User Agent Accessibility Guidelines 1.0

W3C Proposed Recommendation 10 March 2000

This version:
http://www.w3.org/TR/2000/PR-UAAG10-20000310
(plain text, gzip PostScript, PDF, gzip tar file of HTML, zip archive of HTML)
Latest version:
http://www.w3.org/TR/UAAG10
Previous version:
http://www.w3.org/TR/2000/CR-UAAG10-20000128

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

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

Abstract

The guidelines in this document explain to developers how to design user agents that are accessible to people with disabilities. User agents include graphical desktop browsers, multimedia players, text browsers, voice browsers, plug-ins, and other assistive technologies that provide access to Web content . While these guidelines primarily address the accessibility of general-purpose graphical user agents, the principles presented apply to other types of user agents as well. Following these principles will help make the Web accessible to users with disabilities and will benefit all users.

This document is part of a series of accessibility documents published by the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C).

Status of this document

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

This is the 10 March 2000 Proposed Recommendation of User Agent Accessibility Guidelines 1.0, for review by W3C Members and other interested parties. W3C Advisory Committee Members are invited to send formal review comments to wai-ua-review@w3.org, visible only to the W3C Team, until 7 April 2000. This revision reflects resolutions to issues raised during the Candidate Recommendation review period. A history of changes to this document is available on the Web.

Note. Three checkpoints in this document (checkpoint 5.1, checkpoint 5.2, and checkpoint 5.4) refer to the W3C DOM Level 2 [DOM2] specification, which is a Candidate Recommendation as of 10 March 2000. The User Agent Guidelines Working Group will be tracking dependencies on that specification as it advances to Proposed Recommendation.

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

The public is invited to send comments about this document to the public mailing list w3c-wai-ua@w3.org (public archives).

This document has been produced as part of the Web Accessibility Initiative. The goals of the User Agent Working Group are described in the charter. A list of the Working Group participants is available.

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

Table of contents

Abstract 1
Status of this document 1
1. Introduction 4
1.1 Benefits of accessible design 4
1.2 Principles of accessible design 5
1.3 How the guidelines are organized 7
1.4 Related resources 8
1.5 Document conventions 8
1.6 Priorities 8
1.7 Conformance 9
2. User agent accessibility guidelines 12
1. Support input and output device-independence. 12
2. Ensure user access to all content. 13
3. Allow the user to turn off rendering or stop behavior that may reduce accessibility. 14
4. Ensure user control of styles. 15
5. Observe system conventions and standard interfaces. 17
6. Implement accessible specifications. 19
7. Provide navigation mechanisms. 19
8. Orient the user. 21
9. Notify the user of content and viewport changes. 22
10. Allow configuration and customization. 23
11. Provide accessible product documentation and help. 24
3. Appendix: Glossary 26
4. Acknowledgments 34
5. References 35

An appendix to this document [UAAG10-CHECKLIST] lists all checkpoints for convenient reference.

1. Introduction

This introduction (section 1) provides context for understanding the guidelines listed in section 2 . In different sections, the introduction explains:

- The importance and benefits of accessible user agent design ,
- The principles of accessible user agent design behind the guidelines,
- A description of how each guideline is organized and presented,
- A list of related resources to help developers implement the guidelines,
- Some of the editorial conventions used in this document ,
- How requirements in this document are prioritized,
- How people make claims that software conforms to these guidelines and details about the applicability of the requirements to different kinds of user agents.

1.1 Benefits of accessible design

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

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

User agents must be designed to take into account the diverse requirements of users with disabilities. This document specifies requirements that user agent developers must satisfy to ensure accessibility of the user agent.

Software that follows the guidelines in this document will not only benefit users with disabilities, it will be more flexible, manageable, extensible, and beneficial to all users. Many users browse the Web with requirements similar to those of users with disabilities. For instance:

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

The guidelines in this document describe some basic principles of accessible design. As the previous examples illustrate, accessible design generally benefits all users.

1.2 Principles of accessible design

This document is organized according to several principles that, if followed, will improve the design of any type of user agent:

Ensure that the user interface is accessible

A user with a disability must have access to all the functionalities offered by the user agent through its user interface . Since some users cannot use some parts of the user interface, it needs to be adaptable to their particular needs. To ensure the accessibility of the user interface, people with disabilities should be involved in its design and testing.

One requirement is that users be able to operate the user interface with a variety of input devices (mouse, keyboard, speech input, etc.) and output devices (graphical display, speech output, Braille display, etc.). Redundant input and output methods (accomplished through the standard input and output Application Programming Interfaces (APIs) supported by the user agent) help users operate controls of the user agent as well as those included as part of content .

Another requirement is that users have two operational approaches to the user interface:

- Contextual access (e.g., through cascading menus, through help systems, etc.) may help users with cognitive disabilities and any users unfamiliar with the tool.
- Direct access (e.g., through keyboard or voice shortcuts) helps some users with physical disabilities and speeds up use by experienced users. Such access can be vital for users with physical disabilities who may have little or no manual dexterity, and/or increased tendency to push unwanted buttons or keys.

In order for people to use the user agent at all, the installation procedure (and any subsequent software update procedures) must be accessible according to the guidelines of this document. For example, the user agent must provide device-independent access and accessible documentation of the installation procedure. Furthermore, it is important to maintain consistency in the user interface between versions of the software. In particular, developers should make changes conservatively to the layout of user interface controls , the behavior of existing functionalities, and the default keyboard configuration. Consistency is not a rigid requirement, however, and is less important than improved general accessibility and usability.

