User Agent Accessibility Guidelines 1.0

W3C Working Draft, 31 August 2001

This version:
http://www.w3.org/WAI/UA/WD-UAAG10-20010831/
(Formats: single HTML, 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-UAAG10-20010731/

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). Furthermore, all users, not just users with disabilities, are expected to find conforming user agents to be more usable.

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.

With the publication of this document, the User Agent Accessibility Guidelines Working Group (UAWG) requests that the W3C Director advance "User Agent Accessibility Guidelines 1.0" to Candidate Recommendation status (described in section 5.2.3 of the Process Document). The UAWG resolved to request to advance to Candidate Recommendation at its 30 August 2001 teleconference.

This document incorporates resolutions of the User Agent Accessibility Guidelines Working Group to all issues raised during the third last call review of the 9 April 2001 version. A snapshot of the third last call issues list is available, as is the disposition of comments.

A list of changes to this document is available.

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."

Please send comments about this document to the public mailing list w3c-wai-ua@w3.org; public archives are available.

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 9
2. The user agent accessibility guidelines 10
1. Support input and output device-independence. 12
2. Ensure user access to all content. 14
3. Allow configuration not to render some content that may reduce accessibility. 19
4. Ensure user control of rendering. 22
5. Ensure user control of user interface behavior. 28
6. Implement interoperable application programming interfaces. . . 30
7. Observe operating environment conventions. 34
8. Implement specifications that benefit accessibility. 35
9. Provide navigation mechanisms. 37
10. Orient the user. 40
11. Allow configuration and customization. 44
12. Provide accessible user agent documentation and help. . . . 47
3. Conformance 49
3.1 Unconditional conformance 50
3.2 Conditional conformance 51
3.3 Conformance details 54
3.4 Conformance levels 56
3.5 Content type labels 57
3.6 Input modality labels 58
3.7 Selection label 58
3.8 Checkpoint applicability 58
3.9 Well-formed conformance claims 59
3.10 Validity of a claim 60
4. Glossary 63
5. References 81
5.1 How to refer to this document 81
5.2 Normative references 82
5.3 Informative references 82
6. Acknowledgments 85

An appendix to this document *[UAAG10-SUMMARY]* summarizes the document's principal goals and structure.

Another 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. Developers, W3C Working Groups, users, and others are encouraged to contribute techniques for incorporation into the Techniques document.

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, if satisfied by user agent developers, will 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 stakeholders intersect and complement each other as follows:

- Protocol (e.g., HTTP) and content format (e.g., HTML, XHTML, XML, SVG, SMIL, MathML, etc.) specifications allow communication on the Web. Format designers include features that authors should use to create accessible content, and features that user agents should support through an accessible user interface.
- 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. The "Web Content Accessibility Guidelines (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 8.1 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 12.1 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. Formats generally do not enable authors to encode all of their knowledge in a way that a user agent can recognize 100%. 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 2.7 and checkpoint 2.11) for cases where content does not conform to WCAG 1.0. Furthermore, this document includes some requirements to address certain widespread authoring practices that are discouraged because they may 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 user agents with multimedia capabilities running in the following type of environment (typically that of a desktop computer):

- The operating environment includes a keyboard;
- Assistive technologies may be used in the operating environment and may communicate with the conforming user agent;

This document is not designed so that user agents on other types of platforms (e.g., handheld devices, kiosks, etc.) will readily conform. This document does not *forbid* conformance by any user agent, but some requirements (e.g., implementation of certain APIs) are not likely to be satisfied on environments other than the target environment. Future work by the UAWG may address the accessibility of user

agents running on handheld devices, etc.

The target 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.

This document was designed to improve the accessibility of target user agents for users with one or more disabilities (including visual, hearing, physical, and cognitive) in two ways:

1. through its own user interface, and
2. 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, speak, 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 pointing device.

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

- 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., the checkpoints of guideline 7 and guideline 11)) but 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 rendering with groups in W3C's Voice Browser Activity.
- Output modalities. This document does not include requirements for braille rendering. Some requirements are specific to graphical rendering and others

specific to synthesized speech rendering (speech rendering requirements are made by checkpoint 4.12 to checkpoint 4.16). Many of the requirements of this document are generic enough to apply to any output modality (including braille). User agents conform to this document by supporting some combination of graphical and audio/speech rendering output; see the section on content type labels for more information.

- 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]*).
- Background image interference. The requirement of checkpoint 3.1 to allow the user to turn off rendering of background images does not extend to multi-layered rendering.
- User control of every user interface component. This document includes some requirements for user control of user interface components that may be changed through content (see guideline 5). However, these requirements do not account for every user interface component that the author may affect (e.g., the author might supply a script that causes text to scroll in the status bar). User agents are required to follow software usability guidelines (see checkpoint 7.3), which are also expected to include requirements for user control over user interface behavior. **Note.** It is more difficult for users to distinguish content from user interface when both are rendered as sound in one dimension, than it is when both are rendered visually in two dimensions. Developers of aural user agents are therefore strongly encouraged to apply the requirements of this document to both content and user agent components.
- Time. This document includes requirements for control of some time parameters (including checkpoint 2.4, checkpoint 4.4, checkpoint 4.5, and checkpoint 4.12). The requirements are for time parameters that the user agent recognizes and controls. This document does not include requirements for control of time parameters managed on the server.
- 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.

Note: The User Agent Accessibility Guidelines Working Group may address these topics in a future version of the User Agent Accessibility Guidelines. Even though UAAG 1.0 does not address these topics, user agent developers are encouraged to consider them in their designs.

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

- The guideline number.
- The statement of the guideline.
- The rationale behind the guideline and 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 visually rendered text" and second group of "checkpoints for audio volume control"." Within each group, checkpoints are ordered according to their priority , e.g., Priority 1 before Priority 2. Within a guideline, checkpoint groupings and checkpoint order have no bearing on conformance .

Each checkpoint definition includes the following:

- The checkpoint number.
- **Informative:** The checkpoint title. This title is not a requirement, just a phrase to help readers remember an important requirement made by the checkpoint statement.
- **Normative:** The priority of the checkpoint.
- **Normative:** The statement or statements of the checkpoint. These statements include one or more requirements that must be satisfied by the user agent (i.e., the "subject of the claim) for the purposes of conformance .
- **Normative:** An optional "content/rendered content/user agent feature/both" label that indicates whether the requirements of the checkpoint must be satisfied by the subject of the claim for all content , all rendered content , for user agent features only, or for both content and user agent features. The label only appears when necessary to disambiguate the checkpoint.
- **Informative:** A link to rationale, implementation details, references, and more information in "Techniques for User Agent Accessibility Guidelines 1.0" *[UAAG10-TECHS]* .
- **Normative:** Content type labels (zero or more). Content type labels are explained in the section on conformance .
- **Informative:** Optional notes about the checkpoint (beginning with the word "**Note**"). They notes clarify the scope of the checkpoint through further description, examples, cross references, and commentary. Some checkpoints in this document are more general than others, and some may overlap in scope. Therefore, a checkpoint may be identified as a "special case" or an "important special case" of one or more other checkpoints.

Each checkpoint definition expresses one or more requirements. These requirements are not technology specific. In fact, they have been designed to be largely technology independent, in order to make sense for a variety of existing and

future technologies. "Techniques for User Agent Accessibility Guidelines 1.0" [UAAG10-TECHS] is an important resource to help developers understand how to "apply" the requirements to HTML, CSS, SMIL, and SVG, and several operating environments. The User Agent Accessibility Guidelines Working Group welcomes comments and anticipates discussion on how to apply these requirements to new technologies in different operating environments.

Each requirement is a "minimal" requirement, which means that for conformance, the user agent is required to satisfy no more than the stated requirement. In many cases, however, it may be easier or less costly (or just be better design) to implement a general feature that satisfies more than a minimal requirement. One general solution might satisfy five checkpoints and be easier to implement than five disconnected features. For instance, a navigable structure view of content that allows users to query elements for their properties is likely to benefit all users and may be used to satisfy a number of requirements of this document.

Some requirements have a wider impact than others. For instance, the keyboard requirements of checkpoint 1.1 have an impact on all other requirements in the document related to user input: any requirement that involves user input must be satisfied through the keyboard. Because the keyboard requirements of checkpoint 1.1 have been factored out, the other checkpoints are shorter; they are written "Allow configuration" instead of "Allow configuration through the keyboard." First-time readers of the document are encouraged to read the full context provided for each checkpoint, including the guideline prose, the surrounding checkpoints (since nearby checkpoints are generally related), notes after checkpoints, and associated techniques (in the Techniques document *[UAAG10-TECHS]*). The checklist *[UAAG10-CHECKLIST]* is also a useful tool (e.g., for evaluating a user agent for conformance), but does not provide the same contextual support.

Priorities

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

Priority 1 (P1)
: 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 (P2)
: 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 (P3)
: 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 different input and output devices.

Since people use a variety of devices for input and output, user agent developers need to ensure redundancy in the user interface . The user may have to operate the user interface with a variety of input devices (keyboard, pointing device, voice input, etc.) and output modalities (e.g., graphical , speech, or braille rendering).

Though it may seem contradictory, enabling full user agent operation through the keyboard is an important part of promoting device-independence given today's user agents. In addition to the fact that some form of keyboard is supported by most platforms, there are several reasons for this:

- For some users (e.g., users with blindness or physical disabilities), operating a user agent with a pointing device may be difficult or impossible since it requires tracking the pointing device position in a two-dimensional visual space. Keyboard operation does not generally require as much movement "through space".
- Some assistive technologies that support a diversity of input and output mechanisms use keyboard APIs for communication with some user agents; see checkpoint 6.6. People who cannot or do not use a pointing device may interact with the user interface with the keyboard, through voice input, a head wand, touch screen, or other device.

While this document only requires keyboard operation for conformance , it promotes device-independence by also allowing people to claim conformance for full pointing device support or full voice support.

As a way to promote output device independence, this guideline requires support for text messages in the user interface because text may be rendered visually, as synthesized speech, and as braille.

The API requirements of guideline 6 also promote device independence by ensuring communication with specialized software.

Checkpoints

1.1 Full keyboard access. (P1)

1. Ensure that the user can operate through keyboard input alone any user agent functionality available through the user interface .

For both content and user agent.

Note: User agents may support at least two types of keyboard access to functionalities: direct access (where user awareness of a location "in space" is not required, as is the case with keyboard shortcuts and navigation of user agent menus) and spatial access (where the user moves the pointing device "in space" via

the keyboard). To satisfy this checkpoint, user agents are expected to provide a mix of both types of keyboard access. User agents should allow direct keyboard access where possible, and this may be redundant with spatial input techniques. Furthermore, the user agent should satisfy this requirement by offering a combination of keyboard-operable user interface controls (e.g., keyboard operable print menus and settings) and direct keyboard operation of user agent functionalities (e.g., a short cut to print the current page). As examples of functionalities, ensure that the user can interact with enabled elements , select content, navigate viewports, configure the user agent, access documentation, install the user agent, operate controls of the user interface, etc., all entirely through keyboard input. It is also possible to claim conformance to this document for full support through pointing device input and voice input. See the section on input modality labels .

1.2 Activate event handlers. (P1)

1. For the element with content focus , allow the user to activate any explicitly associated input device event handlers through keyboard input alone.
2. The user agent is not required to allow activation of event handlers associated with a given device (e.g., the pointing device) in any order other than what the device itself allows.

Note: The requirements for this checkpoint refer to **any** explicitly associated input device event handlers associated with an element, independent of the input modalities for which the user agent conforms. For example, suppose that an element has an explicitly associated handler for pointing device events. Even when the user agent only conforms for keyboard input (and does not conform for the pointing device, for example), this checkpoint requires the user agent to allow the user to activate that handler with the keyboard. This checkpoint is an important special case of checkpoint 1.1. Please refer to the checkpoints of guideline 9 for more information about focus requirements.

1.3 Provide text messages. (P1)

1. 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 .

Note: For example, if the user is alerted of an event by an audio cue, a visually-rendered text equivalent in the status bar could satisfy this checkpoint. Per checkpoint 6.4, a text equivalent for each such message must be available through an API . See also checkpoint 6.5 for requirements for programmatic alert of changes to the user interface.

Guideline 2. Ensure user access to all content.

Ensure that users have access to all content, notably conditional content that may have been provided to meet the requirements of the Web Content Accessibility Guidelines 1.0 [WCAG10].

