for additional information about characters and character encodings.

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). See also 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, the verbs "to control" and "to 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 limitations of the operating environment or hardware.

Both configuration and control may apply at different "levels": across Web resources (i.e., at the user agent level, or inherited from the operating environment), to the entirety of a Web resource, or to components of a Web resource (e.g., on a per-element basis). In this document, the term ***global configuration*** is used to emphasize when a configuration must apply across Web resources. 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. On the other hand, 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 having to reconfigure the user agent. 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.

User agents may allow users to choose configurations based on various parameters, such as hardware capabilities, natural language, etc.

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

Content

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

1. It is used to mean the document object as a whole or in parts.
2. It is used to mean 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. It 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.

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 Document Object Model (DOM) Level 2 Specifications (*[DOM2CORE]* and *[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.

Document character set

A document character set (an concept taken from SGML) is a sequence of abstract characters that may appear in Web content represented in a particular format (such as HTML, XML, etc.). A document character set consists of:

- a "repertoire", A set of abstract characters, such as the Latin letter "A", the Cyrillic letter "I", the Chinese character meaning "water", etc.
- Code positions: A set of integer references to characters in the repertoire.

For instance, the character set required by the HTML 4 specification *[HTML4]* is defined in the Unicode specification *[UNICODE]* . Refer to "Character Model for the World Wide Web" *[CHARMOD]* for more information about document character sets.

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 environment , 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.

Fee link

For the purpose of this document, the term "fee link" refers to a link that when activated, debits the user's electronic "wallet" (generally, a "micropayment"). The link's role as a fee link must be identified through markup in a manner that the user agent can recognize . This definition of fee link excludes payment mechanisms (e.g., some form-based credit card transactions) that cannot be recognized by the user agent as causing payments. For more information about fee links, refer to "Common Markup for micropayment per-fee-links" *[MICROPAYMENT]* .

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 (e.g., a link or radio button). 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 . See also 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]). Input configurations may be affected by author-specified bindings (e.g., through the "accesskey" attribute of HTML 4 *[HTML4]*).

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). See also the definition of script .

Operating environment

The term "operating environment" refers to the environment that governs the user agent's operation, whether it is an operating system or a programming language environment such as Java.

Placeholder

A placeholder is content generated by the user agent to replace author-supplied content. A placeholder may be generated as the result of a user preference (e.g., to not render images) or as repair content (e.g., when an image cannot be found). Placeholders can be any type of content, including text and images. This

document does not require user agents to include placeholders in the document object . A placeholder inserted in the document object should conform to the Web Content Accessibility Guidelines 1.0 *[WCAG10]* . If a placeholder is not part of the document object, it is part of the user interface only (and subject, for example, to checkpoint 1.3).

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.

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, natural language preferences, etc. In operating environments with distinct user accounts, profiles enable users to reconfigure software quickly when they log on, and profiles 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 loudspeakers, on a braille display, on a mobile device, etc. Style information (e.g., fonts, colors, speech prosody, 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

Authors encode information in markup languages, style sheet languages, scripting languages, protocols, etc. When the information is encoded in a manner that allows the user agent to process it with certainty, the user agent can "recognize" the information. For instance, HTML allows authors to specify a heading with the H1 element, so a user agent that implements HTML can recognize that content as a heading. If the author creates headings using a visual effect alone (e.g., by increasing the font size), then the author has encoded the heading in a manner that does not allow the user agent to recognize it as a heading.

Some requirements of this document depend on content roles, content relationships, timing relationships, and other information that must be supplied by the author. These requirements only apply when the author has encoded that information in a manner that the user agent can recognize. See the section on conformance for more information about applicability.

In practice, user agents will rely heavily on information that the author has encoded in a markup language or style sheet language. On the other hand, information encoded in a script may not be recognized by the user agent as easily. For instance, a user agent is not expected to recognize that, when executed, a script will calculate a factorial. The user agent will be able to recognize some information in a script by virtue of implementing the scripting language or a known program library (e.g., the user agent is expected to recognize when a script will open a viewport or retrieve a resource from the Web). The Techniques document *[UAAG10-TECHS]* lists some markup known to affect accessibility that user agents can recognize.