This document includes a number of user interface requirements that are similar to, or related to, general guidelines for user interface design. 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 Document [UAAG10-TECHS] includes some references to general software design guidelines and platform-specific accessibility guidelines.

Note. This document addresses accessible user agent support for some language features (e.g., tables for layout, etc.) that may be widely deployed, but whose use may be discouraged.

Ensure that the user has access to content

User agents must ensure access to content :

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

Help orient the user

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

- Browsing context. This includes information about the number of frames, the title of the current frame, whether loading for a page or video clip has finished or stalled, etc., Graphical clues about browsing context (such as frames, proportional scroll bars, a visually highlighted selection, etc.) help some, but not all users, so the context information must be available in a device-independent manner.
- Element context. This includes information about specific elements (e.g., the dimensions of a table, the length of an audio clip the structure of a form, etc.) and surrounding information. For instance, users who are blind and who may navigate by jumping from link to link on a page or presentation will benefit from nearby information that helps them decide quickly whether to follow the link, as well as from metadata about the link: whether it has been visited, the type of the target resource, the length of an audio or video clip that will be started, whether activating the link involves a fee, etc.

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

- For changes to the content or viewport that the user does not initiate, allow the user to request notification when these changes occur (e.g., when a viewport

opens, a script is executed, etc.).
- Allow the user to return to a known state (e.g., by providing browsing history mechanism).

Follow operating system standards and conventions and use open specifications

Following platform and operating system standards and guidelines promotes accessibility, usability, and predictability. So that desktop browsers can make information available to assistive technologies , they must communicate through standard interfaces. An architecture that makes information about content and the user interface available programmatically will benefit assistive technologies, scripting tools, and automated test engines. It will also promote software modularity and reuse.

1.3 How the guidelines are organized

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

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

Each checkpoint definition includes:

- The checkpoint number.
- The statement of the checkpoint.
- The priority of the checkpoint.
- Optional informative notes, clarifying examples, and cross references to related guidelines or checkpoints.
- A link to a corresponding section of "Techniques for User Agent Accessibility Guidelines 1.0" [UAAG10-TECHS] , where the checkpoint is examined in detail, including information about implementation and examples.

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

This document includes as an appendix a glossary. Another appendix lists all checkpoints in tabular and linear format for convenient reference [UAAG10-CHECKLIST] .

1.4 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. Readers are strongly encouraged to become familiar with the Techniques Document. Note that the Techniques provided are informative examples only, and other strategies may be used to meet the checkpoint as well as, or in place of, those listed therein. The Techniques Document is expected to be updated more frequently than the current guidelines.

"User Agent Accessibility Guidelines 1.0" is part of a series of accessibility guidelines published by the Web Accessibility Initiative (WAI). The series also includes "Web Content Accessibility Guidelines 1.0" [WCAG10] and "Authoring Tool Accessibility Guidelines 1.0" [ATAG10] . In addition to this series, WAI provides other resources and educational materials about Web accessibility.

1.5 Document conventions

The following editorial conventions are used throughout this document:

- HTML element names are in uppercase letters (e.g., H1, BLOCKQUOTE, TABLE, etc.)
- HTML attribute names are quoted in lowercase letters (e.g., "alt", "title", "class", etc.)

1.6 Priorities

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

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

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

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

1.7 Conformance

This section explains how to make a valid claim that a user agent conforms to this document. Anyone may make a claim (e.g., vendors about their own products, third parties about those products, journalists about products, etc.). Claims may be published anywhere (e.g., on the Web or in product documentation).

Claimants are solely responsible for their claims and the use of the conformance icons . If the subject of the claim (i.e., the software) changes after the date of the claim, the claimant is responsible for updating the claim. Claimants are encouraged to conform to the most recent guidelines available.

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

Conformance levels

A conformance claim must indicate what conformance level is met:

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

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

Well-formed conformance claims

A well-formed claim must include the following information:

About the guidelines:

- The guidelines title/version: "User Agent Accessibility Guidelines 1.0".
- The URI of the guidelines: http://www.w3.org/TR/2000/PR-UAAG10-20000310.
- The conformance level satisfied: "A", "Double-A", or "Triple-A".
- The checkpoints of the chosen conformance level considered not applicable . Claimants may use the checklist [UAAG10-CHECKLIST] for this purpose.

About the subject of the claim:

- The vendor name.
- The product name and version information (version number, minor release number, and relevant bugfix update level).
- The operating system name and version number.

Properties of the claim:

- The date of the claim.

This information may be provided in text or metadata markup (e.g., using the Resource Description Framework (RDF) [RDF10] and an RDF schema designed for WAI conformance claims). All content in the claim must be accessible according to the Web Content Accessibility Guidelines 1.0 [WCAG10] .

Here is an example of a claim expressed in HTML:

```
<p>On 10 March 2000, this product (version 2.3 on MyOperatingSystem)
conforms to <abbr title="the World Wide Web Consortium">W3C</abbr>'s "User
Agent Accessibility Guidelines 1.0",
http://www.w3.org/TR/2000/PR-UAAG10-20000310, level Double-A. The <a
href="http://somewhere.com/checkpoints"> list of checkpoints that do not
apply</a> is available online.</p>
```

Validity of a claim

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

1. The claim is well-formed , and
2. The subject of the claim satisfies all the applicable checkpoints for that level.

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

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

Conformance icons

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

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

Checkpoint applicability

Not every checkpoint or guideline is applicable to every user agent. Generally, a user agent must adhere to checkpoints that ensure accessibility of functionalities that it offers to users and it must implement required functionalities natively . If the user agent supports keyboard input, it must support accessible keyboard input. If the user agent supports images, it must ensure access to each image or an equivalent alternative supplied by the author. If a user agent supports style sheets, it must implement the accessibility features of the style sheet language. If the user agent supports frames, it must ensure access to frame alternatives supplied by the author. In short, if a user agent offers a functionality, it must ensure that people with disabilities have access to that functionality or an equivalent alternative.