The checkpoints in this section require the user agent to provide access to all content through a series of complementary mechanisms designed so that if one fails, another will provide some access. The following preferences are embodied in the checkpoints:

- Not all content is rendered at all times. Automatic decision by the user agent about when and where to render conditional content is preferred, but manual choice by the user may be necessary for access.
- Structure is preferred (both the author's specified preferences and the user's structured access), but unstructured access may be necessary for access to all content.
- Rendering according to format specification is preferred, but a source view of text content may be necessary for access (e.g., because of user-side error conditions, authoring errors, inadequate specification, or incorrect user agent implementation). For example, the user may have to look at URIs for information, HTML comments, XML element names, or script data. The user agent should respect authoring synchronization cues for content that changes over time, but also needs to allow the user to control the time intervals when user input is possible.
- Configuration and control of rendering are important for access.

Authors may use the conditional content mechanisms of a specification to satisfy the requirements of the Web Content Accessibility Guidelines 1.0 *[WCAG10]* . Ensuring access to conditional content 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 movies or images).

Checkpoints

2.1 Render content according to specification. (P1)

1. Render content according to format specification (e.g., for a markup language or style sheet).
2. When a rendering requirement of another specification contradicts a requirement of the current document, the user agent may disregard the rendering requirement of the other specification and still satisfy this checkpoint.
3. Rendering requirements include format-defined interactions between author preferences and user preferences/capabilities (e.g., when to render the "`alt`" attribute in HTML, the rendering order of nested `OBJECT` elements in HTML, test attributes in SMIL, and the cascade in CSS2).

Note: If a conforming user agent does not render a content type, it should allow the user to choose a way to handle that content (e.g., by launching another application, by saving it to disk, etc.). The user agent is not required to satisfy this checkpoint for all implemented specifications; see the section on conformance and implementing specifications for more information.

2.2 Provide text view. (P1)

1. For content authored in text formats, provide a view of the text source . For the purposes of this document, text formats are defined to be:
   - all media objects given an Internet media type of "text" (e.g., text/plain, text/HTML, or text/*) as defined in RFC 2046 *[RFC2046]* , section 4.1.
   - all SGML and XML applications, regardless of Internet media type (e.g., HTML 4.01, XHTML 1.1, SMIL, SVG, etc.).

Note: A user agent would also satisfy this checkpoint by providing a source view for any text format, not just implemented text formats. The user agent is not required to satisfy this checkpoint for all implemented specifications; see the section on conformance and implementing specifications for more information.

2.3 Render conditional content. (P1)

1. Allow configuration to provide access to each piece of unrendered conditional content "C".
2. The configuration may be a switch that, for all content, turns on or off the access mechanisms described in the next provision.
3. When a specification does not explain how to provide access to this content, do so as follows:
   - If C is a summary, title, alternative, description, or expansion of another piece of content D, provide access through at least one of the following mechanisms:
     - (1a) render C in place of D;
     - (2a) render C in addition to D;
     - (3a) provide access to C by querying D. In this case, the user agent must also alert the user, on a per-element basis, to the existence of C (so that the user knows to query D);
     - (4a) allow the user to follow a link to C from the context of D.
   - Otherwise, provide access to C through at least one of the following mechanisms:
     - (1b) render a placeholder for C, and allow the user to view the original author-supplied content associated with each placeholder;
     - (2b) provide access to C by query (e.g., allow the user to query an element for its attributes). In this case, the user agent must also alert the user, on a per-element basis, to the existence of C;
     - (3b) allow the user to follow a link in context to C.
4. To satisfy this checkpoint, the user agent may provide access on a per-element

basis (e.g., by allowing the user to query individual elements) or for all elements (e.g., by offering a configuration to render conditional content all the time).

For all content.

Note: For instance, an HTML user agent might allow users to query each element for access to conditional content supplied for the "`alt`", "`title`", and "`longdesc`" attributes. Or, the user agent might allow configuration so that the value of the "`alt`" attribute is rendered in place of all `IMG` elements (while other conditional content might be made available through another mechanism). See checkpoint 2.10 for additional placeholder requirements.

2.4 Allow time-independent interaction. (P1)

1. For rendered content where user input is only possible within a finite time interval controlled by the user agent, allow configuration to provide a view where user interaction is time-independent.
2. The user agent may satisfy this checkpoint by pausing processing automatically to allow for user input, and resuming processing on explicit user request . When this technique is used, pause at the end of each time interval where user input is possible. In the paused state:
   - Alert the user that the rendered content has been paused (e.g., highlight the "pause" button in a multimedia player's control panel).
   - Highlight which enabled elements are time-sensitive.
   - Allow the user to interact with the enabled elements .
   - Allow the user to resume on explicit user request (e.g., by pressing the "play" button in a multimedia player's control panel; see also checkpoint 4.5).
3. The user agent may satisfy this checkpoint by generating a time-independent ("static") view, based on the original content , that offers the user the same opportunities for interaction. The static view should reflect the structure and flow of the original time-sensitive presentation; orientation cues will help users understand the context for various interaction opportunities.
4. When satisfying this checkpoint for a real-time presentation, the user agent may discard packets that continue to arrive after the construction of the time-independent view (e.g., when paused or after the construction of a static view).

Note: If the user agent satisfies this checkpoint by pausing automatically, it may be necessary to pause more than once when there are multiple opportunities for time-sensitive user interaction When pausing, pause synchronized content as well (whether rendered in the same or different viewports) per checkpoint 2.6. 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). See also checkpoint 3.5, which involves client-driven content refresh.

2.5 Make captions, transcripts available. (P1)

1. Allow configuration or control to render text transcripts , collated text transcripts , captions , and auditory descriptions at the same time as the associated audio tracks and visual tracks .

For all content.

Content type labels : Video , Audio .

Note: This checkpoint is an important special case of checkpoint 2.1.

2.6 Respect synchronization cues. (P1)

1. Respect synchronization cues (e.g., in markup) during rendering.

Content type labels : Video , Audio .

Note: This checkpoint is an important special case of checkpoint 2.1.

2.7 Repair missing content. (P2)

1. Allow configuration to generate repair text when the user agent recognizes that the author has failed to provide conditional content that was required by the format specification.
2. The user agent may satisfy this checkpoint by basing the repair text on any of the following available sources of information: URI reference, content type, or element type.

For all content.

Note: Some markup languages (such as HTML 4 *[HTML4]* and SMIL 1.0 *[SMIL]* require the author to provide conditional content for some elements (e.g., the "`alt`" attribute on the `IMG` element). Repair text based on URI reference, content type, or element type is sufficient to satisfy the checkpoint, but may not result in the most effective repair. Information that may be recognized as relevant to repair might not be "near" the missing conditional content in the document object . For instance, instead of generating repair text on a simple URI reference, the user agent might look for helpful information near a different instance of the URI reference in the same document object, or might retrieve useful information (e.g., a title) from the resource designated by the URI reference.

2.8 No repair text. (P3)

1. Allow at least two configurations for when the user agent recognizes that conditional content required by the format specification is present but empty :
   - generate no repair text , or
   - generate repair as described in checkpoint 2.7.

For all content.

Note: In some authoring scenarios, empty content (e.g., a string of zero characters) may make an appropriate text equivalent , such as when non-text content has no other function than pure decoration, or when 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.

2.9 Render conditional content automatically. (P3)

1. Allow configuration to render all conditional content automatically. The user agent is not required to render all conditional content at the same time in a single viewport.
2. Provide access to this content according to format specifications or where unspecified, by applying one of the techniques described in checkpoint 2.3: 1a, 2a, or 1b.

For all content.

Note: For instance, an HTML user agent might allow configuration so that the value of the "`alt`" attribute is rendered in place of all `IMG` elements (while other conditional content might be made available through another mechanism). The user agent may offer multiple configurations (e.g., a first configuration to render one type of conditional content automatically, a second to render another type, etc.).

2.10 Toggle placeholders. (P3)

1. Once the user has viewed the original author-supplied content associated with a placeholder , allow the user to turn off the rendering of the author-supplied content.

Note: For example, if the user agent substitutes the author-supplied content for the placeholder in context, allow the user to "toggle" between placeholder and the associated content. Or, if the user agent renders the author-supplied content in a separate viewport, allow the user to close that viewport. **Note:** See checkpoint 2.3, provision (1b) for placeholder requirements.

2.11 Alert unsupported language. (P3)

1. Allow configuration not to render content in unsupported natural languages , when that content would otherwise be rendered. Content "in a natural language" includes pre-recorded spoken language and text in a given script , i.e., writing system.
2. Indicate to the user in context that author-supplied content has not been rendered.
3. This checkpoint does not require the user agent to allow different configurations for different natural languages.

Note: For example, use a text substitute or accessible graphical icon to indicate that content in a particular language has not been rendered.

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 other 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.

This guideline requires the user agent to allow configuration so that, when loading Web resources , the user agent does not render content in a manner that may pose accessibility problems. Requirements for interactive control of rendered content are part of guideline 4.

Checkpoints

3.1 Toggle background images. (P1)

1. Allow configuration not to render background image content .
2. In this configuration, the user agent is not required to retrieve background images from the Web.
3. 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".

Content type labels : Image .

Note: See checkpoint 2.3 for information about how to provide access to unrendered background images. When background images are not rendered, user agents should render a solid background color instead (see checkpoint 4.3).

3.2 Toggle audio, video, animated images. (P1)

1. Allow configuration not to render audio, video, or animated image content , except on explicit user request . This configuration is required 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 (e.g., when the user activates a link).
2. The user agent may satisfy this checkpoint by making video and animated images invisible and audio silent , but this technique is not recommended.
3. 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.

Content type labels : Animation , Video , Audio .

Note: See checkpoint 2.3 for information about how to provide access to unrendered audio, video, and animated images. See also checkpoint 4.5, checkpoint 4.9, and checkpoint 4.10.

3.3 Toggle animated/blinking text. (P1)

1. Allow configuration to render animated or blinking text content . as motionless, unblinking text. Blinking text is text whose visual rendering alternates between visible and invisible, any rate of change.
2. In this configuration, the user must still have access to the same text content, but the user agent may render it in a separate viewport (e.g., for large amounts of streaming text).
3. The user agent also satisfies this checkpoint by always rendering animated or blinking text as motionless, unblinking text.

Content type labels : VisualText .

Note: Animation (a rendering effect) differs from streaming (a delivery mechanism). Streaming content might be rendered as an animation (e.g., an animated stock ticker or vertically scrolling text) or as static text (e.g., movie subtitles, which are rendered for a limited time, but do not give the impression of movement). See also checkpoint 3.5. This checkpoint does not apply for blinking and animation effects that are caused by mechanisms that the user agent cannot recognize .

3.4 Toggle scripts. (P1)

1. Allow configuration not to execute any executable content (e.g., scripts and applets).
2. In this configuration, provide an option to alert the user when executable content is available (but has not been executed).
3. The user agent is only required to alert the user to the presence of more than zero scripts or applets (i.e., per-element alerts are not required).

Note: This checkpoint does not refer to plug-ins and other programs that are not part of content . Scripts and applets may provide very useful functionality, not all of which causes accessibility problems. Developers should not consider that the user's ability to turn off scripts is an effective way to improve content accessibility; turning off scripts means losing the benefits they offer. Instead, developers should provide users with finer control over user agent or content behavior known to raise accessibility barriers. The user should only have to turn off scripts as a last resort.

3.5 Toggle content refresh. (P1)

1. Allow configuration so that the user agent only refreshes content on explicit user request .
2. In this configuration, alert the user of the refresh rate specified in content, and allow the user to request fresh content manually (e.g., by following a link or confirming a prompt).
3. When the user chooses not to refresh content, the user agent may ignore that content; buffering is not required.
4. This checkpoint only applies when the user agent (not the server) automatically initiates the request for fresh content.

Note: For example, allow configuration to prompt the user to confirm content refresh, at the rate specified by the author.

3.6 Toggle redirects. (P2)

1. Allow configuration so that a "client-side redirect" (i.e., one initiated by the user agent, not the server) only changes content on explicit user request .
2. Allow the user to access the new content on demand (e.g., by following a link or confirming a prompt).
3. The user agent is not required to provide these functionalities for client-side redirects specified to occur instantaneously (i.e., after no delay).

Note: Some HTML user agents support client-side redirects authored using a `META` element with `http-equiv="refresh"`. Authors (and Web masters) should use the redirect mechanisms of HTTP instead.

3.7 Toggle images. (P2)

1. Allow configuration not to render image content .
2. The user agent may satisfy this checkpoint by making images invisible , but this technique is not recommended.

