Techniques for Authoring Tool Accessibility

W3C Working Draft 2 December 1999

This version:
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-TECHS-19991202
(plain text, HTML gzip tar archive, HTML zip archive, PostScript, PDF)
Latest version:
http://www.w3.org/WAI/AU/WAI-AUTOOLS-TECHS
Previous version:
http://www.w3.org/WAI/AU/WAI-AUTOOLS-TECHS-19991022

Editors:
Jutta Treviranus - ATRC, University of Toronto
Jan Richards - University of Toronto
Ian Jacobs - W3C
Charles McCathieNevile - W3C

Abstract

This document contains example techniques and references to further information, as an informative aid to developers seeking to implement the Authoring Tool Accessibility Guidelines [WAI-AUTOOLS] . The guidelines and checkpoints of that document are included for convenience.

This document is part of a series of accessibility documents published by the W3C Web Accessibility Initiative.

Status of this document

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

This is a Working Draft of the Techniques for Authoring Tool Accessibility. This draft follows the Working Group meeting on 1 December 1999, but the techniques have not been significantly updated since the Working Group meeting on 20 October 1999. For further information consult the minutes of Working Group Meetings.

This is a draft document and may be updated, replaced or rendered obsolete by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by either W3C or its Member organizations.

The goals of the WAI AU Working Group are discussed in the WAI AU charter.

Please send general comments about this document to the public mailing list: w3c-wai-au@w3.org, archived at http://lists.w3.org/Archives/Public/w3c-wai-au

A list of the current AU Working Group participants is available.

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

Table of Contents

1 Introduction

This document complements the Authoring Tool Accessibility Guidelines [WAI-AUTOOLS] . Although it reproduces the guidelines and checkpoints from that document it is not a normative reference; the techniques introduced here are not required for conformance to the Guidelines. The document contains suggested implementation techniques, examples, and references to other sources of information as an aid to developers seeking to implement the Authoring Tool Accessibility Guidelines. These techniques are not necessarily the only way of fulfilling the checkpoint, nor are they necessarily a definitive set of requirements for fulfilling a checkpoint. It is expected to be updated in response to queries raised by implementors of the Guidelines, for example to cover new technologies. Suggestions for additional techniques are welcome and should be sent to the Working Group mailing list at w3c-wai-au@w3.org. The archive of that list at http://lists.w3.org/Archives/Public/w3c-wai-au is also available.

To understand the accessibility issues relevant to authoring tool design, consider that many users may be creating documents in contexts very different from your own:

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

In addition, accessible design will benefit many people who do not have a physical disability but with similar needs. For example they may be working in a noisy environment and unable to hear, or need to use their eyes for another task, and be unable to view a screen. They may be using a small mobile device, with a small screen, no keyboard and no mouse.

1.1 How the Techniques are organized.

This document has the same structure as the Authoring Tool Accessibility Guidelines ([WAI-AUTOOLS]). Each Guideline and checkpoint from that Document is listed, in the same order, with techniques for implementing them, further references, and other information that the working group considers useful for implementing the guidelines but not a normative (required) part of the guidelines themselves. For some guidelines there are techniques or information that are relevant to the entire guideline. These are provided at the end of the section for the relevant guideline.

Some of the techniques describe the implementation of a checkpoint in a real HTML editing tool - W3C's WYSIWYG HTML editor Amaya [AMAYA] . The Amaya techniques are also available as a single "sample implementation" document [AMAYA-SAMPLE] , and it is anticipated that some other sample implementations may be handled in the same way in future drafts.

Each checkpoint is intended to be specific enough that it can be verified, while being sufficiently general to allow developers the freedom to use the most appropriate strategies to meet the checkpoint.

1.2 Checkpoint Priorities

Each checkpoint has a priority level. The priority level reflects the impact of the checkpoint in meeting the goals of this specification. These goals are:

- That the authoring tool be accessible
- That the authoring tool generate accessible content by default
- That the authoring tool encourage the creation of accessible content

The three priority levels are assigned as follows:

[Priority 1]
If the checkpoint is essential to meeting the goals
[Priority 2]
If the checkpoint is important to meeting the goals
[Priority 3]
If the checkpoint is beneficial to meeting the goals
[Relative Priority]

Some checkpoints that refer to generating, authoring, or checking Web content have multiple priorities. The priority is dependent on the priority in the Web Content Accessibility Guidelines [WAI-WEBCONTENT] .

- It is priority 1 to implement the checkpoint for content features that are a priority 1 requirement in Web Content Accessibility Guidelines [WAI-WEBCONTENT] .
- It is priority 2 to implement the checkpoint for content features that are a priority 2 requirement in Web Content Accessibility Guidelines [WAI-WEBCONTENT] .
- It is priority 3 to implement the checkpoint for content features that are a priority 3 requirement in Web Content Accessibility Guidelines [WAI-WEBCONTENT] .

For example:

- providing text equivalents for images and audio is a priority 1 requirement in WCAG [WAI-WEBCONTENT] since without it one or more groups will find it impossible to access the information. Therefore, it is a priority 1 requirement for the authoring tool to check for (4.1) or ask the author for (3.1) equivalent alternatives for these types of content.
- Grouping links in navigation bars is a priority 3 in WCAG [WAI-WEBCONTENT] . Therefore, it is only priority 3 for the authoring tool to check for (4.1) or ask the author for (3.2) groups of links that are not grouped in the markup as a navigation mechanism.