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

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

To summarize, a checkpoint (or portion of a checkpoint) applies to a user agent unless:

- It refers solely to an unsupported input or output device interface. Note that if the device interface is supported at all, it must be supported accessibly.
- It includes requirements about a content type (script, image, video, sound, applet, etc.) that the user agent either does not recognize or recognizes but does not support natively .
- It requires control of properties of an embedded object (e.g., video or animation rate) that may not be controlled or accessed by the user agent.
- It refers to unsupported technologies that are not required by this document. For instance, all conforming user agents are required to support the W3C Document Object Model [DOM2] . However, user agents are *not* required to support a synchronized multimedia markup language such as SMIL 1.0 [SMIL] . If they do, the checkpoints that refer to synchronized multimedia apply.
- It refers to communication with other software but no communication is possible on the system housing the user agent (e.g., a kiosk with no infrared port for communication with assistive technologies)
- It cannot be satisfied due to hardware or system resource limitations.

2. User agent accessibility guidelines

Guideline 1. Support input and output device-independence.

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

Since people use a variety of devices for input and output, user agent developers must ensure redundancy in the user interface . Messages and alerts to the user must not rely on auditory or graphical cues alone; text, beeps, flashes, and other techniques used together will make these alerts accessible. Text messages are generally accessible since they may be used by people with graphical displays, speech synthesizers, or Braille displays.

People who can't or do not use a mouse must be able to operate the user interface with the keyboard, through voice input, a head wand, touch screen, or other device. *Keyboard operation* of all functionalities offered through the user interface is one of the most important aspects of user agent accessibility on almost every platform. The keyboard is available to most users, it is widely supported, and hooks provided for the keyboard can be used for other types of input.

In addition to ensuring device-independent access to all functionalities, developers must use the standard APIs of the operating system for supported devices. This allows assistive technologies to operate the user agent programmatically by simulating events from a mouse, keyboard, pen, or other input device. For instance, when standard APIs are used, some users who are not able to enter text easily through a standard physical keyboard may still use voice input or an on-screen keyboard to operate the user agent.

Checkpoints for user interface accessibility:

1.1 Ensure that every functionality available through the user interface is also available through every input device API supported by the user agent. Excluded from this requirement are functionalities that are part of the input device API itself (e.g., text input for the keyboard API, pointer motion for the pointer API, etc.) [Priority 1]

> **Note.** The device-independence required by this checkpoint applies to functionalities described by the other checkpoints in this document (e.g., installation, documentation, user agent user interface configuration, etc.). This checkpoint does not require user agents to use all operating system input device APIs , only to make the software accessible through those they do use.

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

> Do not bypass the standard output APIs when rendering information (e.g., for reasons of speed, efficiency, etc.). For example, do not bypass standard APIs to manipulate the memory associated with rendered content , since assistive technologies monitor rendering through the APIs .

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

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

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

Note. This checkpoint is an important special case of checkpoint 1.1. The comment about low-level functionalities in checkpoint 1.1 applies to this checkpoint as well. Refer also to checkpoint 10.8.

1.5 Ensure every non-text message (e.g., prompt, alert, etc.) available through the user interface also has a text equivalent in the user interface. [Priority 1]

Note. For example, if the user interface provides access to a functionality through a graphical button, ensure that the user interface also provides access to the same functionality through a control that includes a text equivalent. If a sound is used to notify the user of an event , announce the event in text on the status bar as well. Refer also to checkpoint 5.7.

Guideline 2. Ensure user access to all content.

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

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

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

Checkpoints for content accessibility:

2.1 Ensure that the user has access to all content , including equivalent alternatives for content . [Priority 1]

Refer to guideline 5 for more information about programmatic access to content.

2.2 For presentations that require user input within a specified time interval, allow the user to configure the time interval (e.g., to extend it or to cause the user agent to pause the presentation automatically and await user input before proceeding). [Priority 1]
2.3 When the author has not supplied a text equivalent for content as required by the markup language, make available other author-supplied information about the content (e.g., object type, file name, etc.). [Priority 2]
2.4 When a text equivalent for content is explicitly empty (i.e., an empty string), render nothing. [Priority 3]

Checkpoints for user interface accessibility:

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

For example, if a multimedia presentation has several captions (or subtitles) available, allow the user to choose from among them. Captions might differ in level of detail, address different reading levels, differ in natural language , etc.

2.6 Allow the user to specify that text transcripts , collated text transcripts , captions , and auditory descriptions be rendered at the same time as the associated auditory and visual tracks. Respect author-supplied synchronization cues during rendering. [Priority 1]
2.7 For author-identified but unsupported natural languages, allow the user to request notification of language changes in content . [Priority 3]

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

Ensure that the user may turn off rendering or stop behavior specified by the author that may reduce accessibility by obscuring content or disorienting the user.

Some content or behavior specified by the author may make the user agent unusable or may obscure information. For instance, flashing content may trigger seizures in people with photosensitive epilepsy, or may make a Web page too distracting to be usable by someone with a cognitive disability. Blinking can affect screen reader users, since screen readers (in conjunction with speech synthesizers or Braille displays) may repeat the text every time it blinks. Distracting background images or sounds make make it impossible for users to see or hear other content. Some color combinations may affect users with some visual disabilities.