Content type labels : Image .

Note: See checkpoint 2.3 for information about how to provide access to unrendered images.

Guideline 4. Ensure user control of rendering.

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

Providing access to content (see guideline 2) includes enabling users to configure and control its rendering. Users with low vision may require that text be rendered at a size larger than the size specified by the author or by the user agent's default rendering. 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 frame assumed by the author. To make the presentation accessible to these users, user agents rendering multimedia content (audio, video, and other animations), have to allow the user to control the playback rate of this content, and also to stop, start, pause, fast reverse, and fast advance it manually. User agents rendering audio have to allow the user to control the audio volume globally and to allow the user to control independently distinguishable audio tracks.

User agents with speech synthesis capabilities need to allow users to control various synthesized speech rendering parameters. For instance, users who are blind and hard of hearing may not be able to make use of high or low frequencies; these users have to be able to configure their speech synthesizers to use suitable frequencies.

Checkpoints for visually rendered text

4.1 Configure text size. (P1)

1. Allow global configuration of the reference size of visually rendered text , with an option to override reference sizes specified by the author or user agent defaults.
2. Offer a range of text sizes to the user that includes at least:
   - the range offered by the conventional utility available in the operating environment that allows users to choose the text size (e.g., the font size),
   - or, if no such utility is available, the range of text sizes supported by the conventional APIs of the operating environment for drawing text.

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). For example, in HTML, this might be paragraph text. The default reference size of rendered text may vary among user agents. User agents may offer different

mechanisms to allow control of the size of rendered text (e.g., font size control, zoom, magnification, etc.). Refer, for example to the Scalable Vector Graphics specification *[SVG]* for information about scalable rendering.

4.2 Configure font family. (P1)

1. Allow global configuration of the font family of all visually rendered text , with an option to override font families specified by the author or by user agent defaults.
2. Offer a range of font families to the user that includes at least:
   - the range offered by the conventional utility available in the operating environment that allows users to choose the font family,
   - or, if no such utility is available, the range of font families supported by the conventional APIs of the operating environment for drawing text.
3. For text that cannot be rendered properly using the user's preferred font family, the user agent may substitute an alternative font family.

Content type labels : VisualText .

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

4.3 Configure text colors. (P1)

1. Allow global configuration of the foreground and background color of all visually rendered text , with an option to override foreground and background colors specified by the author or user agent defaults.
2. Offer a range of colors to the user that includes at least:
   - the range offered by the conventional utility available in the operating environment that allows users to choose colors,
   - or, if no such utility is available, the range of colors supported by the conventional APIs of the operating environment for specifying colors.

Content type labels : ColorText .

Note: User configuration of foreground and background colors may inadvertently lead to the inability to distinguish ordinary text from selected text, focused text, etc. See checkpoint 10.3 for more information about highlight styles.

Checkpoints for multimedia presentations and other presentations that change continuously over time

4.4 Slow multimedia. (P1)

1. Allow the user to slow the presentation rate of rendered audio and animations (including video and animated images).
2. For a visual track , provide at least one setting between 40% and 60% of the original speed.
3. For a prerecorded audio track including audio-only presentations , provide at

least one setting between 75% and 80% of the original speed.
4. 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 .
5. The user agent is not required to satisfy this checkpoint for audio and animations whose recognized role is to create a purely stylistic effect.

Content type labels : Animation , 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 checkpoint 2.6 and checkpoint 4.7.

4.5 Start, stop, pause, advance, reverse multimedia. (P1)

1. Allow the user to stop, pause, and resume rendered audio and animations (including video and animated images) that last three or more seconds at their default playback rate.
2. Allow the user to navigate efficiently within audio and animations (including video and animated images) that last three or more seconds at their default playback rate. The user agent may satisfy this requirement through forward and backward sequential access techniques (e.g., advance three seconds), or direct access techniques (e.g., play starting at the 10-minute mark), or some combination.
3. When serial techniques are used to satisfy the previous requirement, the user agent is not required to play back content during serial advance or rewind (though doing so may help orient the user).
4. The user agent is not required to satisfy this checkpoint for audio and animations whose recognized role is to create a purely stylistic effect.
5. When the user pauses a real-time audio or animation, the user agent may discard packets that continue to arrive during the pause.

Content type labels : Animation , Audio .

Note: See checkpoint 4.4 for more information about the exception for purely stylistic effects. This checkpoint applies to content that is either rendered automatically or on request from the user. Respect synchronization cues per checkpoint 2.6.

4.6 Position captions. (P1)

1. For graphical viewports, allow the user to position rendered captions with respect to synchronized visual tracks as follows:
   - if the user agent satisfies this checkpoint by using a markup language or style sheet language to provide configuration or control, then the user agent must allow the user to choose from among at least the range of positions enabled by the format
   - otherwise the user agent must allow both non-overlapping and overlapping positions (e.g., by rendering captions in a separate viewport that may be positioned on top of the visual track).
2. In either case, the user agent must allow the user to override the author's specified position.
3. The user agent is not required to change the layout of other content (i.e., reflow) after the user has changed the position of captions.
4. The user agent is not required to make the captions background transparent when those captions are rendered above a related video track.

4.7 Slow other multimedia. (P2)

1. Allow the user to slow the presentation rate of rendered audio and animations (including video and animated images) not covered by checkpoint 4.4.
2. The same speed percentage requirements of checkpoint 4.4 apply.

Content type labels : Animation , Audio .

Note: User agents automatically satisfy this checkpoint if they satisfy checkpoint 4.4 for all audio and animations.

4.8 Control other multimedia. (P2)

1. Allow the user to stop, pause, resume, fast advance, and fast reverse rendered audio and animations (including video and animated images) not covered by checkpoint 4.5.

Content type labels : Animation , Audio .

Note: User agents automatically satisfy this checkpoint if they satisfy checkpoint 4.5 for all audio and animations.

Checkpoints for audio volume control

4.9 Global volume control. (P1)

1. Allow global configuration of the volume of all rendered audio, with an option to override audio volumes specified by the author or user agent defaults.
2. Allow the user to choose zero volume (i.e., silent).

Content type labels : Audio .

Note: User agents should allow configuration of volume through available operating environment controls.

4.10 Independent volume control. (P1)

1. Allow independent control of the volumes of rendered audio sources synchronized to play simultaneously.
2. The user agent is not required to satisfy this checkpoint for audio whose recognized role is to create a purely stylistic effect.
3. The user control required by this checkpoint includes the ability to override author-specified volumes for the relevant sources of audio.

Content type labels : Audio .

Note: See checkpoint 4.4 for more information about the exception for purely stylistic effects. The user agent should satisfy this checkpoint by allowing the user to control independently the volumes of all audio sources (e.g., by implementing a general audio mixer type of functionality). See also checkpoint 4.13.

4.11 Control other volume. (P2)

1. Allow independent control of the volumes of rendered audio sources synchronized to play simultaneously that are not covered by checkpoint 4.10.

Content type labels : Audio .

Note: User agents automatically satisfy this checkpoint if they satisfy checkpoint 4.10 for all audio.

Checkpoints for synthesized speech rendering

4.12 Configure synthesized speech rate. (P1)

1. Allow configuration of the synthesized speech rate, according to the full range offered by the speech synthesizer.

Content type labels : Speech .

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

4.13 Configure synthesized speech volume. (P1)

1. Allow control of the synthesized speech volume, independent of other sources of audio .
2. The user control required by this checkpoint includes the ability to override author-specified synthesized speech volume.

Content type labels : Speech .

Note: See also checkpoint 4.10.

4.14 Configure synthesized speech characteristics. (P1)

1. Allow configuration of synthesized speech characteristics according to the full range of values offered by the speech synthesizer.

Note: Some speech synthesizers allow users to choose values for synthesized speech characteristics at a higher abstraction layer, i.e., by choosing from present options that group several characteristics. Some typical options one might encounter include: "adult male voice", "female child voice", "robot voice", "pitch", "stress", etc. Ranges for values may vary among speech synthesizers.

4.15 Specific synthesized speech characteristics. (P2)

1. Allow configuration of the following synthesized speech characteristics: pitch, pitch range, stress, richness.
2. Pitch refers to the average frequency of the speaking voice.
3. Pitch range specifies a variation in average frequency.
4. Stress refers to the height of "local peaks" in the intonation contour of the voice.
5. Richness refers to the richness or brightness of the voice.

Note: This checkpoint is more specific than checkpoint 4.14: it requires support for the voice characteristics listed. Definitions for these characteristics are based on descriptions in section 19 of the Cascading Style Sheets Level 2 Recommendation *[CSS2]* ; please refer to that specification for additional informative descriptions. Some speech synthesizers allow users to choose values for synthesized speech characteristics at a higher abstraction layer, i.e., by choosing from present options distinguished by "gender", "age", "accent", etc. Ranges of values may vary among speech synthesizers.

Content type labels : Speech .

4.16 Configure synthesized speech features. (P2)

1. Provide support for user-defined extensions to the synthesized speech dictionary, as well as the following functionalities:
   - spell-out: spell text one character at a time or according to language-dependent pronunciation rules;
   - speak-numeral: speak a numeral as individual digits or as a full number; and
   - speak-punctuation: speak punctuation literally or render as natural pauses.

Note: Definitions for the functionalities listed are based on descriptions in section 19 of the Cascading Style Sheets Level 2 Recommendation *[CSS2]* ; please refer to that specification for additional informative descriptions.

Checkpoints related to style sheets

4.17 Choose style sheets. (P1)

1. For user agents that support style sheets:
   - Allow the user to choose from and apply available author style sheets (in content).
   - Allow the user to choose from and apply available user style sheets .
   - Allow the user to ignore author and user style sheets.

Note: By definition, the user agent's default style sheet is always present, but may be overridden by author or user styles. Developers should not consider that the user's ability to turn off author and user style sheets is an effective way to improve content accessibility; turning off style sheet support means losing the many benefits they offer. Instead, developers should provide users with finer control over user agent or content behavior known to raise accessibility barriers. The user should only have to turn off author and user style sheets as a last resort.

Guideline 5. Ensure user control of user interface behavior.

Ensure that the user can control the behavior of viewports and other user interface controls, including those that may be manipulated by the author (e.g., through scripts).

Control of viewport behavior is important to accessibility. For people with visual disabilities or certain types of learning disabilities, it is important that the point of regard – what the user is presumed to be viewing – remain as stable as possible. Unexpected changes may cause users to lose track of how many viewports are open, which viewport has the current focus , etc. This guideline includes requirements for control of opening and closing viewports, the relative position of graphical viewports, changes to focus, and inadvertent form submissions and micropayments.

Checkpoints

5.1 No automatic content focus change. (P2)

1. Allow configuration so that if a viewport opens without explicit user request , its content focus does not automatically become the current focus .
2. Configuration is preferred, but is not required if the content focus can only ever be moved on explicit user request .

5.2 Keep viewport on top. (P2)

1. For graphical user interfaces, allow configuration so that the viewport with the current focus remains "on top" of all other viewports with which it overlaps.

5.3 Manual viewport open only. (P2)

1. Allow configuration so that viewports only open on explicit user request .
2. 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).
3. Allow the user to close viewports.
4. If a viewport (e.g., a frame set) contains other viewports, these requirements only apply to the outermost container viewport.
5. Configuration is preferred, but is not required if viewports can only ever open on explicit user request .
6. 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.

Note: Generally, viewports open automatically as the result of instructions in content . See also checkpoint 5.1 (for control over changes of focus when a viewport opens) and checkpoint 6.5 (for programmatic alert of changes to the user interface).

5.4 Selection and focus in viewport. (P2)

1. Ensure that when a viewport's selection or content focus changes, it is at least partially in the viewport after the change.

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.

5.5 Confirm form submission. (P2)

1. Allow configuration to prompt the user to confirm (or cancel) any form submission.
2. Configuration is preferred, but it not required if forms can only ever be submitted on explicit user request .

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" or "change" event occurs.

5.6 Confirm fee links. (P2)

1. Allow configuration to prompt the user to confirm (or cancel) any payment that results from activation of a fee link .
2. Configuration is preferred, but is not required if fee links can only ever be activated on explicit user request .

5.7 Manual viewport close only. (P3)

1. Allow configuration to prompt the user to confirm (or cancel) closing any viewport that starts to close without explicit user request .

Guideline 6. Implement interoperable application programming interfaces.

Implement interoperable interfaces to communicate with other software (e.g., assistive technologies, the operating environment, plug-ins, etc.).