> All authoring tools should support all applicable Web Content Guideline checkpoints, but the nature of that support may vary according to the nature of the authoring tool, the expected skill level of the author using the tool, and the requirements of each WCAG checkpoint. In some cases support can be provided automatically, without the need for explicit author participation, in other cases human judgment is required and support is provided by the tool in the form of prompts and documentation.

In choosing priority levels for checkpoints, the Working Group has assumed that "the author" is a competent, but not necessarily expert, user of the authoring tool, and that the author has no prior knowledge of accessibility. For example, the author is not expected to have read all of the documentation but is expected to know how to turn to the documentation for assistance.

2 Guidelines

Guideline 1. Support accessible authoring practices

If the tool automatically generates markup, many authors will be unaware of the accessibility status of the final content unless they expend extra effort to make appropriate corrections by hand. Since many authors are unfamiliar with accessibility, the onus is on the authoring tool to generate accessible markup, and where appropriate, to guide the author in producing accessible content.

Many applications feature the ability to convert documents from other formats (e.g., Rich Text Format) into a markup format specifically intended for the Web such as HTML. Markup changes may also be made to facilitate efficient editing and manipulation. It is essential that these processes do not introduce inaccessible markup , or remove accessibility content, particularly since the markup changes are hidden from the author's view in many tools.

Checkpoints:

1.1 Ensure that the author can produce accessible content in the markup language(s) supported by the tool. [Priority 1]

- Provide options for accessibility information such as equivalent alternatives to be included whenever an object is added to a document. Refer also to checkpoint 3.1.
- Allow direct source editing (but note also the requirements in 1.2 and 5.1)
- Most image formats, including PNG, SVG, WebCGM, JPEG and GIF allow the inclusion of text content. Where it is available in the format this feature should be used by authoring tools.
- Many formats for audio or video allow for the use of equivalent alternatives.
- General: Web Content Accessibility Guidelines 1.0 [WAI-WEBCONTENT]
- Techniques for Web Content Accessibility Guidelines 1.0 [WAI-WEBCONTENT-TECHS]
- The Web Accessibility Initiative publishes a series of notes on the accessibility features of different W3C specifications. W3C Notes are currently available for:
  1. CSS2 [CSS2-ACCESS]
  2. SMIL [SMIL-ACCESS]

  For HTML 4.0, refer to [HTML4-ACCESS] .

  Working drafts are available for

  1. SVG [SVG-ACCESS]

  In each case the specifications themselves also provide information.
- Amaya implements all of the accessibility features of HTML. The CSS cascade order, an accessibility feature of CSS2, is yet to be completely implemented in Amaya.

- An audio/video editing suite can use SMIL to separate audio, video, descriptive signing and text tracks.An audio/video editing suite can use SMIL to separate audio, video, descriptive signing and text tracks.

1.2 Ensure that the tool preserves all accessibility information during authoring, transformations and conversions . [Priority 1]

- When transforming a table to a list or list of lists, ensure that table headings are transformed into headings, and summary or caption information is retained as rendered content. (This transformation is not necessarily cleanly reversible)
- Ensure that a sensible reading order can be maintained when document layout is rearranged, for example by prompting the author to confirm linearized reading order after positioning text. (Desktop publishing software often has such a feature).
- When importing images with associated descriptions to an HTML document make the descriptions available for use as `longdesc`, `alt`, or `title`
- When converting from a word-processor format to HTML ensure that headings and list items are transformed into appropriate headings of the appropriate level, and list items in the appropriate type of list (rather than as plain text with font formatting)
- Do not transform text into images - use style sheets for presentation control, or an XML application such as Scalable Vector Graphics that keeps the text as text. If this is not possible, ensure that the text that is converted is available as equivalent text for the image.
- Ensure that the tool recognizes and preserves elements that are defined in the relevant specification(s) even if it is unable to render them in a publishing view or preview mode. This is relevant for WYSIWYG page authoring tools, tools that handle image formats which allow the incorporation of titles, descriptions, etc.
- When converting linked elements such as footnotes or endnotes either provide them as inline content or or maintain two-way linking. In HTML this should be hypertext links rather than plain-text references.
- The predefined transformations shipped with Amaya preserve all element content. The transformation language allows the preservation of attribute values, but this is not done by all the supplied transformations.

1.3 Ensure that the tool generates markup that conforms to the W3C's Web Content Accessibility Guidelines [WAI-WEBCONTENT] . [Relative Priority]

- Use consistent document structures. For a tool that does site-wide management provide consistent navigation systems and document structures.
- Include markup that provides equivalent alternatives for media-dependent elements or content.
- Do not use structural markup for presentational effects, or presentation markup for known structures. For example, use list markup of an appropriate type rather than creating multiple line paragraphs and beginning each line with an image of a bullet, and do not use list markup for an indentation effect.

- Do not publish Web content in markup languages that do not allow for equivalent alternative information to be included for media-specific presentations (such as images or video, sound, etc).
- The Web Accessibility Initiative's Protocols and Formats group have a draft set of notes about creating accessible markup languages [XMLGL] .
- New markup languages are constantly being developed, and in many cases offer improvements to the structure and utility of Web content. In implementing a new or extended markup language it is important to ensure that a tool does not remove access to information that had been inherent in the base markup language.