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.

Users may need to turn off these effects in order to have access to content. A user agent must provide on/off control even when it hands off content (e.g., a sound file) to the operating system or to a helper application for rendering; the user agent is aware of the content type and thus can choose not to render it. Please also refer to guideline 4 and guideline 10.

Checkpoints for content accessibility:

3.1 Allow the user to turn on and off rendering of background images. [Priority 1]
3.2 Allow the user to turn on and off rendering of background audio. [Priority 1]
3.3 Allow the user to turn on and off rendering of video. [Priority 1]
3.4 Allow the user to turn on and off rendering of audio. [Priority 1]
3.5 Allow the user to turn on and off animated or blinking text. [Priority 1]
3.6 Allow the user to turn on and off animations and blinking images. [Priority 1]
3.7 Allow the user to turn on and off support for scripts and applets. [Priority 1]
Note. This is particularly important for scripts that cause the screen to flicker, since people with photosensitive epilepsy can have seizures triggered by flickering or flashing, particularly in the 4 to 59 flashes per second (Hertz) range.
3.8 For automatic content changes specified by the author (e.g., redirection and content refresh), allow the user to slow the rate of change. [Priority 2]
3.9 Allow the user to turn on and off rendering of images. [Priority 3]

Guideline 4. Ensure user control of styles.

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

Providing access to content (refer to guideline 2) includes enabling users to configure its presentation. Users with low vision may require larger text than the default size specified by the author or the user agent. Users with color blindness may need to impose or prevent certain color combinations. Users with physical or cognitive disabilities may need to configure the rate of a multimedia presentation.

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

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

For more information about configuration, refer to guideline 10.

Note. The checkpoints in this guideline apply to all content, including equivalent alternatives .

Checkpoints for fonts and colors:

4.1 Allow the user to configure the size of text. [Priority 1]
For example, allow the user to specify a font family and style directly through the user agent user interface or in a user style sheet . Or, allow the user to zoom or magnify content.
4.2 Allow the user to configure font family. [Priority 1]
4.3 Allow the user to configure foreground color. [Priority 1]
4.4 Allow the user to configure background color. [Priority 1]

Checkpoints for multimedia and audio presentations:

4.5 Allow the user to slow the presentation rate of audio, video, and animations. [Priority 1]
4.6 Allow the user to start, stop, pause, advance, and rewind audio, video, and animations. [Priority 1]
4.7 Allow the user to configure the position of text transcripts , collated text transcripts , and captions on graphical displays. [Priority 1]
4.8 Allow the user to configure the audio volume. [Priority 2]

Checkpoints for synthesized speech:

4.9 Allow the user to configure synthesized speech playback rate. [Priority 1]
4.10 Allow the user to configure synthesized speech volume. [Priority 1]
4.11 Allow the user to configure synthesized speech pitch, gender, and other articulation characteristics. [Priority 2]

Checkpoints for user interface accessibility:

4.12 Allow the user to select from available author and user style sheets or to ignore them. [Priority 1]
Note. By definition, the browser's default style sheet is always present, but may be overridden by author or user styles.
4.13 Allow the user to configure how the selection is highlighted (e.g., foreground and background color). [Priority 1]
4.14 Allow the user to configure how the content focus is highlighted (e.g., foreground and background color). [Priority 1]
4.15 Allow the user to configure how the focus changes. [Priority 2]
For instance, allow the user to require that user interface focus not move automatically to a newly opened viewport .
4.16 For those viewports , prompts, and windows that open without an explicit request from the user, allow the user to configure how they open. [Priority 2]

Guideline 5. Observe system conventions and standard interfaces.

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

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

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

Using interoperable APIs and following system conventions increases predictability for users and for developers of assistive technologies .

Checkpoints for content accessibility:

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

Note. These modules are defined in DOM Level 2 [DOM2] , chapters 1 and 2. Please refer to that specification for information about which versions of HTML and XML are supported and for the definition of a "read-only" DOM. This checkpoint is an important special case of checkpoint 2.1. For content other than HTML and XML, refer to checkpoint 5.3.

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

For example, if the user interface allows users to complete HTML forms, this must also be possible through the DOM APIs . **Note.** These modules are defined in DOM Level 2 [DOM2] , chapters 1 and 2. Please refer to DOM Level 2 [DOM2] for information about which versions of HTML and XML are supported. This checkpoint is an important special case of checkpoint 2.1. For

content other than HTML and XML, refer to checkpoint 5.3.

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

Note. This checkpoint addresses content not covered by checkpoints checkpoint 5.1 and checkpoint 5.2. This checkpoint is an important special case of checkpoint 2.1.

5.4 Provide programmatic access to Cascading Style Sheets (CSS) by conforming to the W3C Document Object Model (DOM) Level 2 CSS module and exporting the interfaces it defines. [Priority 3]

Note. This module is defined in DOM Level 2 [DOM2] , chapter 5. Please refer to that specification for information about which versions of CSS are supported. This checkpoint is an important special case of checkpoint 2.1.

Checkpoints for user interface accessibility:

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

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

5.6 Implement selection , content focus , and user interface focus mechanisms. [Priority 1]

Refer also to checkpoint 7.1 and checkpoint 5.5. **Note.** This checkpoint is an important special case of checkpoint 5.5.

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

Refer also to checkpoint 5.5.

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

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

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

Refer also to checkpoint 10.2.

Guideline 6. Implement accessible specifications.