This guideline addresses interoperability between a conforming user agent and other software, in particular assistive technologies . The checkpoints of this guideline require implementation of application programming interfaces (APIs) for communication. There are three types of requirements in this guideline:

1. Requirements for what information must be communicated through an API.
2. Requirements for which APIs or types of APIs must be used to communicate this information.
3. Requirements for additional characteristics of these APIs.

Note: The User Agent Accessibility Guidelines Working Group believes that, in order to promote interoperability between a conforming user agent and *more than one* assistive technology, it is more important to implement conventional APIs than custom APIs, even though custom APIs may superior access. When conventional APIs do not allow users to satisfy the requirements of these checkpoints, however, developers may implement alternative APIs in order to conform to this document.

Checkpoints

6.1 DOM read access. (P1)

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:
   - the Core module for HTML;
   - the Core and XML modules for XML.

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

6.2 DOM write access. (P1)

1. 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:
   - the Core module for HTML;
   - the Core and XML modules for XML.

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.

6.3 Programmatic access to non-HTML/XML content. (P1)

1. For markup languages other than HTML and XML, provide programmatic read access to content .
2. Provide programmatic write access for those parts of content that the user can modify through the user interface. To satisfy these requirements, implement at least one API that is either
   - defined by a W3C Recommendation, or
   - a publicly documented API designed to enable interoperability with assistive technologies.
3. If no such API is available, or if available APIs do not enable the user agent to satisfy the requirements, implement at least one publicly documented API to satisfy the requirements, *and* follow operating environment conventions for the use of input and output APIs .
4. An API is considered available if the specification of the API is published (e.g., as a W3C Recommendation) in time for integration into a user agent's development cycle.

Note: This checkpoint addresses content not covered by checkpoints checkpoint 6.1 and checkpoint 6.2.

6.4 Programmatic operation. (P1)

1. Provide programmatic read access to user agent user interface controls.
2. Provide programmatic write access for those controls that the user can modify through the user interface. For security reasons, user agents are not required to allow instructions in content to modify user agent user interface controls.
3. To satisfy these requirements, implement at least one API that is either
   - defined by a W3C Recommendation, or

- a publicly documented API designed to enable interoperability with assistive technologies.

4. If no such API is available, or if available APIs do not enable the user agent to satisfy the requirements, implement at least one publicly documented API that allows programmatic operation of all of the functionalities that are available through the user agent user interface, *and* follow operating environment conventions for the use of input and output APIs .
5. An API is considered available if the specification of the API is published (e.g., as a W3C Recommendation) in time for integration into a user agent's development cycle.

For user agent features.

Note: APIs used to satisfy the requirements of this checkpoint may be platform-independent APIs such as the W3C DOM, conventional APIs for a particular operating environment, conventional APIs for programming languages, plug-ins , virtual machine environments, etc. User agent developers are encouraged to implement APIs that allow assistive technologies to interoperate with multiple types of software in a given operating environment (user agents, word processors, spreadsheet programs, etc.), as this reuse will benefit users and assistive technology developers. User agents should always follow operating environment conventions for the use of input and output APIs.

6.5 Programmatic alert of changes. (P1)

1. Provide programmatic alert of changes to content , user interface controls, selection , content focus , and user interface focus .
2. To satisfy these requirements, implement at least one API that is either
   - defined by a W3C Recommendation, or
   - a publicly documented API designed to enable interoperability with assistive technologies.
3. If no such API is available, or if available APIs do not enable the user agent to satisfy the requirements, implement at least one publicly documented API to satisfy the requirements, *and* follow operating environment conventions for the use of input and output APIs .
4. An API is considered available if the specification of the API is published (e.g., as a W3C Recommendation) in time for integration into a user agent's development cycle.

For both content and user agent.

Note: For instance, when user interaction in one frame causes automatic changes to content in another, provide a programmatic alert. This checkpoint does not require the user agent to alert the user of *rendering changes* caused by content (e.g., an animation effect or an effect caused by a style sheet), just changes to the content itself.

6.6 Conventional keyboard APIs. (P1)

1. Follow operating environment conventions when implementing APIs for the keyboard .
2. If such APIs for the keyboard do not exist, implement publicly documented APIs for the keyboard.

Note: An operating environment may define more than one conventional API for the keyboard. For instance, for Japanese and Chinese, input may be processed in two stages, with an API for each.

6.7 API character encodings. (P1)

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

For both content and user agent.

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.

6.8 DOM CSS access. (P2)

1. For user agents that implement Cascading Style Sheets (CSS), provide programmatic access to those style sheets in content by conforming to the CSS module of the W3C Document Object Model (DOM) Level 2 Style Specification *[DOM2STYLE]* and exporting the interfaces it defines.
2. For the purposes of satisfying this checkpoint, Cascading Style Sheets (CSS) are defined by either CSS Level 1 *[CSS1]* or CSS Level 2 *[CSS2]* .

Note: Please refer to the "Document Object Model (DOM) Level 2 Style Specification" *[DOM2STYLE]* for information about CSS versions covered.

6.9 Timely access. (P2)

1. Ensure that programmatic exchanges proceed in a timely manner.

For both content and user agent.

Note: For example, the programmatic exchange of information required by other checkpoints in this document should 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. Timely exchange is also important for the proper synchronization of alternative renderings. The techniques for this checkpoint explain how developers can reduce communication delays. This will help ensure that assistive technologies have timely access to the document object model and other

information that is important for providing access.

Guideline 7. Observe operating environment conventions.

Observe operating environment conventions for the user agent user interface, documentation, installation, etc.

Part of user agent accessibility involves following the conventions of the user's operating environment. This includes:

- 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).

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

7.1 Focus and selection conventions. (P1)

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

Note: This checkpoint is an important special case of checkpoint 7.3. See also checkpoint 9.1 and checkpoint 9.2.

7.2 Respect input configuration conventions. (P1)

1. Ensure that default input configurations of the user agent do not interfere with operating environment accessibility conventions (e.g., for keyboard accessibility).

For user agent features.

Note: Information about operating environment accessibility conventions is available in the Techniques document *[UAAG10-TECHS]* . See also checkpoint 11.5.

7.3 Operating environment conventions. (P2)

1. Follow operating environment conventions that benefit accessibility. In particular, follow conventions that benefit accessibility for user interface design, keyboard configuration, product installation, and documentation .
2. For the purposes of this checkpoint, an operating environment convention that benefits accessibility is either
   - one identified as such in operating environment design or accessibility guidelines, or
   - one that allows the author to satisfy any requirement of the "Web Content Accessibility Guidelines 1.0" *[WCAG10]* or of the current document.

For user agent features.

Note: Some of these conventions (e.g., sticky keys, mouse keys, show sounds, etc.) are discussed in the Techniques document *[UAAG10-TECHS]* .

7.4 Input configuration indications. (P2)

1. Follow operating environment conventions to indicate the input configuration .

For user agent features.

Note: For example, in some operating environments, when a functionality may be triggered through a menu and through the keyboard, the developer may design the menu entry so that the character of the activating key is also shown. This checkpoint is an important special case of checkpoint 7.3. See also checkpoint 11.5.

Guideline 8. 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 6).

While developers should implement the accessibility features of any specification (checkpoint 8.1), this document recommends conformance to W3C Recommendations in particular (checkpoint 8.2) 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

8.1 Implement accessibility features. (P1)

1. Implement the accessibility features of specifications (markup languages, style sheet languages, metadata languages, graphics formats, etc.). For the purposes of this checkpoint, an accessibility feature is either
   - one identified as such, or
   - one that allows the author to satisfy any requirement of the "Web Content Accessibility Guidelines 1.0" *[WCAG10]* .

For all content.

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. The user agent is not required to satisfy this checkpoint for all implemented specifications; see the section on conformance and implementing specifications for more information.

8.2 Conform to specifications. (P2)

1. Use and conform to either
   - W3C Recommendations when they are available and appropriate for a task, or
   - non-W3C specifications that enable the creation of content that conforms at level A or better to the Web Content Accessibility Guidelines 1.0 *[WCAG10]* .
2. When a requirement of another specification contradicts a requirement of the current document, the user agent may disregard the requirement of the other specification and still satisfy this checkpoint.
3. 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.

For all content.

Note: For instance, for markup, the user agent may conform to HTML 4 *[HTML4]* , XHTML 1.0 *[XHTML10]* , or XML 1.0 *[XML]* . For style sheets, the user agent may conform to CSS (*[CSS1]* , *[CSS2]*). For mathematics, the user agent may conform to MathML 2.0 *[MATHML20]* . For synchronized multimedia, the user agent may

conform to SMIL 1.0 *[SMIL]* . The user agent is not required to satisfy this checkpoint for all implemented specifications; see the section on conformance and implementing specifications for more information.

Guideline 9. 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 enabled element), and review the surrounding context (to orient themselves). Providing a variety of navigation and search mechanisms helps users with disabilities (and all users) access content more efficiently. Navigation and searching are 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 enabled 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 (e.g., to a particular link or paragraph) 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." Direct navigation may be possible with the pointing device or the keyboard (e.g., keyboard shortcuts).

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, enabled elements, navigation mechanisms, containers, etc. For information about programmatic access to document structure, see guideline 6.

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

9.1 Provide content focus. (P1)

1. Provide at least one content focus for each viewport (including frames) where enabled elements are part of the rendered content .
2. Allow the user to make the content focus of each viewport the current focus .

Note: For example, when two frames of a frameset contain enabled elements, allow the user to make the content focus of either frame the current focus. Note that viewports "owned" by plug-ins that are part of a conformance claim are also covered by this checkpoint.

9.2 Provide user interface focus. (P1)

1. Provide a user interface focus .

9.3 Move content focus. (P1)

1. Allow the user to move the content focus to any enabled element in the viewport .
2. Allow configuration so that the content focus of a viewport only changes on explicit user request . Configuration is not required if the content focus only ever changes on explicit user request. See also checkpoint 5.1.
3. If the author has not specified a navigation order, allow at least forward sequential navigation to each element, in document order.
4. The user agent may also include disabled elements in the navigation order.

Note: In addition to forward sequential navigation, the user agent should also allow reverse sequential navigation. This checkpoint is an important special case of checkpoint 9.9.

9.4 Restore history. (P1)

1. For user agents that implement a viewport history mechanism, for each state in a viewport's browsing history, maintain information about the point of regard , content focus , and selection .
2. When the user returns to any state in the viewport history, restore the saved values for these three state variables.

Note: For example, when the user uses the "back button", restore the point of regard, content focus, and selection for previous state in the viewport's history.

9.5 No events on focus change. (P2)

1. Allow configuration so that moving the content focus to or from an enabled element does not automatically activate any explicitly associated event handlers .

Note: For instance, in this configuration for an HTML document, do not activate any handlers for the '`onfocus`', '`onblur`', or '`onchange`' attributes. In this configuration, user agents should still apply any stylistic changes (e.g., highlighting) that may occur when there is a change in content focus .

9.6 Show event handlers. (P2)

1. For the element with content focus , make available the list of input device event handlers explicitly associated with the element.

Note: For example, allow the user to query the element with content focus for the list of input device event handlers, or add them directly to the serial navigation order described in checkpoint 9.3. See checkpoint 1.2 for information about activation of event handlers associated with the element with focus.

9.7 Move content focus optimally. (P2)

1. Allow the user to move the content focus to any enabled element in the viewport .
2. If the author has not specified a navigation order, allow at least forward and reverse sequential navigation to each element, in document order.
3. The user agent must not include disabled elements in the navigation order.

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

9.8 Text search. (P2)

1. Allow the user to search within rendered text for a sequence of characters from the document character set .
2. Allow the user to start a forward search (in document order) from any selected or focused location in content.
3. When there is a match do both of the following:
   - move the viewport so that the matched text content is within it, and
   - allow the user to search for the next instance of the text from the location of the match.
4. Alert the user when there is no match, when the search reaches the end of content, and prior to any wrapping. A wrapping search is one that restarts automatically at the beginning of content once the end of content has been reached.
5. Provide a case-insensitive search option for text in scripts (i.e., writing systems) where case is significant.

For all rendered content.

Note: If the user has not indicated a start position for the search, the search should start from the beginning of content. Per checkpoint 7.3, use operating environments conventions for indicating the result of a search (e.g., selection or content focus).

9.9 Structured navigation. (P2)

1. Allow the user to navigate efficiently to and among important structural elements in rendered content .
2. Allow forward and backward sequential navigation to these important structural elements.

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.