  The same can apply to a format which is made by simplifying an existing format. For example, producing a modified HTML DTD that did not include the "`alt`" attribute for `IMG`, or effectively working to such a DTD by not implementing a means to include the attribute, compromises the accessibility of any included `IMG` elements.

- Amaya generates markup that conforms to level-A, and allows the author to generate markup that is triple-A through the user interface.

1.4 Ensure that templates provided by the tool conform to the Web Content Accessibility Guidelines [WAI-WEBCONTENT] . [Relative Priority]

- Produce accessible representations for site maps generated by the authoring tool.
- Provide equivalent alternatives for all non-text content (images, audio, etc).
- Use consistent navigation mechanisms.
- Ensure that event-handlers for scripts are device-independent.
- Ensure that color schemes provide sufficient contrast for people or technology with poor color separation.
- Ensure that the natural language of the template is identified.
- Provide navigation bars.
- Provide keyboard shortcuts for important links, etc.
- Amaya has just introduced templates, which will be checked for conformance to [WAI-WEBCONTENT] .

Guideline 2. Generate standard markup

Conformance with standards promotes interoperability and accessibility, by making it easier to create specialized user agents that address the needs of users with disabilities. In particular many assistive technologies used with browsers and multimedia players are only able to provide access to Web documents that use valid markup. Therefore, valid markup is an essential aspect of authoring tool accessibility.

Where applicable use W3C Recommendations, which have been reviewed to ensure accessibility and interoperability. If there are no applicable W3C Recommendations, use a published standard that enables accessibility.

Checkpoints:

2.1 Use the latest versions of W3C Recommendations when they are available and appropriate for a task. [Priority 2]

W3C specifications have undergone review specifically to ensure that they do not compromise accessibility, and where possible they enhance it.

- When creating documents or markup languages, make full use of W3C Recommendations (Specifications that have been approved by the W3C. These specifications have undergone review specifically to ensure that they do not compromise, and where possible they enhance, accessibility). For example when creating mathematical content for the Web use MathML [MATHML] rather than another markup language. Use applicable HTML [HTML40] structures.
- Ensure that the tool recognizes and preserves elements that are defined in the relevant specification(s) even if it is unable to render them. This is particularly important for WYSIWYG editing tools.
- Amaya supports HTML 4.0 [HTML40] , XHTML 1.0 [XHTML10] and most of CSS1 [CSS1] . It provides partial support for MathML [MATHML] and some experimental support for Scalable Vector Graphics [SVG] .
- Sketch has a very basic experimental implementation of Scalable Vector Graphics [SVG] .

2.2 Ensure that the tool generates valid markup. [Priority 1]

This is necessary for user agents to be able to render Web content in a manner appropriate to a particular user's needs.

- Produce valid HTML/XML
- Publish proprietary language specifications or DTDs on the Web, to allow documents to be validated.
- Use namespaces and schemas to make documents that can be automatically transformed to a known markup language.
- Amaya implements each language according to the published specifications.

2.3 If markup generated by the tool does not conform to W3C specifications, inform the author. [Priority 3]

- If Amaya imports or generates markup that does not conform to W3C specifications it is highlighted in the structure view. (This occurs when Amaya tries to repair invalid markup and cannot successfully do so).
- Word does not do this.
- ASWedit automatically drops out of HTML mode if markup is added or imported that does not conform to the DTD (and to get back into HTML mode requires correcting the errors. Refer also to checkpoint 4.1..

Guideline 3. Support the creation of accessible content

Well structured information, and equivalent alternative information are cornerstones of accessible design, allowing information to be presented in a way most appropriate for the needs of the user without constraining the creativity of the author. Yet generating equivalent information, such as textual alternatives for images and auditory descriptions of video, can be one of the most challenging aspects of Web design, and authoring tool developers should attempt to facilitate and automate the mechanics of this process. For example, prompting authors to include equivalent alternative information such as text equivalents , captions , and auditory descriptions at appropriate times can greatly ease the burden for authors. Where such information can be mechanically determined and offered as a choice for the author (e.g., the function of icons in an automatically-generated navigation bar, or expansion of acronyms from a dictionary) the tool can assist the author. At the same time it can reinforce the need for such information and the author's role in ensuring that it is used appropriately in each instance.

Checkpoints:

3.1 Prompt the author to provide equivalent alternative information (e.g., captions , auditory descriptions and collated text transcripts for video). [Relative Priority]

- When a multimedia object is inserted, prompt the author for relevant alternatives: functional replacement and long description for images, text captions (as text or as a URI), video of signed translations for audio, audio descriptions for video (as well as alternatives for its audio components).
- Provide an author with the option of specifying alternative information, or electing to insert null alternative information for images, audio, video. Default to an accessibility error such as no `TITLE` or `DESC` element for SVG images. Prompt the author to identify the type of image (decorative, a navigation icon, etc.).
- When video is inserted, prompt the author for a still image as part of the alternative information.
- When inserting objects such as spreadsheets, or word processor documents, offer the option of providing a Web-formated version. For example a spreadsheet or a word processor document in a proprietary format could also be published as an HTML document. Tools which dynamically generate Web content may use HTTP content negotiation to facilitate this.
- Amaya prompts the author to provide alt text for `IMG` and `AREA`, and `CAPTION` for `TABLE`.
- Meeting checkpoint 3.5 would provide much of the required functionality Refer also to checkpoint 4.1..
- (refer also to checkpoint 3.5, checkpoint 6.2)

3.2 Help the author create structured content and separate information from its presentation. [Relative Priority]

- Recognize collections of upper-case letters as likely abbreviations (in

languages that have case) and prompt the author for an expansion, to be provided in markup.