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

Developers should implement open and accessible specifications. Conformance to open specifications promotes interoperability and accessibility by making it easier to design assistive technologies (also discussed in guideline 5). W3C specifications promote accessibility for the following reasons:

- W3C specifications include "built-in" accessibility features.
- W3C specifications undergo early review to ensure that accessibility issues are considered during the design phase.
- W3C specifications are developed in a consensus-based process that includes stakeholders in accessibility.

Checkpoints for content accessibility:

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

Note. The Techniques Document [UAAG10-TECHS] addresses the accessibility features of W3C specifications.

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

For instance, for markup, implement HTML 4.01 [HTML4] , XHTML 1.0 [XHTML10] , or XML 1.0 [XML] . For style sheets, implement CSS ([CSS1] , [CSS2]). For mathematics, implement MathML [MATHML] . For synchronized multimedia, implement SMIL 1.0 [SMIL] . For information about programmatic access to HTML and XML content, refer to guideline 5.

Note. For reasons of backward compatibility, user agents should continue to support deprecated features of specifications. The current guidelines refer to some deprecated language features that do not necessarily promote accessibility but are widely deployed. Information about deprecated language features is generally part of the language's specification.

Guideline 7. Provide navigation mechanisms.

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

Providing a variety of navigation mechanisms helps users with disabilities (and all users) access content more quickly. Content navigation is particularly important to users who access content serially (e.g., as synthesized speech or Braille).

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

Direct navigation (go to a particular link or paragraph, search for instances of a string, etc.) is faster than sequential navigation, but generally requires familiarity with the content. Direct navigation is important to users with some physical disabilities and benefits all "power users." Selecting text or structured content with the pointing device is another form of direct navigation. Searching on text is one important variant of direct navigation.

Structured navigation mechanisms such as navigation of headers, tables, lists, etc., offer both context and speed. For information about programmatic access to document structure, refer to guideline 5.

User agents should allow users to configure navigation mechanisms (e.g., to allow navigation of links only, or links and headers, or tables and forms, etc.). For more information about configuration, refer to guideline 10.

Checkpoints for user interface accessibility:

7.1 Allow the user to navigate viewports (including frames). [Priority 1]
Note. For example, when all frames of a frameset are displayed side-by-side, allow the user to navigate among them with the keyboard. Or, when frames are accessed or viewed one at a time (e.g., by a text browser or speech synthesizer), provide a list of links to other frames. Navigating into a viewport makes it the current viewport .

7.2 For user agents that offer a browsing history mechanism, when the user returns to a previous viewport, restore the point of regard in the viewport . [Priority 1]
For example, when users navigate "back" and "forth" among viewports, they should find the viewport position where they last left it.

7.3 Allow the user to navigate all active elements . [Priority 1]
Navigation may include non-active elements in addition to active elements.
Note. This checkpoint is an important special case of checkpoint 7.6.

7.4 Allow the user to choose to navigate only active elements . [Priority 2]

7.5 Allow the user to search for rendered text content , including rendered text equivalents . [Priority 2]
Note. Use operating system conventions for marking the result of a search (e.g., selection or content focus).

7.6 Allow the user to navigate according to structure. [Priority 2]
For example, allow the user to navigate familiar elements of a document: paragraphs, tables and table cells, headers, lists, etc. **Note.** Use operating system conventions to indicate navigation progress (e.g., selection or content focus).

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

Guideline 8. Orient the user.

Provide information to the user about content structure and metadata to help the user understand browsing context.

All users require clues to help them understand their "location" when browsing. Some mechanisms that provide such clues include:

- Proportional scroll bars on graphical viewports , which indicate how much content has been viewed.
- Highlighted (either graphically or aurally) selection and content focus mechanisms, which distinguish the selected or focused content from other content.
- A history mechanism, which allows users to return to a familiar or comprehensible "state".

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

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

Checkpoints for content accessibility:

8.1 Make available to the user the author-specified purpose of each table and the relationships among the table cells and headers. [Priority 1]
For example, provide information about table headers, how headers relate to cells, table summary information, cell position information, table dimensions, etc. Refer also to checkpoint 5.3. **Note.** This checkpoint is an important special case of checkpoint 2.1.
8.2 Indicate to the user whether a link has been visited. [Priority 2]
Note. Do not use color as the only distinguishing factor between visited and unvisited links as some users may not perceive colors and some devices may not render them. This checkpoint is an important special case of checkpoint 8.4.
8.3 Indicate to the user whether a link has been marked up to indicate that following it will involve a fee. [Priority 2]
Note. This checkpoint is an important special case of checkpoint 8.4. The W3C specification "Common Markup for micropayment per-fee-links" [MICROPAYMENT] describes how authors may mark up micropayment

information in an interoperable manner.

8.4 To help the user decide whether to follow a link, make available link information supplied by the author and computed by the user agent. [Priority 3]

Information supplied by the author includes link content, link title, whether the link is internal, whether it involves a fee, and hints on the content type, size, or natural language of the linked resource. Information computed by the user agent includes whether the user has already visited the link. **Note.** User agents are not required to retrieve the resource designated by a link as part of computing information about the link.

Checkpoints for user interface accessibility:

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

Note. This includes highlighting and identifying frames. **Note.** This checkpoint is an important special case of checkpoint 1.1. Refer also to checkpoint 8.4.

8.6 Make available to the user an "outline" view of content , built from structural elements (e.g., frames, headers, lists, forms, tables, etc.). [Priority 2]

For example, for each frame in a frameset, provide a table of contents composed of headers where each entry in the table of contents links to the header in the document. **Note.** The outline view does not have to be navigable, but if it is, it may satisfy checkpoint 7.6.