9.10 Configure important elements. (P3)

1. Allow configuration of the set of important elements required by checkpoint 9.9 and checkpoint 10.5.
2. Allow the user to include and exclude element types in the set of elements.

Note: For example, allow the user to navigate only paragraphs, or only headings and paragraphs, or to suppress and restore navigation bars, to navigate within and among tables and table cells, etc.

Guideline 10. 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 through the user interface (visually, as audio, or as braille) include:

- information about the current state of the user's interaction with content: where the viewport is in content (shown, for example, through proportional scroll bars), which viewport has the current focus , where the user has selected content, a history mechanism, the title of the current document or frame, etc. These clues need to be available to the user in a device-independent manner;
- information about specific 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.
- information about the structure of content. For instance, a navigable outline view can accelerate access to content while preserving context.

Orientation mechanisms such as these are especially important to users who view content serially, (e.g., when rendered as synthesized 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 need to 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.

This guideline also includes requirements to allow the user to control some user agent behavior (form submission and activation of fee links) that, if carried out automatically, might go unnoticed by some users (e.g., users with blindness) or might disorient others (e.g., some users with a cognitive disability).

Checkpoints

10.1 Table orientation. (P1)

1. Make available to the user the purpose of each rendered table (e.g., as expressed in a summary or table caption) and the relationships among the table cells and headers.

Note: This checkpoint refers only to table purpose and cell/header relationship information that the user agent 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. This checkpoint is an important special case of checkpoint 2.1.

10.2 Highlight selection and content focus. (P1)

1. Provide a mechanism for highlighting the selection and content focus of each viewport.
2. The highlight mechanism must not rely on color alone.
3. Allow global configuration of selection and focus highlight styles.
4. For graphical viewports, if the highlight mechanism involves colors or text decorations , offer a range of colors or text decorations to the user that includes at least:
   - the range offered by the conventional utility available in the operating environment that allows users to choose colors or text decorations,
   - or, if no such utility is available, the range of colors or text decorations supported by the conventional APIs of the operating environment for specifying colors or drawing text.

Note: Examples of highlight mechanisms include foreground and background color variations, underlining, distinctive synthesized speech prosody, border styling, 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 7.1.

10.3 Distinct default highlight styles. (P1)

1. Ensure that all of the default highlight styles for the selection and content focus , as well as for enabled elements , recently visited links, and fee links in rendered content :
   - do not rely on color alone, and
   - differ from each other, and not by color alone.
2. This checkpoint does not apply to those highlight styles inherited from the operating environment as default values, as long as the user can change the styles in the operating environment.

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, enabled 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.

10.4 Highlight special elements. (P2)

1. Provide a mechanism for highlighting all enabled elements , recently visited links, and fee links in rendered content .
2. Allow the user to configure the highlight styles. The highlight mechanism must not rely on color alone.
3. For graphical viewports, if the highlight mechanism involves text size, font family, colors, or text decorations , offer the corresponding range of values required by checkpoint 4.1, checkpoint 4.2, checkpoint 4.3, or checkpoint 10.2.
4. For a graphically rendered enabled elements, highlight the most specific rendered element that:
   - encompasses the enabled element, and
   - is rendered as a coherent unit according to specification.

   For example, an HTML user agent rendering a PNG image as part of an image map is only required to highlight the image as a whole, not each enabled region. On the other hand, an SVG user agent rendering an SVG image with embedded graphical links is required to highlight each graphical link that may be rendered independently according to the SVG specification.

Note: Examples of highlight mechanisms include foreground and background color variations, font variations, underlining, distinctive synthesized speech prosody, border styling, etc.

10.5 Outline view. (P2)

1. 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.).
2. What constitutes a label is defined by each markup language specification. A label is not required to be text only.

Note: This checkpoint is meant to provide the user with a simplified view of content (e.g, a table of contents). 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. 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 9.9. By making the outline view navigable, it is possible to satisfy this checkpoint and checkpoint 9.9 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 9.10.

10.6 Provide link information. (P3)

1. To help the user decide whether to traverse a link, make available the following information about it:
   - link element content,
   - link title,
   - whether the link is internal to the resource (e.g., the link is to a target in the same Web page),
   - 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.
2. The user agent is not required to compute or make available information that requires retrieval of linked Web resources .

Checkpoints for the user interface

10.7 Highlight current viewport. (P1)

1. Provide a mechanism for highlighting the viewport with the current focus (including any frame that takes current focus).
2. For graphical viewports, the default highlight mechanism must not rely on color alone.
3. This default color requirement does not apply if the highlight mechanism is inherited from the operating environment as the default and the user can change it in the operating environment.

Note: This checkpoint is an important special case of checkpoint 1.1. See also to checkpoint checkpoint 7.1.

10.8 Indicate rendering progress. (P3)

1. Indicate the viewport's position relative to 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.).
2. 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.
3. For two-dimensional renderings, relative position includes both vertical and horizontal positions.
4. 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.

Guideline 11. 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 need to be able to configure the user agent according to their preferences for styles, graphical user interface configuration, keyboard configuration, etc. Most of the checkpoints in this guideline pertain to the input configuration: how user agent behavior is controlled through keyboard input, pointing device input, and voice input.

Checkpoints

11.1 Current user bindings. (P1)

1. Provide information to the user about current user preferences for input configurations .
2. 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).

For user agent features.

11.2 Current author bindings. (P2)

1. Provide a centralized view of the current author-specified input configuration bindings.
2. The user agent may satisfy this checkpoint by providing different views for different input modalities (keyboard, pointing device, voice, etc.).

For all content.

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.

11.3 Override bindings. (P2)

1. Allow the user to override any binding that is part of the user agent default input configuration .
2. The user agent is not required to allow the user to override conventional bindings for the operating environment (e.g., for access to help).
3. The override requirement only applies to bindings for the same input modality (e.g., the user must be able to override a keyboard binding with another keyboard binding).

For user agent features.

Note: See also checkpoint 11.5, checkpoint 11.7, and checkpoint 12.3.

11.4 Single key access. (P2)

1. Allow the user to override any binding in the user agent default keyboard configuration with a binding to either a key plus modifier keys or to a single-key. In this checkpoint, "key" refers to a physical key of the keyboard (rather than, say, a character of the document character set).
2. For each functionality in the set required by checkpoint 11.5, allow the user to configure a single-key binding (i.e., one key press performs the task, with zero modifier keys).
3. If the number of physical keys on the keyboard is less than the number of functionalities required by checkpoint 11.5, allow single-key bindings for as many of those functionalities as possible.
4. The single-key binding requirements may be satisfied with a "single-key mode" (i.e., a mode where the current bindings are replaced by a set of single-key bindings).
5. The user agent is not required to allow the user to override conventional bindings for the operating environment (e.g., for access to help).
6. This checkpoint does not require single physical key bindings for character input, only for the activation of user agent functionalities.

For user agent features.

Note: Because single-key access is so important to some users with physical disabilities, user agents should ensure that (1) most keys of the physical keyboard may be configured for single-key bindings, and (2) most functionalities of the user agent may be configured for single-key bindings. For information about access to user agent functionality through a keyboard API, see checkpoint 6.6.

11.5 Default binding requirements. (P2)

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

For user agent features.

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 11.3.

11.6 User profiles. (P2)

1. For the configuration requirements of this document, allow the user to save user preferences in at least one user profile .
2. Allow the user to choose from among available default profiles, profiles created by the same user, and no profile (i.e., the user agent default settings).

For user agent features.

11.7 Configure tool bars. (P3)

1. 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.

For user agent features.

Note: This checkpoint is a special case of checkpoint 11.3.

Guideline 12. Provide accessible user agent documentation and help.

Ensure that the user can learn about software features that benefit accessibility from the documentation. Ensure that the documentation is accessible.

Documentation of the user interface is important, as is documentation of the user agent's underlying functionalities. While intuitive user interface design is valuable to many users, some users may still not be able to understand or be able to operate the native user interface without thorough documentation (e.g., a user with blindness may not find a graphical user interface intuitive without supporting documentation).

There are three types of requirements in this guideline:

1. accessibility of the documentation (checkpoint 12.1);
2. minimal requirements of what must be documented (checkpoint 12.2, checkpoint 12.3, and checkpoint 12.4). Documentation should include much more to explain how to install, get help for, use, or configure the user agent;
3. organization of the documentation (checkpoint 12.5).

Refer to checkpoint 7.3 for information about following system conventions for documentation.

Checkpoints

12.1 Accessible documentation. (P1)

1. Ensure that at least one version of the user agent documentation conforms to at least Level Double-A of the Web Content Accessibility Guidelines 1.0 *[WCAG10]* .

For user agent features.

12.2 Document accessibility features. (P1)

1. Document all user agent features that benefit accessibility.
2. For the purposes of this checkpoint, a user agent feature that benefits accessibility is one implemented to satisfy the requirements of this document (including the requirements of checkpoints 8.1 and 7.3).
3. The user agent may satisfy this checkpoint either by
   - providing a centralized view of the accessibility features, or
   - integrating accessibility features into the rest of the documentation.

For user agent features.

Note: The help system should include discussion of user agent features that benefit accessibility. The user agent should satisfy this checkpoint by providing both centralized and integrated views of accessibility features in the documentation.

12.3 Document default bindings. (P1)

1. Document the default user agent input configuration (e.g., the default keyboard bindings).

For user agent features.

Note: If the default input configuration is inconsistent with conventions of the operating environment, the documentation should alert the user.

12.4 Document changes. (P2)

1. Document changes from the previous version of the user agent to accessibility features, including accessibility features of the user interface.
2. Accessibility features are those defined in checkpoint 12.2.

For user agent features.

12.5 Dedicated section on accessibility. (P2)

1. Provide a centralized view of all features of the user agent that benefit accessibility in a dedicated section of the documentation .
2. The features that benefit accessibility are those defined in checkpoint 12.2.

For user agent features.

Note: The user agent satisfies this checkpoint automatically by providing a centralized view of accessibility features to satisfy checkpoint 12.2. However, developers are encouraged to integrate descriptions of accessibility features into the documentation alongside other features, in addition to providing a centralized view.

3. Conformance

This normative section defines what it means to conform to this document and explains how to make a valid conformance claim . The following are important conformance concepts.

- **Conformance and conformance claims differ.** This document distinguishes conformance requirements and conformance *claim* requirements. The sections on unconditional conformance and conditional conformance explain the conformance requirements. The section on well-formed claims explains the claim requirements (e.g., identification of the components that make up the user agent, the operating environment in which they run, etc.) Here is a sample claim (expressed in HTML):

```
  <p>On 31 August 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-UAAG10-20010831, level
Double-A. Unsupported content types: Video, Speech.
Unsupported input modalities: Voice. (see section 3.1 of
the UAAG 1.0). The <a
href="http://example.com/checkpoints">list of checkpoints
that do not apply</a> is available online.</p>
```

- **Modular conformance.** A conforming user agent is not required to be a single piece of software. In general, a conforming user agent will consist of several coordinated components, such as a browser, a multimedia player, documentation on the Web, etc. The current document places no restrictions on the type or number of components that make up the "subject of a conformance claim", i.e., the user agent (i.e., set of components) about which someone has made a conformance claim.
- **Conditional conformance.** A user agent is not required to satisfy every checkpoint in order to conform. This document allows "conditional conformance ", which means conformance to less than (or more than) a default set of requirements. Claimants may not pick and choose which requirements they wish to satisfy in order to conform conditionally; conditional conformance is governed by several mechanisms described below:
  1. conformance levels ,
  2. content type labels ,
  3. input modality labels ,
  4. selection label .

  When a user agent conforms conditionally, a conformance claim about the user agent must indicate how the set of satisfied requirements differs from the default set; see the section on well-formed claims .
- **Applicability.** Some checkpoints may not apply to a particular user agent because of the nature of the user agent's user interface or the nature of the

format(s) implemented by the user agent. If a checkpoint (or portion of a checkpoint) doesn't apply, the user agent is not required to satisfy it for conformance. A claimant must state in a well-formed conformance claim which checkpoints, if any, do not apply. See the section on applicability for information about how to determine whether a checkpoint applies.

In this document (notably in the checkpoints and in this section on conformance), the terms "must", "should", and "may" (and related terms) are used in accordance with RFC 2119 *[RFC2119]* .

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, some software may conform to this document but still be inaccessible to some users with disabilities. Some requirements of this document may not benefit some users for some content, but the requirements are expected to benefit many users with disabilities, for general purpose content. For more information, please see the sections on known limitations of this document and restricted functionality and conformance .