- Prompt the author to identify the structural role of content which has been emphasized through styling.
- In Japanese, Chinese, and other appropriate languages, prompt the author for text that can be used as a ruby for unusual ideographs or ideographic groups.
- Prompt the author for header information for tabular data.
- Prompt the author (and allow them to specify a default suggestion) for the language of a document.
- In future releases Amaya will prompt the author for title for `ABBR`, `ACRONYM`, `OBJECT`, `IMG` and `LABEL` for `FORM` controls. The user interface of Amaya was developed to guide authors to produce structured documents. Style in Amaya is created as a stylesheet.

3.3 Ensure that prepackaged content conforms to Web Content Accessibility Guidelines [WAI-WEBCONTENT] . [Relative Priority]

For example include synchronized text and audio equivalents (such as captions , auditory description and a collated text transcript) with movies. Refer also to checkpoint 3.4.

- Use formats that allow for accessible annotation to be included in the files, such as SMIL, PNG and SVG.
- Provide long descriptions, and associated text files with appropriate "alt-text" in clip-art collections.
- Provide video description files with prepackaged video.
- Provide text caption files for prepackaged audio, or video with audio track(s).
- Including pre-written descriptions for all multimedia files (e.g., clip-art) packaged with the tool would save users time and effort, cause a significant number of professionally written descriptions to circulate on the Web, provide users with convenient models to emulate when they write their own descriptions and show authors the importance of description writing.
- Amaya does not provide any clip art
- Refer also to checkpoint 3.5.

3.4 Do not insert automatically generated or place-holder equivalent alternatives . [Priority 1]

For example, "search" may be appropriate alternative text for a graphic button linked to a search function, but the filename of an image should not be inserted as a default.

Note. Human-authored equivalent alternatives may be available for an object (for example through checkpoint 3.5 and/or checkpoint 3.3). It is appropriate for the tool to offer these to the author as defaults.

- Items used throughout a Website, such as graphical navigation bars, should have standard alternative information. However the author should be prompted to edit or approve this the first time it is used in a site, and when the destination of the links is changed by the author.

- If the author has not specified alternative text for an IMG, or specified that none is required, default to having no alt attribute, so that an accessibility problem will be noted. Refer also to checkpoint 4.1.
- Where an object has already been used in a document, the tool should offer the alternative content that was supplied for the first or most recent use as a default.
- Amaya does not provide default alt text except when copying and pasting images, in which case it copies all attributes with the image.

3.5 Provide a mechanism to manage alternative information for multimedia objects, that retains and offers for editing pre-written or previously linked equivalent alternative information. [Priority 3]

- Maintain a database registry that associates object identity information with alternative information. Whenever an object is used and an equivalent alternative is provided, ask the author whether they want to add the object (or identifying information) and the alternative information to the database. In the case of alt-text the alternate information might be stored directly while more substantial information (such as video captions or audio descriptions) would be entered as pointers to external files. Allow different alternative information to be associated with a single object.
- Allow authors to make keyword searches of a description database (to simplify the task of finding relevant images, sound files, etc.). A paper describing a method to create searchable databases for video and audio files is available (refer to [SEARCHABLE]).
- Suggest pre-written descriptions as default text whenever one of the associated files is inserted into the author's document.
- The use of RDF, or formats like SVG can enable a tool to maintain and use libraries of information within the tool and on the Web.
- This checkpoint is prioritized as a level 3, meaning that in itself, it does not have a critical effect on an authoring tool's likelihood of producing accessible mark-up. However, several limited extensions to this alternative information management mechanism (AIMM) have the potential to simultaneously meet several higher priority checkpoints and dramatically improve the usability of an access aware authoring tool. In particular:
  1. The AIMM should maintain a list of associations between object file names and authored responses to prompts for alternative information (as per checkpoint 3.1). The alternative information may take the form of short strings (i.e. "alt"-text) or pointers to descriptive files (i.e., "longdesc", transcripts, etc.). Multiple associations for the same object for different languages or contexts should also be handled.
  2. The AIMM would offer the associated alternative information as a default whenever the appropriate associated object is selected for insertion. If no previous association is found, the field should be left empty (i.e., no purely rule-generated alternative information should be used). **Note.** The term "default" implies that the alternative information is offered for the author's approval. The term does not imply that the default alternative information is automatically placed without the

author's approval. Such automatic placement may only occur when in situations where the function of the object is known with certainty, as per checkpoint 3.4. Such a situation might arise in the case of a "navigation bar builder" that places a navigation bar at the bottom of every page on a site. In this case, it would be appropriate to use the same "alt"-text automatically for every instance of a particular image (with the same target) on every page.

3. The alternative information mechanism should be closely integrated with the pre-written alternative information provided for all packaged multimedia files, as per checkpoint 3.3. This would allow the alternative information to be automatically retrieved whenever the author selected one of the packaged objects for insertion. An important benefit of the system would be the ease of adding a keyword search capability that would allow efficient location of multimedia based on its alternative information.