8.7 Provide a mechanism for highlighting and identifying active elements (through a standard interface where available). [Priority 2]

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

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

For example, allow the user to configure the level of detail of the outline. Refer also to checkpoint 8.6 and checkpoint 5.5.

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

Note. Refer also to checkpoint 8.4.

Guideline 9. Notify the user of content and viewport changes.

Alert users, in an output device independent fashion, of changes to content or viewports.

For people with visual disabilities or certain types of learning disabilities, it is important that the point of regard remain as stable as possible. Unexpected changes may cause users to lose track of how many viewports are open, which is the current viewport, etc. User agents should notify the user of content and viewport changes caused by dynamic content, or allow users to turn off scripts entirely (refer to checkpoint 3.7).

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

Checkpoints for user interface accessibility:

9.1 Ensure that when the selection or content focus changes, it is in a viewport after the change. [Priority 2]
For example, users navigating links may navigate to a portion of the document outside the viewport, so the viewport should scroll to include the new location of the focus.
9.2 Prompt the user to confirm any form submission triggered indirectly, that is by any means other than the user activating an explicit form submit control. [Priority 2]
For example, do not submit a form automatically when a menu option is selected, when all fields of a form have been filled out, or when a mouseover event occurs.
9.3 Allow the user to configure notification preferences for common types of content and viewport changes. [Priority 3]
For example, allow the user to choose to be notified (or not) that a script has been executed, that a new viewport has been opened, that a pulldown menu has been opened, that a new frame has received focus , etc.
9.4 When loading content (e.g., document, image, audio, video, etc.) indicate what proportion of the content has loaded and whether loading has stalled. [Priority 3]
9.5 Indicate the relative position of the viewport in rendered content (e.g., the percentage of an audio or video clip that has been played, the percentage of a Web page that has been viewed, etc.). [Priority 3]
Note. The user agent may calculate the percentage according to content focus position, selection position, or viewport position, depending on how the user has been browsing.

Guideline 10. Allow configuration and customization.

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

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

Checkpoints for user interface accessibility:

10.1 Provide information to the user about current user preferences for input configurations (e.g., keyboard or voice bindings). [Priority 1]
10.2 Avoid default input configurations that interfere with operating system accessibility conventions. [Priority 1]
In particular, default configurations should not interfere with the mobility access keyboard modifiers reserved for the operating system. For information about system conventions and accessibility settings, refer to checkpoint 5.9.

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

10.4 Allow the user to change the input configuration . [Priority 2]

For voice-activated browsers, allow the user to modify which voice commands activate functionalities. Similarly, allow the user to modify the graphical user agent user interface for quick access to commonly used functionalities (e.g., through buttons). Refer also to checkpoint 10.5 and checkpoint 10.9.

10.5 Allow the user to configure the user agent so that the user's preferred one-step operations may be activated with a single input command (e.g., key stroke, voice command, etc.). [Priority 2]

Note. User agents are not required to provide single command activation of all user agent functionalities at once, only some of them. This checkpoint is an important special case of checkpoint 10.4.

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

For example, on some operating systems, if a functionality is available from a menu, the letter of the key that will activate that functionality is underlined. **Note.** This checkpoint is an important special case of checkpoint 5.9.

10.7 For the configuration requirements of this document, allow the user to save user preferences in a profile . [Priority 2]

Note. This includes user preferences for styles, presentation rates, input configurations , navigation, views, and notification. Users must be able to select from among available profiles or no profile (i.e., the user agent default settings).

10.8 Ensure that frequently used functionalities are easily activated in the default input configuration . [Priority 3]

Make the most frequent operations easy to access and operable through a single command.

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

Note. This checkpoint is an important special case of checkpoint 10.4.

Guideline 11. Provide accessible product documentation and help.

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

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

Features that support accessibility must be clearly documented so that users with disabilities can learn to operate the user agent efficiently. Documentation of keyboard accessibility is particularly important to users with visual disabilities and some types of physical disabilities. Without this documentation, a user with a disability (or multiple disabilities) may not think that a particular task can be

performed. Or the user may try to use a much less efficient technique to perform a task, such as using a mouse, or using an assistive technology's mouse emulation key strokes.

Refer also to checkpoint 5.9.

Checkpoints for user interface accessibility:

11.1 Provide a version of the product documentation that conforms to the Web Content Accessibility Guidelines 1.0 [WCAG10] . [Priority 1]

User agents may provide documentation in many formats, but at least one must conform to the Web Content Accessibility Guidelines 1.0 [WCAG10] .

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

For example, review the documentation or help system to ensure that it includes information about the accessibility features discussed in this document.

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

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

Note. This is a more specific requirement than checkpoint 11.2.

11.5 Document changes between software releases. [Priority 2]

3. Appendix: Glossary

Active element

An active element is an element with behaviors that may be ***activated*** (or "triggered") either through the user interface or through scripts. Which elements are active depends on the document language and whether the features are supported by the user agent. In HTML 4.01 [HTML4] documents, for example, active elements include links, image maps, form controls, element instances with a value for the "longdesc" attribute, and element instances with scripts (event handlers) explicitly associated with them (e.g., through the various "on" attributes). Most systems use the content focus to navigate active elements and identify which is to be activated. An active element's behavior may be triggered through any number of mechanisms, including the mouse, keyboard, an API , etc. The effect of activation depends on the element. For instance, when a link is activated, the user agent generally retrieves the linked resource. When a form control is activated, it may change state (e.g., check boxes) or may take user input (e.g., a text field). Refer also to the definition of event handler .