3.1 Unconditional conformance

A user agent conforms unconditionally to this document if:

1. it satisfies all of the requirements of all the checkpoints. Note that each checkpoint statement includes one or more requirements. The requirements made by a checkpoint include those associated with any content type labels for that checkpoint. Certain checkpoints also include labels that indicate (when there might be ambiguity) whether the requirements are for all content, for all rendered content, for user agent features, or for both user agent features and content ;
2. for each checkpoint in guideline 6, it satisfies the requirements by implementing APIs . For every other checkpoint, it satisfies the requirements by implementing at least one mechanism other than an API . **Note:** The checkpoints outside of guideline 6 may be satisfied by assistive technologies as well, but are required by the current document to be satisfied by a conforming user agent. For example, checkpoint 9.3 involves navigation that must be possible through the user interface, not just via an API. Note that an assistive technology may be part of the subject of a claim .

These requirements together form the "default" set of conformance requirements.

3.2 Conditional conformance

To allow user agents with different capabilities to conform, and to facilitate comparisons of claims about different user agents, this document defines allows conditional conformance. A user agent conforms conditionally if it satisfies any set of requirements that results from starting with the default set of requirements and removing or adding requirements according to these steps:

1. Choose a conformance level ; conformance levels A or Double-A remove requirements from the default set.
2. Remove the requirements associated with any unsupported content type labels . In order to conform conditionally, a user agent must satisfy the requirements of at least one content type label.
3. Add requirements associated with any supported input modality label . **Note:** In the default set of requirements, the only input device requirements relate to keyboard input.
4. If the user agent does not implement a selection mechanism, remove the requirements of any checkpoints or parts of checkpoints associated with the selection label .
5. Remove the requirements of any checkpoints or parts of checkpoints that do not apply .

Since these steps may produce very different sets of checkpoints for different user agents, a well-formed conformance claim must indicate how the set of requirements chosen for the claim differs from the default set . The checklist *[UAAG10-CHECKLIST]* may prove useful when documenting the details of a conditional conformance claim.

Example of a conditional conformance requirement set

The following example illustrates how to apply the above steps to determine which requirements must be satisfied for conformance, and what would be required as part of a well-formed conformance claim. This informative example does not illustrate a complete user agent evaluation.

Consider a user agent with these capabilities:

- it supports keyboard and pointing device input;
- it renders text (in color) and implements:
  - one audio format,
  - two image formats,
  - two other animation formats (besides video, which is considered an animation format in this document);
- it feeds video to a plug-in for rendering;
- it doesn't support synthesized speech output;
- it implements a selection mechanism.

Step 1: Choose a conformance level.

The claimant wishes to conform at level Double-A. This establishes a set of requirements consisting of all of the requirements of all the priority 1 and 2 checkpoints.

Step 2: Remove the requirements associated with any unsupported content type labels.

The claimant wishes to claim conformance for the user agent's support of text, images, audio, and video. The claimant does not wish to claim conformance for other animation formats.

The following content type labels are therefore relevant: VisualText, ColorText, Image, Animation, Video, and Audio. This means that:

- the claimant must remove the set of requirements associated with the Speech content type label.
- the claimant must satisfy the requirements associated with the other content type labels.

Step 3: Remove the requirements of any checkpoints or parts of checkpoints that do not apply.

Consider checkpoint 4.4, for example, which is associated with both the Audio and Animation content type labels:

4.4 Slow multimedia. (P1)

1. Allow the user to slow the presentation rate of rendered audio and animations (including video and animated images).
2. For a visual track , provide at least one setting between 40% and 60% of the original speed.
3. For a prerecorded audio track including audio-only presentations , provide at least one setting between 75% and 80% of the original speed.
4. 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 .
5. The user agent is not required to satisfy this checkpoint for audio and animations whose recognized role is to create a purely stylistic effect.

Suppose that:

1. The claimant wishes to claim support for the two image formats, the one audio format, and the one video format;
2. The claimant does not wish to claim support for the other two animation formats (e.g., because the user agent doesn't satisfy the requirements of checkpoint 4.4 for those animation formats);

3. The user agent does not implement any synchronized multimedia formats.

The resulting applicable requirements from this checkpoint would be:

- For the audio format: Allow the user to slow the presentation rate of audio. For a prerecorded audio track including audio-only presentations, provide at least one setting between 75% and 80% of the original speed.
- For the video format: Allow the user to slow the presentation rate of video. For a visual track, provide at least one setting between 40% and 60% of the original speed.
- For the image formats: None, since the Image content type label does not include checkpoint 4.4.
- Limitation of scope for any format: The user agent is not required to satisfy the requirements of this checkpoint for audio and animations whose recognized role is to create a purely stylistic effect.

The following requirements would not apply:

- 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. **Note:** The relevant applicability provision is provision three: control of a content property that the subject cannot recognize. In this case, no format implemented by the user agent supports synchronized multimedia.

Step 4: Add requirements related to the selection.

In this example, since the user agent implements a selection mechanism , it must satisfy the requirements associated with the selection label .

Step 5: Add requirements associated with any supported input modality label.

In this example, the claimant does not wish to claim conformance for complete operation for pointing device or voice input, so no requirements are added.

Construct a well-formed conformance claim.

The following information is an excerpt of that required for a well-formed claim:

- Conformance level satisfied: Double-A
- Information about the subject. Both the "main" user agent and the plug-in used to support video must be identified in the claim (since the plug-in is the component used to satisfy the requirements for video).

The user agent does not conform unconditionally , therefore, the claim must also include the following information (excerpted from a complete claim):

- A general statement about lack of support for the Speech content type label: "This user agent does not support the requirements of the Speech content type label. "
- A specific statement about content type support for checkpoint 4.4: "This user agent satisfies the requirements of the Animation content type label for the audio format A and the video format V. It does not satisfy the Animation requirements for animation formats Y and Z."
- A specific statement about applicability for checkpoint 4.4: "The synchronized multimedia requirements of checkpoint checkpoint 4.4 do not apply because the user agent does not implement any formats that support synchronized multimedia."

3.3 Conformance details

The following normative subsections provide detail that is relevant to both unconditional and conditional conformance.

Requirements for content, for rendered content, for user agent features, or both user agent features and content

The requirements of certain checkpoints might apply equally well to content or to user agent user interface features. When it is necessary to remove ambiguity about the scope of a checkpoint, the checkpoint includes a label to indicate whether the requirements must be satisfied:

1. for content only, i.e., the document object only;
2. for rendered content only;
3. for user agent features only, i.e., everything that is not content (such as components of the user agent user interface , user preferences, the user agent documentation , and the user interface focus);
4. for both content and user agent features.

Many of the content-only and rendered content-only requirements also make sense for the user agent user interface (e.g., allow the user to render blinking text as motionless text). User agent developers are encouraged to consider the content-only requirements when designing the user agent's user interface.

The user agent may satisfy a content-only or rendered content-only requirement with a mechanism that *also* involves user agent features. For instance, to satisfy checkpoint 4.9, the user agent may provide a single control for all volume (including content and user interface features). Similarly, to satisfy checkpoint 3.3, the user agent may offer a single configuration that turns off blinking in both content and the user interface.

Conformance and implementing specifications

A user agent may conform by satisfying the checkpoint requirements of this document for *some*, but not all, implemented specifications and APIs . For example, a developer may implement ten image formats but only wish to claim "Image " conformance for three of them.

In particular, the following requirements may be satisfied for some but not all implemented specifications:

- requirements associated with a content type label ;
- the API requirements of checkpoint checkpoint 6.3, checkpoint 6.4, and checkpoint 6.5.
- the rendering requirements of checkpoints 2.1 and 2.2;
- the format requirements of checkpoints 8.1 and 8.2.

Configuration requirements

The user agent may satisfy the configuration requirements of this document through configuration files (e.g., profiles, initialization files, themes, etc.). For instance, style sheets might be used as a mechanism to satisfy the highlight and configuration requirements of checkpoints 10.2 and 10.4. Any functionality that is configurable through a configuration file should also be configurable through the user agent user interface . Furthermore, if configuration files may be edited by hand, the user agent documentation should explain the configuration file format, or refer to an explanation (such as a format specification).

For some of the checkpoints in this document (checkpoint 3.3, checkpoint 5.1, checkpoint 5.3, checkpoint 5.5, checkpoint 5.6), configuration is preferred, but not required to satisfy the checkpoint in some circumstances. For other checkpoints, the configuration requirement is considered as important as the functionality being configured.

Since this document allows conformance by multiple software components (e.g., a browser, a media player, and several plug-ins), there are likely to be times when, to satisfy the configuration requirements of the document, each component has to provide for configuration independently. To make configuration easier for the user, components should share and inherit configurations (including from the operating environment).

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, including its user interface) 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.

Some of the checkpoints in this document involve operating environment conventions. When a user agent runs in more than one operating environment (e.g., a user agent implemented in Java on top of another operating system), developers may satisfy the requirements of this document by following the conventions of a single operating environment. Developers should follow the conventions that benefit accessibility most, while meeting the developers' design goals. For instance, some developers may prefer cross-platform consistency over consistency with other user agents running in a given operating environment, and this might affect which conventions would be preferred.

Restricted functionality and conformance

User agents do not conform to this document on a per-resource basis; claims are not as specific as "the user agent conforms for this particular Web page." A user agent conforms if it satisfies the requirements of this document for most general-purpose content, in ordinary operating conditions.

In some cases, an author may wish to limit the user agent's functionality for specific reasons, such as to protect intellectual property rights, for security reasons, or to provide a read-only view (allowing no user interaction). Content that limits the functionality of the user agent in some cases does not automatically invalidate a conformance claim. A valid conformance claim remains valid as long as the user agent is capable of satisfying the requirements of the document (i.e., the functionalities have been implemented), and does so for most 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. Digital rights management refers to methods of describing and perhaps enforcing intellectual property associated with Web resources.

3.4 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 synchronized speech.

3.5 Content type labels

Each content type label defines a set of requirements related to support for images, video, animations generally, 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. To conform, the user agent must support visually rendered text.

ColorText
: This content type label refers to all of the requirements related to text foreground and background color for the following checkpoint: 10.4. To conform, the user agent must support more than one text foreground color and more than one text background color.

Image
: This content type label refers to all of the requirements related to images (excluding animated images) for the following checkpoints: 3.1, and 3.7. To conform, the user agent must implement at least one image format. The image requirements apply to content that is recognized as distinct and that, according to the encoding format, may be rendered as a coherent unit.

Animation
: This content type label refers to all of the requirements related to animations (including video and animated images) for the following checkpoints: 3.2, 4.4, 4.5, 4.7, and 4.8. To conform, the user agent must implement at least one animation format. The animation requirements apply to animation content that is recognized as distinct and that, according to the encoding format, may be rendered as a coherent unit.

Video
: This content type label refers to all of the requirements related to video for the following checkpoints: 2.5, 2.6, and 3.2. To conform, the user agent must implement at least one video format. The video requirements apply to video content that is recognized as distinct and that, according to the encoding format, may be rendered as a coherent unit.

Audio
: This content type label refers to all of the requirements related to audio for the following checkpoints: 2.5, 2.6, 3.2, 4.4, 4.5, 4.7, 4.8, 4.9, 4.10, and 4.11 To conform, the user agent must implement at least one audio format. The audio requirements apply to audio content that is recognized as distinct and that, according to the encoding format, may be rendered as a coherent unit.

Speech
: This content type label refers to all of the requirements related to synthesized speech for the following checkpoints: 4.12, 4.13, 4.14, 4.15, and 4.16. 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 8.2.

3.6 Input modality labels

Each input modality label defines a set of requirements related to support for pointing device and voice input. Input device requirements in this document are either stated generically (e.g., "input configuration" requirements) or as keyboard-specific requirements (e.g., "keyboard API").

Pointer
: This input modality label refers to all of the input device requirements of this document, applied to pointing device input. For keyboard-specific requirements, substitute "pointing device input" for "keyboard." The set of pointing device input requirements does not include the requirements of checkpoint 11.4.

Voice
: This input modality label refers to all of the input device requirements of this document, applied to voice input. For keyboard-specific requirements, substitute "voice input" for "keyboard." The set of voice input requirements does not include the requirements of checkpoint 11.4.

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

3.7 Selection label

This document does not require the user agent to implement a selection mechanism in order to conform. However, if the user agent does implement a selection mechanism, in order to conform it must satisfy the relevant portions of the following checkpoints: 6.5, 7.1, 9.4, 10.2, 10.3, and 5.4. The Selection label refers to the selection requirements of these checkpoints.