- Amaya has no registry of alternate text associated with images, although when an image is copied and pasted its alt and other attributes are copied too.

Guideline 4. Provide ways of checking and correcting inaccessible content

Many authoring tools allow authors to create documents with little or no knowledge about the underlying markup. To ensure accessibility, authoring tools must be designed so that they can (where possible automatically) identify inaccessible markup , and enable its correction even when the markup itself is hidden from the author.

In supporting the creation of accessible Web content, authoring tools should take into account differing authoring styles. In general, authors will prefer to be able to configure their tools to support their working style. Tools that allow such configuration can help authors to feel that accessible authoring is a natural practice (refer to guideline 5) rather than an intrusion on their normal work pattern. For example some users may prefer to be alerted to accessibility problems when they occur, whereas others may prefer to perform a check at the end of an editing session. This is analogous to programming environments that allow users to decide whether to check for correct code during editing or at compile time.

Note. Validation of markup is an essential aspect of checking the accessibility of content.

Checkpoints:

4.1 Check for and alert the author to accessibility problems . [Relative Priority]
Note: Accessibility problems should be detected automatically where possible. Where this is not possible, the tool may need to prompt the user to make decisions, or to manually check for certain types of problem.

- The WAI Evaluation and Repair group [WAI-ER] is developing a document that discusses which aspects of the Web Content Accessibility Guidelines can be automatically tested. A draft of that document is available [AUTO-TOOL] .
- Where the tools cannot test for accessibility errors provide the author with the necessary information, wizards, etc to check for themselves. Refer also to checkpoint 5.1.
- Include alerts for [WAI-WEBCONTENT] Priority 1 checkpoints in the default configuration.
- Providing an editing view that shows equivalent alternatives in the main content view will make it clear that they are necessary, and will make it obvious when they are missing.
- Provide a preview mode that uses alternative content. Although this can give authors a clear understanding of some problems very easily, it should be made clear that there are many ways in which a page may be presented (aurally, text-only, text with pictures separately, on a small screen, on a large screen, etc.). A view that renders the document as it might appear without technologies such as style sheets and images enabled, or the ability to turn those features off and on in the editing view, will also give an author some idea of whether a document's logical order has been correctly preserved, whether alternative text is appropriate, etc.
- Highlight problems detected when documents are opened, when an editing or insertion action is completed, or while an author is editing. Using CSS classes to indicate accessibility problems will enable the author to easily configure the presentation of errors.
- Where there is a change in the writing script used, prompt the author to identify whether there has been a change in language
- Alert authors to accessibility problems when saving.
- Accessibility problems can be highlighted using strategies similar to spell checking within a word processor. Accessibility alerts within the document can be linked to context sensitive help Refer also to checkpoint 4.2..
- Allow users to choose different alert levels based on the priority of authoring accessibility recommendations.
- If interruptive warnings are used, provide a means for the author to quickly set the warning to non-obtrusive to avoid frustration.
- There are online tools whose output can be integrated with the user interface. Other tools are available for incorporation in existing software, either as licensed products or in some cases as "open source" solutions. The WAI Evaluation and Repair group maintains information about available tools [WAI-ER] .
- Amaya currently checks for validity, but the author can only find warning of invalid markup in the structure view. The team is investigating automating an accessibility check and author notification. Where Amaya detects an error it identifies and highlights the offending code in the structure view, allowing the author to delete it.

- Word has spell and grammar check, but no validity or accessibility

4.2 Assist authors in correcting accessibility problems . [Relative Priority]

At a minimum, provide context-sensitive help with the accessibility checking required by 4.1

- Do this in a way that is consistent with the look and feel of the authoring tool.
- Provide context sensitive-help for accessibility errors. Refer also to 6.
- Where there are site-wide errors, to make correction more efficient the author can be given the choice to make site-wide changes or corrections. For example this may be appropriate for a common error in markup, but may not be appropriate in providing alt text that is appropriate for one use of an image but completely inappropriate for the other uses of the image on the same site (or even the same page).
- Allow authors to control both the nature and timing of the correction process.
- Provide a mechanism for authors to navigate sequentially among uncorrected accessibility errors Refer also to checkpoint 7.4..
- Amaya currently does not implement this checkpoint. Amaya uses its own internal representation for the document markup that is translated on output. Possible implementation strategy: Where there are errors in a document Amaya could alert the author and warn that the document must be changed, and present the structure view highlighting areas where it has changed the markup, allowing the author to abort the editing session or save the changed version under a new name.

4.3 Allow the author to preserve markup not recognized by the tool. [Priority 2]

Note. The author may have included or imported markup that enhances accessibility but is not recognized by the tool.

- Provide a summary of all automated structural changes that may affect accessibility.
- Provide options for the author to confirm or override removal of markup on a change-by-change basis or as a batch process.
- Do not change the DTD without notifying the author.
- Amaya currently does not implement this checkpoint.
- Word sometimes warns that information will be lost (allowing the author to cancel the transformation), and sometimes it doesn't.

4.4 Provide the author with a summary of the document's accessibility status. [Priority 3]

- Provide a summary of accessibility problems remaining by type and/or by number.
- HoTMetaL 5 has an accessibility checker that says how many errors are found in a document.
- Bobby provides a list of errors found in a Web page.
- Amaya currently does not implement this checkpoint.