Application Programming Interface (API)

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

Assistive technology

In the context of this document, an assistive technology is a user agent that relies on one or more other user agents to help people with disabilities interact with a computer. For example, screen reader software is an assistive technology because it relies on browsers or other application 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 text and images.
- screen readers, which are used by people who are blind or have reading disabilities to read textual information through synthesized speech or Braille displays.
- speech recognition software, which may be used by people who have some physical disabilities.
- alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard.
- alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations.

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

Audio presentation

An audio presentation is a stand-alone audio track. Examples of audio presentations include a musical performance, a radio-style news broadcast, and a book reading. When an audio presentation includes natural language, one can create a text equivalent for it (e.g., a text transcript).

Auditory description

An auditory description is either a prerecorded human voice or a synthesized voice (recorded or generated dynamically) describing the key visual elements of a presentation. The auditory description is synchronized with the auditory track of the presentation, usually during natural pauses in the auditory track. Auditory descriptions include information about actions, body language, graphics, and scene changes.

Author styles

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

Captions

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

Collated text transcript

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

Configure

In the context of this document, to configure means to choose, from a set of options, preferences for interface layout, user agent behavior, rendering style, and other parameters required by this document. This may be done through the user agent's user interface , through profiles , style sheets, by scripts, etc. Users should be able to save their configurations across user agent sessions (e.g., in a profile).

Content

In this document, content means the document source, including its ***elements***, ***attributes***, comments, and other features defined by a markup language specification such as HTML 4.01 or an XML application. Refer also to the

definitions of rendered content and equivalent alternatives for content

Control

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

Device-independence

Device-independence refers to the ability to make use of software via the API for any supported input or output device API . User agents should follow operating system conventions and use standard system APIs for device input and output.

Document Object Model (DOM)

A document object model is an interface to a standardized tree structure representation of a document. This interface allows authors to access and modify the document with a scripting language (e.g., JavaScript) in a consistent manner across scripting languages. As a standard interface, a document object model makes it easier not just for authors but for assistive technology developers to extract information and render it in ways most suited to the needs of particular users. The relevant W3C DOM Recommendations are listed in the references .

Documentation

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

Equivalent alternatives for content

Since rendered content in some forms is not always accessible to users with disabilities, authors must supply equivalent alternatives for content. 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. For example, the text "The Full Moon" might convey the same information as an image of a full moon when presented to users. Note that equivalent information focuses on fulfilling the same function. If the image is part of a link and understanding the image is crucial to guessing the link target, an equivalent must also give users an idea of the link target. Equivalent alternatives of content include ***text equivalents*** (long and short, synchronized and unsynchronized) and non-text equivalents (e.g., an auditory description , or a visual track that shows a sign language translation of a written text, etc.). Please also consult the Web Content Accessibility Guidelines 1.0 [WCAG10] and its associated Techniques document [WCAG10-TECHS] . Each markup language defines its own mechanisms for specifying equivalent alternatives. For instance, in HTML 4.01 [HTML4] or SMIL 1.0 [SMIL] , the "alt" attribute specifies alternative text for many elements. In HTML 4.01, authors may provide alternatives 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 alternatives, and NOSCRIPT for script alternatives), and in prose.

Events and scripting, event handler

User agents often perform a task when a certain event occurs, caused by user interaction (e.g., mouse motion or a key press), a request from the operating system, etc. Some markup languages allow authors to specify that a script, called an ***event handler***), be executed when a specific event occurs, such as document loading and unloading, mouse press or hover events, keyboard events, and other user interface events. **Note.** The combination of HTML, style sheets, the Document Object Model , 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.

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

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

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

The term "focus" encompasses both types of focus. Where one is meant specifically in this document, it is identified.

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

Graphical

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

Highlight

A highlight mechanism emphasizes selected or focused content. For example, graphical highlight mechanisms include dotted boxes, underlining, and reverse video. Synthesized speech highlight mechanisms include alterations of voice pitch and volume.

Input configuration

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

Native support

A user agent supports a feature natively if it does not require another piece of software (e.g., plug-in or external program) for support. Operating system features adopted as part of the user agent are considered part of native support. However, since the user agent is responsible for the accessibility of native features, it is also considered responsible for the accessibility of adopted operating system features.

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 4.01 ([HTML4] section 8.1), the "xml:lang" attribute in XML 1.0 ([XML] , section 2.12), the HTML 4.01 "hreflang" attribute for links in HTML 4.01 ([HTML4] , section 12.1.5), the HTTP Content-Language header ([RFC2616] , section 14.12) and the Accept-Language request header ([RFC2616] , section 14.4).

Offscreen model

An offscreen model is rendered content created by an assistive technology that is based on the rendered content of another user agent. Assistive technologies that rely on an offscreen model generally construct it by intercepting standard system drawing calls. For example, in the case of display drivers, some screen readers are designed to monitor what is drawn on the screen by hooking drawing calls at different points in the drawing process. While knowing about the user agent's formatting may provide some useful information to assistive technologies, this document emphasizes access to the document object model rather than a particular rendering. For instance, instead of relying on system calls to draw text, assistive technologies should access the text through the document object model.