Rendered content, ***rendered text***

Rendered content is the part of content capable of being perceived by a user through a given viewport (whether visual, auditory, or tactile). Some rendered content may lie "outside" of a viewport at some times (e.g., when the user can only view a portion of a large document through a small graphical viewport, when audio content has already been played, etc.). By changing the viewport's position, the user can view the remaining rendered content.
Note: In the context of this document, "invisible content" is content that influences graphical rendering of other content but is not rendered itself. Similarly, "silent content" is content that influences audio rendering of other content but is not rendered itself. Neither invisible nor silent content is considered rendered content.

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. "Repair text" means repair content consisting only of text . 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.);

This document does not require user agents to include repair content in the document object . Repair content inserted in the document object should conform to the Web Content Accessibility Guidelines 1.0 *[WCAG10]* . For more information about repair techniques for Web content and software, refer to "Techniques for Authoring Tool Accessibility Guidelines 1.0" *[ATAG10-TECHS]* .

Script

In this document, the term "script" almost always refers to a scripting (programming) language used to create dynamic Web content. However, in checkpoints referring to the written (natural) language of content, the term "script" is used as in Unicode *[UNICODE]* to mean "A collection of symbols used to represent textual information in one or more writing systems."

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 through changes in speech prosody, 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" refers 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 developers 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

In this document, the term "text" used by itself refers to a sequence of characters from a markup language's document character set . 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 .

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., a 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 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 (see checkpoint 1.3).

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 decoration

In this document, a "text decoration" is any stylistic effect that the user agent may apply to visually rendered text that does not affect the layout of the document (i.e., does not require reformatting when applied or removed). Text decoration mechanisms include underline, overline, and strike-through.

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). See also the definitions of captions and collated text transcripts .

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.

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-only presentation

An visual-only presentation is content consisting exclusively of one or more visual tracks presented concurrently or in series. Examples of an visual-only presentation include a silent movie.

Visual track

A visual object is content rendered through a graphical viewport . Visual objects include graphics, text, and visual portions of movies and animations. An 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 loudspeaker, a virtual magnifying glass, etc. A viewport may contain another viewport (e.g., nested frames). User interface controls such as prompts, menus, alerts, etc. are not viewports.

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.

Voice browser

From "Introduction and Overview of W3C Speech Interface Framework" *[VOICEBROWSER]* : "A voice browser is a device (hardware and software) that interprets voice markup languages to generate voice output, interpret voice input, and possibly accept and produce other modalities of input and output."

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.

5.1 Normative references

[DOM2CORE]
"Document Object Model (DOM) Level 2 Core Specification", A. Le Hors, P. Le Hégaret, L. Wood, G. Nicol, J. Robie, M. Champion, S. Byrne, eds., 13 November 2000. This W3C Recommendation is http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113

[DOM2STYLE]
"Document Object Model (DOM) Level 2 Style Specification", V. Apparao, P. Le Hégaret, C. Wilson, eds., 13 November 2000. This W3C Recommendation is http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113.

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

[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.

5.2 Informative references

[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.

[ATAG10-TECHS]
"Techniques for Authoring Tool Accessibility Guidelines 1.0", J. Treviranus, C. McCathieNevile, I. Jacobs, and J. Richards, eds., 4 May 2000. This W3C Note is http://www.w3.org/TR/2000/NOTE-ATAG10-TECHS-20000504/.

[CHARMOD]
"Character Model for the World Wide Web", M. Dürst and F. Yergeau, eds., 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.

[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.

[MICROPAYMENT]
"Common Markup for micropayment per-fee-links", T. Michel, ed., 25 August 1999. This W3C Working Draft is http://www.w3.org/TR/1999/WD-Micropayment-Markup-19990825.

[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.

[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", I. Jacobs, J. Gunderson, E. Hansen, eds. The latest draft of the techniques document is available at http://www.w3.org/WAI/UA/UAAG10-TECHS/.

[UNICODE]
"The Unicode Standard, Version 3.0", The Unicode Consortium, Reading, MA, Addison-Wesley Developers Press, 2000. ISBN 0-201-61633-5. Refer also to http://www.unicode.org/unicode/standard/versions/. For information about character encodings , refer to Unicode Technical Report #17 "Character Encoding Model".

[VOICEBROWSER]
"Voice Browsers: An introduction and glossary for the requirements drafts", M. Robin, J. Larson, 23 December 1999. This document is http://www.w3.org/TR/1999/WD-voice-intro-19991223. This document includes references to additional W3C specifications about voice browser technology.

[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/.

[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, 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.