If a user agent does not implement a selection mechanism, then a well-formed claim must say so.

Note: This document does require implementation of both content focus and user interface focus ; see checkpoint 9.1 and checkpoint 9.2.

3.8 Checkpoint applicability

A checkpoint (or part 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, associated

conditional content , 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 specified mechanisms for conditional content . HTML user agents are not expected to recognize that a nearby paragraph is a text equivalent for the image (when not marked up as such).

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 encoded in an unrecognized XML namespace;
   - 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 unrecognized in a script. Some input device behavior may be controlled by scripts in a manner that the user agent cannot recognize. For example, when the author uses event bubbling to dispatch events, the user agent is not likely to recognize the full set of elements that may receive those events; the user agent is expected to recognize which element has the explicitly associated event handler .

3.9 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-UAAG10-20010831.
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 user agent name and version information. Version information must be sufficient to identify the user agent (e.g., vendor name, version number, minor release number, required patches or updates, natural language of the user interface or documentation). The version information may refer to a range of user agents (e.g., "this claim refers to all user agents version 6.x").
   - The name and version number of the operating environment(s) in which the user agent is running.
   - If a conformance icon is part of a claim on the Web, it must link to the W3C explanation of the icon.

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

1. Content type labels . Content type labels are used in assertions that the subject either (1) does not satisfy the requirements associated with the label (e.g., for a specific checkpoint, for any checkpoint, etc.), or (2) does satisfy the requirements associated with the label (e.g., for a particular format when satisfying the requirements of a checkpoint). In order to conform conditionally, a user agent must satisfy the requirements of at least one content type label.
2. Input modality labels . Each input modality label ("Pointer" or "Voice") is an assertion that the user agent satisfies the requirements associated with the label.
3. Selection label . If the user agent does not implement a selection mechanism, the claim must say so.
4. A list of requirements (checkpoints or portions of checkpoints) that the claim asserts do **not** apply .

Condition 3: At least one version of the claim must conform to the "Web Content Accessibility Guidelines 1.0" *[WCAG10]* , level A. This claim may appear on the Web, on a CD-ROM, etc.

A well-formed claim should also include the following information:

- Information about which specifications have been implemented to satisfy the requirements of the document (e.g., those of guideline 6 and guideline 8).
- Rationale for any requirements that do not apply .

This specification imposes no restrictions 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]* .

3.10 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 the default set of requirements , in addition to (or except) those requirements added (or exempted) by the allowable mechanisms: conformance levels , content type labels , input modality labels , and applicability .

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

Notes:

- The subject of the claim may consist of more than one software component, and taken together they must satisfy all requirements that are not excluded through the claim. These components may run on the user's computer or on a server.

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 6 as long as the subject of the claim as a whole makes *all content* available through those APIs.

- The document has been designed so that non-experts can determine the validity of a claim. In some cases, a requirement might be clear, but without documentation or feedback from developers (e.g., about implemented APIs), it may be difficult to verify that the subject of the 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 open to interpretation than others.
- Ideally, the 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, etc.

Responsibility for claims

This specification imposes no restrictions about:

- who may make a claim (e.g., vendors about their own user agents, third parties about those user agents, journalists about user agents, etc.), or
- where claims may be published (e.g., on the Web or in paper 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.

Conformance icons

As part of a conformance claim, people may use a conformance icon (or, "conformance logo") on a Web site, on user agent packaging, in documentation, etc. A conformance icon does not represent that a claim is valid , only that a claim has been made. The appearance of a conformance icon does not imply that W3C has reviewed the claim.

It is inappropriate and meaningless to use a conformance icon on its own, i.e., to use the icon without an associated 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

This glossary is normative . Some terms (or parts of explanations of terms) may not have an impact on conformance.

Note: In this document, glossary terms generally link to the corresponding entries in this section. These terms are also highlighted through style sheets and identified as glossary terms through markup.

Activate

In this document, the verb "to activate" means (depending on context) either:

- To execute or carry out one or more behaviors associated with an enabled element .
- To execute or carry out one or more behaviors associated with a component of the user agent user interface .

The effect of activation depends on the type of enabled element or user interface control. For instance, when a link is activated, the user agent generally retrieves the linked Web resource . When a form element is activated, it may change state (e.g., check boxes) or may take user input (e.g., a text entry field).

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.

Animation

In this document, an "animation" refers to content that, when rendered, creates a visual movement effect automatically (i.e., without manual user interaction). This definition of animation includes video and animated images. Animation techniques include:

- graphically displaying a sequence of snapshots within the same region (e.g., as is done for video and animated images). The series of snapshots may be provided by a single resource (e.g., an animated GIF image) or from distinct resources (e.g., a series of images downloaded continuously by the user agent).
- scrolling text (e.g., achieved through markup or style sheets).
- displacing graphical objects around the viewport (e.g., a picture of a ball that is moved around the viewport giving the impression that it is bouncing off of the viewport edges). For instance, the SMIL 2.0 *[SMIL20]* animation modules explain how to create such animation effects in a declarative manner (i.e., not by composition of successive snapshots).

Applet

An applet is a program (generally written in the Java programming language) that is part of content , and that the user agent executes.

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

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

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 conventional APIs 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.

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 "conventional input/output API" is one that is expected to be implemented by software running on a particular operating environment. For example, on desktop computers today, the conventional input APIs are for the mouse and keyboard. For touch screen devices or mobile devices, conventional input APIs may include stylus, buttons, voice, etc. The graphical display and sound card are considered conventional ouput devices for a graphical desktop computer environment, and each has an associated 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 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.
- voice 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. For example, the following very general definition of "assistive technology device" comes from the (U.S.) Assistive Technology Act of 1998 *[AT1998]* :

> Any item, piece of equipment, or product system, whether acquired commercially, modified, or customized, that is used to increase, maintain, or improve functional capabilities of individuals with disabilities.

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

In this document, the term "audio" refers to content that encodes pre-recorded sound.

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 narration.

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 (sometimes, "audio description") is either a prerecorded human voice or a synthesized voice (recorded or generated dynamically) describing the key visual elements of a movie or other animation. The auditory description is synchronized with (and possibly included as part of) 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 content (e.g., style sheets within a document, that are associated with a document, or that are generated by a server).

Captions

Captions (sometimes, "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 other 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.

Conditional content

Conditional content is content that, by format specification, should be made available to users through the user interface, generally under certain conditions (e.g., based on user preferences or operating environment limitations). Some examples of conditional content mechanisms include:

- The "`alt`" attribute of the `IMG` element in HTML 4. According to section 13.2 of the HTML 4 specification (*[HTML4]*): "User agents must render alternate text when they cannot support images, they cannot support a certain image type or when they are configured not to display images.
- `OBJECT` elements in HTML 4. Section 13.3.1 of the HTML 4 specification (*[HTML4]*) explains the conditional rendering rules of (nested) `OBJECT` elements. The rules select among ordered alternatives according to user preferences or error conditions.
- The `switch` element and test attributes in SMIL 1.0. Sections 4.3 and 4.4, respectively, of SMIL 1.0 *[SMIL]* explain the conditional rendering rules of these features.
- SVG 1.0 *[SVG]* also includes a `switch` element and several attributes for conditional processing.
- The `NOSCRIPT` and `NOFRAMES` elements in HTML 4 *[HTML4]* allow the author to provide content under conditions when the user agent does not

support scripts or frames, or the user has turned off support for scripts or frames.

Specifications vary in how completely they define how and when to render conditional content. For instance, the HTML 4 specification includes the rendering conditions for the "`alt`" attribute, but not for the "`title`" attribute. The HTML 4 specification does indicate that the "`title`" attribute should be available to users through the user interface ("Values of the title attribute may be rendered by user agents in a variety of ways...").

Note: The Web Content Accessibility Guidelines 1.0 requires that authors provide text equivalents for non-text content. This is generally done by using the conditional content mechanisms of a markup language. Since conditional content may not be rendered by default, the current document requires the user agent to provide access to unrendered conditional content (checkpoint 2.3 and checkpoint 2.9) as it may have been provided to promote accessibility.

Configure, ***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 needs to 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).

A ***global configuration*** is one that applies across elements of the same Web resource, as well as across Web resources. A global configuration may be implemented by more than one setting (e.g., per component of the user agent). For instance, when a user agent consists of a browser that renders HTML and a plug-in that renders SVG, to satisfy the global configuration requirements of this document, the browser may provide one setting and the plug-in another.

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" refers to a component of the user agent user interface .

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 .

Empty content is either a null value or a string consisting of zero characters. For instance, in HTML, "`alt=''`" sets the value of the "`alt`" attribute to the empty string. In some markup languages, an element may have empty content (e.g., the `HR` element in HTML).

Device-independence

Device-independence refers to the ability to make use of software with any appropriate supported input or output device.

Document object, ***Document Object Model*** (DOM)

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- or schema-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 kinds of data through the user interface. Most of the requirements of this document apply to the document object after its construction. However, a few checkpoints (e.g., checkpoint 2.7 and checkpoint 2.11) may affect the construction of the document object.

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 (a 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, ***text source***

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", or as the result of viewing a resource on the local file system). The document source generally refers to the "payload" of the user agent's request, and doesn't generally include information exchanged as part of the transfer protocol. The document source is data that is prior to any repair by the user agent (e.g., prior to repairing invalid markup). "Text source" refers to document source that is composed of text .

Documentation

Documentation refers to information that supports the use of a user agent. This information may be found in manuals, installation instructions, the help system, tutorials, etc. Documentation may be distributed (e.g., some parts may be delivered on CD-ROM, others on the Web). Refer to guideline 12 for information about documentation requirements.

Element, ***element type***,

This document uses the terms "element" and "element type" in the sense employed by the XML 1.0 specification (*[XML]* , section 3): an element type is a syntactic construct of a Document Type Definition (DTD) for its application. This sense is also relevant to structures defined by XML schemas. The document also uses the term "element" more generally to mean a type of content (such as video or sound) or a logical construct (such as a header or list).

Enabled element, ***disabled element***

An enabled element is a piece of content with associated behaviors that may be activated through the user interface or through an API . The set of elements that a user agent enables is generally derived from, but is not limited to, the set of interactive elements defined by implemented markup languages.

Some elements may only be enabled elements for part of a user session. For instance, an element may be disabled by a script as the result of user interaction. Or, an element may only be enabled 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 disable some elements.

A disabled element is a piece of content that is potentially an enabled element, but is not in the current session. Generally, disabled elements will be interactive elements that are not enabled in the current session. This document distinguishes disabled elements (not currently enabled) from non-interactive elements (never enabled).

For the requirements of this document, user selection does not constitute user interaction with enabled elements. See the definition of content focus .

Note: Enabled and disabled elements come from content; they are not part of the user agent user interface .

Note: The term "active element" is not used in this document since it may suggest several different concepts, including: interactive element, enabled element, an element "in the process of being activated" (which is the meaning of ':active' in CSS2 *[CSS2]* , for example).

Equivalent (for content)

The term "equivalent" is used in this document as it is used in the Web Content Accessibility Guidelines 1.0 *[WCAG10]* :

> Content is "equivalent" to other content when both fulfill essentially the same function or purpose upon presentation to the user. In the context of this document, the equivalent must fulfill essentially the same function for the person with a disability (at least insofar as is feasible, given the nature of the disability and the state of technology), as the primary content does for the person without any disability.

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.).

Each markup language defines its own mechanisms for specifying conditional content , and these mechanisms may be used by authors to provide text 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. An event handler is "***explicitly associated with an element***" when the event handler is associated with that element through markup or the DOM. The term "***event bubbling***" describes a programming style where a single event handler dispatches events to more than one element. In this case, the event handlers are not explicitly associated with the elements receiving the events (except for the single element that dispatches the events).

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 this document, the term "explicit user request" refers to any user interaction with a control provided by the user agent user interface (**not** those in content), the focus , or selection . Control behavior should be documented .

Some examples of explicit user requests include when the user selects "New viewport", responds "Yes" to a prompt in the user agent's user interface, configures the user agent to behave in a certain way, or changes the selection or focus with the keyboard or pointing device.