Point of regard

The point of regard of a viewport is its position in rendered content . Since users may be viewing rendered content with browsers that render in various ways (graphically , as speech, as Braille, etc.), what is meant precisely by "the point of regard" may vary. Depending on the user agent and browsing context, it may refer to a two dimensional area (e.g., for graphical rendering) or a single point (e.g., for aural rendering or voice browsing). The point of regard may also refer to a particular moment in time for content that changes over time (e.g., an audio presentation). User agents may use the focus , selection , or other means to designate the point of regard. A user agent should not change the point of regard unexpectedly as this may disorient the user.

Profile

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

Properties, values, and defaults

A user agent renders a document by applying formatting algorithms and style information to the document's elements. Formatting depends on a number of factors, including where the document is rendered: on screen, on paper, through speakers, on a Braille display, on a mobile device, etc. Style information (e.g., fonts, colors, voice inflection, etc.) may come from the elements themselves (e.g., certain style attributes in HTML), from style sheets, or from user agent

settings. For the purposes of these guidelines, each formatting or style option is governed by a property and each property may take one value from a set of legal values. Generally in this document, the term "property" has the meaning defined in CSS 2 ([CSS2] , section 3). A reference to "styles" in this document means a set of style-related properties.

The value given to a property by a user agent when it is installed is called the property's ***default value***.

Recognize

A user agent is said to recognize markup, content types, or rendering effects when it can identify the information. Recognition may occur through built-in mechanisms, Document Type Definitions (DTDs) style sheets, headers, and other means. An example of failure of recognition is that HTML 3.2 user agents may not recognize the new elements or attributes of HTML 4.01 [HTML4] . While a user agent may recognize blinking content specified by elements or attributes, it may not recognize blinking in an applet. The Techniques Document [UAAG10-TECHS] lists some markup known to affect accessibility that should be recognized by user agents.

Rendered content

Rendered content is the part of content that is rendered after the application of style sheets, transformations, user agent settings, etc. The content rendered for a given element may be what appears between the element's start and end tags, the value of an attribute (e.g., the "alt", "title", and "summary" attributes in HTML), or external data (e.g., the IMG element or the resourced designated by the "longdesc" attribute in HTML). Content may be rendered to a graphical display, to an auditory display (to a speaker device as speech and non-speech sounds) or to a tactile display (Braille and haptic displays).

Selection, ***current selection***

The selection generally identifies a range of content (e.g., text, images, etc.) in a document. The selection may be structured (based on the document tree) or unstructured (e.g., text-based). Content may be selected through user interaction, scripts, etc. The selection may be used for a variety of purposes: for cut and paste operations, to designate a specific element in a document, to identify what a screen reader should read, etc.

The selection may be set by the user (e.g., by a pointing device or the keyboard) or through an application programming interface (API). A viewport has at most one selection (though the selection may be rendered graphically as discontinuous text fragments). When several viewports co-exist, each may have a selection, but only one is active, called the current selection.

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

Standard device APIs

Operating systems are designed to be used by default with devices such as pointing devices, keyboards, voice input, etc. The operating system (or windowing system) provides "standard APIs " for these devices. On desktop computers today, the standard input APIs are for the mouse and keyboard. For

touch screen devices or mobile devices, standard input APIs may include stylus, buttons, voice, etc. The display and sound card are considered standard ouput devices for a graphical desktop computer environment, and each has a standard API.

Text transcript

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

User agent

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

User interface

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

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

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.

User-initiated, user agent initiated

An action initiated by the user is one that results from user operation of the user interface. An action initiated by the user agent is one that results from the execution of a script (e.g., an event handler bound to an event not triggered through the user interface), from operating system conditions, or from built-in user agent behavior.

Views, viewports, and current viewport

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

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

The view corresponds to *how* source information is rendered and the viewport is *where* it is rendered. The viewport that contains both the current focus and the current selection is called the ***current viewport***. The current viewport is generally highlighted when several viewports co-exist.

A viewport may not give users access to all rendered content at once. In this case, the user agent should provide a scrolling mechanism or advance and rewind mechanism.

4. Acknowledgments

The active participants of the User Agent Guidelines Working Group who produced this document were: James Allan, Denis Anson, Kitch Barnicle, Harvey Bingham, Dick Brown, Al Gilman, Jon Gunderson, Ian Jacobs, Marja-Riitta Koivunen, Charles McCathieNevile, Mark Novak, David Poehlman, Mickey Quenzer, Gregory Rosmaita, Madeleine Rothberg, and Rich Schwerdtfeger.

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

5. References

For the latest version of any W3C specification please consult the list of W3C Technical Reports at http://www.w3.org/TR.

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

[CSS1]
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. This CSS 1 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 CSS 2 Recommendation is http://www.w3.org/TR/1998/REC-CSS2-19980512.

[DOM2]
"Document Object Model (DOM) Level 2 Specification", L. Wood, A. Le Hors, V. Apparao, L. Cable, M. Champion, J. Kesselman, P. Le Hégaret, T. Pixley, J. Robie, P. Sharpe, C. Wilson, eds. The latest version of the specification is available at: http://www.w3.org/TR/DOM-Level-2.

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

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

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

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

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

[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 SMIL 1.0 Recommendation is http://www.w3.org/TR/1998/REC-smil-19980615.

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

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

[WCAG10]
"Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds., 5 May 1999. This WCAG 1.0 Recommendation is http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505.

[WCAG10-TECHS]
"Techniques for Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. The latest version of this document is available at http://www.w3.org/TR/WCAG10-TECHS.

[XHTML10]
"XHTML[tm] 1.0: The Extensible HyperText Markup Language", S. Pemberton, et al. The 26 January 2000 XHTML 1.0 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 XML 1.0 Recommendation is http://www.w3.org/TR/1998/REC-xml-19980210.