4.5 Allow the author to transform presentation markup that is misused to convey structure into structural markup, and to transform presentation markup that is stylistic into style sheets. [Priority 3]

- Some examples of transformations include: HTML table-based layout into CSS, HTML br to p, HTML (deprecated) FONT into heuristically or author-determined structure, Word processor styles to Web styles, HTML deprecated presentational markup into CSS, XHTML `span` into `ruby`, MathML presentational markup to semantic markup.
- Allow the user to define transformations for imported documents that have presentation, rather than structural, markup.
- Allow the user to create style rules based on the formatting properties of an element, and then apply the rule to other elements in the document, to assist conversion of documents to the use of style sheets
- Include pre-written transformations to rationalize multiple tables, and to transform (deprecated) presentation HTML into style sheets.
- Remember that accessibility information, including attributes or properties of the elements being transformed, must be preserved - see checkpoint 1.2
- Amaya provides a language for specifying structure transformations, along with a large number of transformations being included.

Techniques for this guideline:

- Prompts can be used to encourage authors to provide information needed to make the content accessible (such as alternative text equivalents). Prompts are simple requests for information. For example, an "alt-text" entry field prominently displayed in an image insertion dialog would constitute a prompt. Prompts are relatively unintrusive and address a problem before it has been committed. However, once the user has ignored the prompt, its message is unavailable.

  Alerts warn the author that there are problems that need to be addressed. The art of attracting users' attention is a tricky issue. The way users are alerted, prompted, or warned can influence their view of the tool and even their opinion of accessible authoring. Refer also to 5.

  User Configurable Schedule

  A user configurable schedule allows the user to determine the type of prompts and alerts that are used, including when they are presented. For example, a user may wish to include multiple images without being prompted for alternative information, and then provide the alternative information in a batch process, or may wish to be reminded each time they add an image. If the prompting is done on a user configurable schedule they will be able to make that decision themselves. This technique allows a tool to suit the needs a wide range of authors.

  Interruptive Alerts

  Interruptive alerts are informative messages that interrupt the edit process for the user. For example, interruptive alerts are often presented when a user's action could cause a loss of data. Interruptive alerts allow problems to be brought to the user's attention immediately. However, users may

resent the constant delays and forced actions. Many people prefer to finish expressing an idea before returning to edit its format.

Unintrusive Alerts

Unintrusive alerts are alerts such as icons, underlines, and gentle sounds that can be presented to the user without necessitating immediate action. for example, in some word processors misspelled text is highlighted without forcing the user to make immediate corrections. These alerts allow users to continue editing with the knowledge that problems will be easy to identify at a later time. However, users may become annoyed at the extra formatting or may choose to ignore the alerts altogether.

Guideline 5. Integrate accessibility solutions into the overall "look and feel"

When a new feature is added to an existing software tool without proper integration, the result is often an obvious discontinuity. Differing color schemes, fonts, interaction styles and even application stability can be factors affecting user acceptance of the new feature. In addition, the relative prominence of different ways to achieve the same thing can be an important factor in which method an author chooses. Therefore, it is important that creating accessible content is a natural process when using an authoring tool.

Checkpoints:

5.1 Ensure that functions related to accessible authoring practices are naturally integrated into the tool. [Priority 2]

- Ensure that author can utilize the tool's accessible authoring features by the same interaction styles used for other features in the program. For example, if the tool makes use of onscreen symbols such as underlines or coloration change rather than dialogs for conveying information, then the same interface techniques should be used to convey accessibility information.
- The same fonts, text sizes, colors, symbols, etc. that characterize other program features should also characterize those dealing with accessibility.
- Include considerations for accessibility - such as the "alt" and "longdesc" attributes of the HTML `IMG` element - right below the "src" attribute in a dialogue box, not buried behind an "Advanced..." button.
- Allow efficient and fast access to accessibility-related settings with as few steps as possible needed to make any changes that will generate accessible content.
- The accessibility features should be designed as integral components of the authoring tool application, not plug-ins or other peripheral components that need to be separately obtained, installed, configured or executed
- The default installation of the authoring tool should include all accessibility features enabled. The author may have the option to disable these features later on.

- A help page that describes how to make an HTML image map should include adding alternative information for each AREA in the MAP as part of the process. Any examples of code should give either block content with text links, or AREA elements that all have relevant "alt" attribute values.
- When a user creates an HTML frameset, suggest the links from the navigation bar (and perhaps the content of the "first page") as the content for the NOFRAMES element.
- In Amaya some accessibility features are part of relevant dialogs. Others, such as longdesc and title attributes must be separately generated by the author. The development team will integrate these into the relevant dialogues.

5.2 Ensure that Web Content Accessibility Guidelines [WAI-WEBCONTENT] Priority 1 accessible authoring practices are among the most obvious and easily initiated by the author. [Priority 2]

- When the user has selected text to format, the use of CSS should be emphasized rather than the HTML FONT element.
- Highlight the most accessible solutions when presenting choices for the author.
- Providing an editing view that shows equivalent alternatives in the main content view will make it clear that they are necessary, and will make it obvious when they are missing.
- If there is more than one option for the author, and one option is more accessible than another, place the more accessible option first and make it the default. For example, when requesting equivalent alternatives for an image with the HTML OBJECT element, offer an unchecked option for a null value (i.e., there is no content, implying the image has no real function) with the cursor positioned in the entry field for alternative text (and if available provide the appropriate value from the "Alternative Information Management Mechanism" Refer also to checkpoint 3.5., rather than offering the filename as a default suggestion, or selecting the null "alt" value as a default.
- Amaya's user interface guides the author to produce structured content, with presentation elements separated into style sheets. Providing an equivalent alternative is mandatory at the time of inserting some elements.
- Word does this with some features such as style (the interface for dealing with style is extremely well-integrated, although the alternative approach is at least as obvious), and to a lesser extent outlining. Adding equivalent alternatives for images requires knowing how it is done - it is not obvious.