Note: Users make mistakes. For example, a user may inadvertently respond "yes" to a prompt when they meant "no." 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 is 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***

In this document, the term "content focus" refers to a user agent mechanism that satisfies all of the following properties:

1. It designates zero or one element in content that is either enabled or disabled . (In general, the focus should only designate enabled elements, but it may also designate disabled elements.)
2. The user may "set" content focus (programmatically or through the user interface) on an enabled element without triggering the associated behaviors.
3. It has state. The user may prefer to always move the content focus manually from one element to another.
4. It may be used (programmatically or through the user interface) to trigger the behaviors associated with an enabled element. This is generally

implemented by making the focus respond to input device events (often just keyboard events).

User interface mechanisms may resemble content focus, but do not satisfy all of the properties. For example, text editors often implement a "caret" that indicates the current location of text input or editing. The caret may have state and may respond to input device events, but it does not enable users to activate the behaviors associated with enabled elements.

The user interface focus shares the properties of the content focus except the first: the user interface focus designates zero or one control of the user agent user interface that has associated behaviors (e.g., radio button, text box, menu, etc.).

On the screen, the content focus may be highlighted using colors, fonts, graphics, magnification, etc. The content focus may also be highlighted when rendered as synthesized speech, for example through changes in speech prosody. The dimensions of the rendered content focus may exceed those of the viewport.

In this document, each viewport is expected to have at most one content focus and at most one user interface focus. This document includes requirements for content focus only, for user interface focus only, and for both. When a requirement refers to both, the term "focus" is used.

When several viewports coexist, at most one viewport's content focus **or** user interface focus responds to input events; this is called the current focus.

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. Graphical highlight mechanisms include dotted boxes, underlining, and reverse video. Synthesized speech highlight mechanisms include alterations of voice pitch and volume ("speech prosody").

Image
In this document, an "image" refers to content that encodes static (i.e., unmoving) visual information. See also the definition of animation .

Input configuration
An input configuration is the mapping of user agent functionalities to some user interface input 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 documented (per checkpoint 12.3]). Input configurations may be affected by author-specified bindings (e.g., through the "accesskey" attribute of HTML 4 *[HTML4]*).

Interactive element, ***non-interactive element***,
An interactive element is piece of content that, by specification, may have associated behaviors to be executed or carried out as a result of user or

programmatic interaction. For instance, the interactive elements of HTML 4 *[HTML4]* include: links, image maps, form elements, elements with a value for the "longdesc" attribute, and elements with event handlers explicitly associated with them (e.g., through the various "on" attributes). The role of an element as an interactive element is subject to applicability . A non-interactive element is an element that, by format specification, does not have associated behaviors. The expectation of this document is that interactive elements become enabled elements in some sessions, and non-interactive elements never become enabled elements.

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 .

Normative, ***informative***

As used in this document, the term "normative" refers to "that on which the requirements of this document depend for their most precise statement." What is normative is required for conformance (though the conformance scheme of this document allows claimants to exempt certain normative provisions as long as the claim discloses the exemption). What is identified as "informative" (sometimes, "non-normative") is never required for conformance.

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.

Override

In this document, the term "override" means that one configuration or behavior preference prevails over another. Generally, the requirements of this document involve user preferences prevailing over author preferences and user agent default settings and behaviors. Preferences may be multi-valued in general (e.g., the user prefers blue over red or yellow), and include the special case of two values (e.g., turn on or off blinking text content).

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, images, and audio cues.

This document includes requirements that the user be able to view the original author-supplied content associated with a placeholder. To satisfy these requirements, the user agent might render the content in place of the placeholder or in a separate viewport (leaving the placeholder as is). A request

to view the original content associated with a placeholder is considered an explicit user request to render that content.

This document does not require user agents to include placeholders in the document object . A placeholder that is 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).

Plug-in

A plug-in is a program that runs as part of the user agent and that is *not* part of content . Users generally choose to include or exclude plug-ins from their user agent.

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 the viewport, but it may exceed the spatial or temporal dimensions of the viewport (see the definition of rendered content for more information about viewport dimensions). 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 determine the point of regard in a number of ways, including based on viewport position in content, content focus , selection , etc. 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 prompt me before doing X, always do X without prompting me, never do X but tell me when you could have, never do X and never tell me that you could have, 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, synthesized 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 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, behaviors, style, meaning encoded in a script , and markup in an unfamiliar XML namespace may not be recognized by the user agent as easily or at all. 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 that the user agent makes available to the user's senses of sight and hearing (and only those senses for the purposes of this document). Any content that causes an effect that may be perceived through these senses constitutes rendered content. This includes text characters, images, style sheets, scripts, and anything else in content that, once processed, may be perceived through sight and hearing.

The term "rendered text" refers to text content that is rendered in a way that communicates information about the characters themselves, whether visually or as synthesized speech.

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 conditional content that is required by format specification, 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."

Information encoded in scripts may be difficult for a user agent to recognize 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).

Selection*, *current selection

In this document, the term "selection" refers to a user agent mechanism for identifying a range of content (e.g., text, images, etc.). Generally, user agents limit selection to text content (e.g., one or more fragments of text). The selection may be structured (based on the document tree) or unstructured (e.g., text-based). The range may be empty.

On the screen, the selection may be highlighted using colors, fonts, graphics, magnification, etc. The selection may also be highlighted when rendered as synthesized speech, for example through changes in speech prosody. The dimensions of the rendered selection may exceed those of the viewport.

The selection may be used for a variety of purposes: for cut and paste operations, to designate a specific element in a document for the purposes of a query, as an indication of point of regard , etc.

The selection has state. It may be set programmatically or through the user interface.

In this document, each viewport is expected to have at most one selection. When several viewports coexist, at most one viewport's selection responds to input events; this is called the current selection.

See the section on the selection label for information about implementing a selection and conformance .

Note: Some user agents may also implement a selection for designating a range of information in controls of the user agent user interface . The current document only includes requirements for a content selection mechanism.

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 conformance requirements to other specifications (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***

As used in this document a "text element" adds text characters to either content or the user interface . Both in the Web Content Accessibility Guidelines 1.0

[WCAG10] and in this document, text elements are presumed to produce text that can be understood when rendered visually, as synthesized speech, or as Braille. Such text elements benefit at least these three groups of users:

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

A text element may consist of both text and non-text data. For instance, a text element may contain markup for style (e.g., font size or color), structure (e.g., heading levels), and other semantics. The essential function of the text element should be retained even if style information happens to be lost in rendering.

A user agent may have to process a text element in order to have access to the text characters. For instance, a text element may consist of markup, it may be encrypted or compressed, or it may include embedded text in a binary format (e.g., JPEG).

"Text content" is content that is composed of one or more text elements. A "text equivalent" (whether in content or the user interface) is an equivalent composed of one or more text elements. Authors generally provide text equivalents for content by using the conditional content mechanisms of a specification.

A "non-text element" is an element (in content or the user interface) that does not have the qualities of a text element. "Non-text content" is composed of one or more non-text elements. A "non-text equivalent" (whether in content or the user interface) is an equivalent composed of one or more non-text elements.

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). Both text elements and non-text elements should be understood as "pre-rendering" content in contrast to the "post-rendering" content that they produce.

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 other 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 voice-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 (e.g., menus, buttons, prompts, etc.) and mechanisms (e.g., selection and focus) provided by the user agent ("out of the box") that are not created by content .
2. the "content user interface", i.e., the enabled elements that are part of content, such as form elements, 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

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

Visual track

A visual object is content rendered through a graphical viewport . Visual objects include graphics, text, and visual portions of movies and other animations. A visual track is a visual object that is intended as a whole or partial presentation. A visual track does not necessarily correspond to a single physical object or software object. A visual track may be text-based or graphic. A visual track may be static or involve animation .

Views, ***viewports***

The user agent renders content through one or more viewports. Viewports include windows, frames, pieces of paper, loudspeakers, virtual magnifying glasses, etc. A viewport may contain another viewport (e.g., nested frames). User interface controls such as prompts, menus, alerts, etc. are not viewports.

When the dimensions (spatial or temporal) of rendered content exceed the dimensions of the viewport (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.), the user agent provides mechanisms such as scroll bars and advance and rewind controls so that the user can access the rendered content "outside" the viewport.

When several viewports coexist, only one has the current focus at a given moment. This viewport is highlighted to make it stand out.

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); refer to 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.

Note: In this document, bracketed labels such as "[HTML4]" link to the corresponding entries in this section. These labels are also identified as references through markup.

5.1 How to refer to this document

There are two recommended ways to refer to the "User Agent Accessibility Guidelines 1.0" (and to W3C documents in general):

1. References to a specific version of "User Agent Accessibility Guidelines 1.0". For example, use the "this version" URI to refer to the current document: http://www.w3.org/WAI/UA/WD-UAAG10-20010831/.
2. References to the latest version of "User Agent Accessibility Guidelines 1.0". Use the "latest version" URI to refer to the most recently published document in the series: http://www.w3.org/WAI/UA/UAAG10/.

In almost all cases, references (either by name or by link) should be to a specific version of the document. W3C will make every effort to make this document indefinitely available at its original address in its original form. The top of this document includes the relevant catalog metadata for specific references (including title, publication date, "this version" URI, editors' names, and copyright information).

An XHTML 1.0 *[XHTML10]* paragraph including a reference to this specific document might be written:

```
<p>
<cite><a href="http://www.w3.org/WAI/UA/WD-UAAG10-20010831/">
"User Agent Accessibility Guidelines 1.0"</a></cite>,
I. Jacobs, J. Gunderson, E. Hansen, eds.,
W3C Working Draft, 31 August 2001.
The <a href="http://www.w3.org/WAI/UA/UAAG10/">latest
version</a> of this document is available at
http://www.w3.org/WAI/UA/UAAG10/.</p>
```

For very general references to this document (where stability of content, anchors, etc., is not required), it may be appropriate to refer to the latest version of this document. In this case, please use the "latest version" URI at the top of this document.

See also information about making conformance claims to this document.

5.2 Normative references

[CSS1]
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. This W3C Recommendation is http://www.w3.org/TR/1999/REC-CSS1-19990111.

[CSS2]
"CSS, level 2 Recommendation", B. Bos, H. Wium Lie, C. Lilley, and I. Jacobs, eds., 12 May 1998. This W3C Recommendation is http://www.w3.org/TR/1998/REC-CSS2-19980512/.

[DOM2CORE]
"Document Object Model (DOM) Level 2 Core Specification", 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/.

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

[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/.

5.3 Informative references

Some of the references in this section become normative if they are used to satisfy the requirements of guideline 6 and guideline 8.

[AT1998]
The *Assistive Technology Act of 1998, 13 November 1998, United States P.L. 105-394*.

[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/

[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/.

[MATHML20]
"Mathematical Markup Language (MathML) Version 2.0", D. Carlisle, P. Ion, R. Miner, N. Poppelier, et al., 21 February 2001. This W3C Recommendation is http://www.w3.org/TR/2001/REC-MathML2-20010221/.

[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/.

[SMIL20]
Synchronized Multimedia Integration Language (SMIL 2.0) Specification, J. Ayars, et al., eds., 7 August 2001. This W3C Recommendation is http://www.w3.org/TR/2001/REC-smil20-20010807/.

[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-SUMMARY]
An appendix to this document provides a summary of the goals and structure of

User Agent Accessibility Guidelines 1.0.

[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 3.1". This technical report of the Unicode Consortium is available at http://www.unicode.org/unicode/reports/tr27/. This is a revision of "The Unicode Standard, Version 3.0", The Unicode Consortium, 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), Harvey Bingham, Al Gilman, Jon Gunderson (Chair of the Working Group, University of Illinois, Urbana-Champaign), Eric Hansen (Educational Testing Service), Ian Jacobs (Team Contact, W3C), Tim Lacy (Microsoft), Charles McCathieNevile (W3C), David Poehlman, Mickey Quenzer, Gregory Rosmaita (Visually Impaired Computer Users Group of New York City), and Rich Schwerdtfeger (IBM).

Many thanks to the following people who have contributed through review and past participation in the Working Group: Paul Adelson, Jonny Axelsson, Kitch Barnicle, Olivier Borius, Judy Brewer, Dick Brown, 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, Marja-Riitta Koivunen, Peter Korn, Josh Krieger, Catherine Laws, Aaron Leventhal, Greg Lowney, Susan Lesch, Scott Luebking, William Loughborough, Napoleon Maou, Peter Meijer, Karen Moses, Masafumi Nakane, Mark Novak, Charles Oppermann, Mike Paciello, David Pawson, Michael Pederson, Helen Petrie, Michael Pieper, Richard Premack, Jan Richards, Hans Riesebos, Joe Roeder, Lakespur L. Roca, Madeleine Rothberg, 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.