Guideline 6. Promote accessibility in help and documentation

The issues surrounding the creation of accessible Web content are often unknown to Web authors. Help and documentation must include explanations of accessibility problems , and should demonstrate solutions with examples.

Checkpoints:

6.1 Document all features that promote the production of accessible content. [Priority 1]

- Ensure that accessibility solutions are present in all help text descriptions of markup practices (e.g., IMG elements should appear with "alt-text" and a "longdesc" attribute wherever appropriate).
- Ensure that electronic documentation complies with the Web Content Accessibility Guidelines [WAI-WEBCONTENT]
- Link from help text to any automated correction utilities.
- Provide examples of accessible design practices in online tutorials.
- Include help documentation for all accessible authoring practices supported by the tool.
- Link those mechanisms used to identify accessibility problems (e.g., icons, outlining or other emphasis within the user interface) to help files.
- Amaya help pages for images and image maps [AMAYA-HELP-IMG] include providing text alternatives as part of the process. There is a help page on configuring Amaya, that documents how to change the default keyboard bindings. Some pages need to be updated.

6.2 Ensure that creating accessible content is a naturally integrated part of the documentation, including examples. [Priority 2]

- In help text, when explaining the accessibility issues related to non-deprecated elements, emphasize appropriate solutions rather than explicitly discouraging the use of the element.
- Explain the importance of utilizing accessibility features generally and for specific instances.
- Take a broad view of accessibility-related practices; for example, do not refer to "alt-text" as being "for blind users" but rather as "for users who are not viewing images".
- Avoid labelling accessibility features of the tool with a "handicapped" icon, as this can give the impression that accessible design practices only benefit disabled users.
- In help text, emphasize accessibility features that benefit multiple groups. In particular the principles of supporting flexible display and control choices have obvious advantages for the emergence of hands free, eyes-free, voice-activated browsing devices such as Web phone, the large number of slow Web connections, and Web users who prefer text-only browsing to avoid "image clutter".
- Provide examples of all accessibility solutions in help text, including those of lower priority in [WAI-WEBCONTENT] .
- Implement context-sensitive help for all special accessibility terms as well as tasks related to accessibility.
- Document the tool’s conformance to [WAI-AUTOOLS] .
- Include current versions of, or links to relevant specifications in the documentation (e.g. HTML 4.0 [HTML40] , CSS [CSS2] .) This is

particularly relevant for markup languages that are easily hand edited, such as most [XML] languages.

- Include a tutorial specifically on checking for and correcting Web accessibility problems.
- Link to or provide URIs for more information on accessible Web authoring, such as [WAI-WEBCONTENT] , and other accessibility-related resources.
- Accessible authoring features are added to the documentation as they are incorporated into Amaya, as part of the normal documentation of the relevant feature.
- Ensure that documentation examples conform to [WAI-WEBCONTENT] .
- Clearly label any examples that display practices that reduce accessibility.
- Word documentation includes integrated accessibility requirements in some areas, but some areas need further work.

6.3 In a dedicated section, document all features of the tool that promote the production of accessible content. [Priority 3]

- Amaya does not currently implement this checkpoint. An accessibility section will be provided in the next release version.

Guideline 7. Ensure that the authoring tool is accessible to authors with disabilities

The authoring tool is a software program with standard user interface elements and as such must be designed according to relevant user interface accessibility guidelines. When custom interface components are created it is essential that they are accessible through standard access mechanisms.

Some additional user interface design considerations apply specifically to Web authoring tools . For instance, authoring tools must ensure that the author can edit (in the editing view) using one set of stylistic preferences and publish using different styles. For instance, authors with low vision may need large text when editing but want to publish with a smaller default text size. The style preferences of the editing view must not affect the markup of the published document.

Authoring tools must also ensure that the author can navigate a document efficiently while editing, regardless of disability. Authors who use screen readers, refreshable braille displays, or screen magnifiers can make limited use (if at all) of graphical artifacts that communicate the structure of the document and act as signposts when traversing it. For authors with blindness or motor impairments, fatigue and other problems that arise when serial access is the only navigation technique are major usability issues. Authoring tools should therefore provide an editing view that conveys a sense of the overall structure and allows structured navigation.

Note. Documentation, help files, and installation are part of the software and need to be available in an accessible form.

Checkpoints:

7.1 Use all applicable operating system and accessibility standards and conventions (Priority 1 for standards and conventions that are essential to accessibility, Priority 2 for those that are important to accessibility, Priority 3 for those that are beneficial to accessibility).

The techniques for this checkpoint include references to checklists and guidelines for a number of platforms and to general guidelines for accessible applications.

- Not all of the guidelines and checklists for application accessibility are prioritized according to their impact on Accessibility. For instance the priorities in [MS-SOFTWARE] are partially determined by a logo requirement program. Therefore developers may need to compare the documents they are using to other guidelines. [WAI-WEBCONTENT] and [WAI-USERAGENT] both have priority systems that are directly compatible with the priorities in [WAI-AUTOOLS] .
- Guidelines for specific platforms include
  1. "IBM Guidelines for Writing Accessible Applications Using 100% Pure Java" [JAVA-ACCESS] R. Schwerdtfeger, IBM Special Needs Systems.
  2. "An ICE Rendezvous Mechanism for X Window System Clients" [ICE-RAP] , W. Walker. A description of how to use the ICE and RAP protocols for X Window clients.
  3. "Information for Developers About Microsoft Active Accessibility" [MSAA] Microsoft Corporation.
  4. "The Inter-Client communication conventions manual" [ICCCM] . A protocol for communication between clients in the X Window system.
  5. "Lotus Notes accessibility guidelines" [NOTES-ACCESS] IBM Special Needs Systems.
  6. "Java accessibility guidelines and checklist" [JAVA-CHECKLIST] IBM Special Needs Systems.
  7. "The Java Tutorial. Trail: Creating a GUI with JFC/Swing" [JAVA-TUT] . An online tutorial that describes how to use the Swing Java Foundation Class to build an accessible User Interface.
  8. "Macintosh Human Interface Guidelines" [APPLE-HI] Apple Computer Inc.
  9. "The Microsoft Windows Guidelines for Accessible Software Design" [MS-SOFTWARE] .
- Guidelines for specific software types include
  1. "The Three-tions of Accessibility-Aware HTML Authoring Tools" [ACCESS-AWARE] , J. Richards.
  2. "User Agent Accessibility Guidelines (Working Draft)" J. Gunderson, I. Jacobs eds. (This is a work in progress) [WAI-USERAGENT]
- General guidelines for producing accessible software include:
  1. "Accessibility for applications designers" [MS-ENABLE] Microsoft

Corporation.

2. "Application Software Design Guidelines" [TRACE-REF] compiled by G. Vanderheiden. A thorough reference work.
3. "Designing for Accessibility" [SUN-DESIGN] Eric Bergman and Earl Johnson. This paper discusses specific disabilities including those related to hearing, vision, and cognitive function.
4. "EITAAC Desktop Software standards" [EITAAC] Electronic Information Technology Access Advisory (EITACC) Committee.
5. "Requirements for Accessible Software Design" [ED-DEPT] US Department of Education, version 1.1 March 6, 1997.
6. "Software Accessibility" [IBM-ACCESS] IBM Special Needs Systems
7. "Towards Accessible Human-Computer Interaction" [SUN-HCI] Eric Bergman, Earl Johnson, Sun Microsytems 1995. A substantial paper, with a valuable print bibliography.
8. "What is Accessible Software" [WHAT-IS] James W. Thatcher, Ph.D., IBM, 1997. This paper gives a short example-based introduction to the difference between software that is accessible, and software that can be used by some assistive technologies.

- User Interfaces are sometimes built as Web content, and as such should follow the Web Content Accessibility Guidelines [WAI-WEBCONTENT] . Refer also to 1.
- The following are common requirements for producing accessible software. This list does not necessarily cover all requirements for all platforms, and items may not be applicable to some software.

Following Standards

- Draw text and objects using system conventions
- Make mouse, keyboard, and API activation of events consistent
- Provide a User Interface that is "familiar" (to system standards, or across platform)
- Use system standard indirections and APIs wherever possible
- Ensure all dialogs, subwindows, etc meet these requirements
- Avoid blocking assistive technology functions (sticky/mouse keys, screenreader controls, etc) where possible

Configurability

- Allow users to create profiles
- Allow control of timing, colors, sizes, input/output devices and media
- Allow users to reshape the user interface - customize toolbars, keyboard commands, etc

Input Device Independence

- Provide Keyboard access to all functions
- Document all keyboard bindings
- Provide customizable keyboard shortcuts for common functions
- Provide logical navigation order for the keyboard interface.
- Avoid repetitive keying wherever possible
- Provide mouse access to functions where possible

Icons, Graphics, Sounds

- Provide graphical (text) equivalents for sound warnings
- Allow sounds to be turned off
- Provide text equivalents for images/icons
- Use customizable (or removable) colors/patterns
- Ensure high contrast is available (as default setting)
- Provide text equivalents for all audio
- Use icons that are resizeable or available in multiple sizes

Layout

- Do not rely on color alone for meaning. Use color for differentiation, in combination with accessible cues (text equivalents, natural language, etc)
- Position related text labels and objects consistently, and in an obvious manner (labels before objects is recommended)
- Group related controls
- Ensure default window sizes fit in screen
- Allow for window resizing (very small to very large)

User Focus

- Clearly identify the user focus (and expose it via API)
- Unexpected events should not be caused by viewing content (for example by moving the focus to a new point)
- Allow user control of timing - delays, time-dependent response, etc
- Allow for navigation between as well as within windows

2.7.1 Documentation

- Provide documentation for all features of the tool
- Ensure that help functions are accessible

- Amaya is currently available for two platforms: Unix and Windows. There is some work required on both platforms to bring it into line with conventions, in particular to provide conformance with the User Agent Guidelines [WAI-USERAGENT] , and to implement Microsoft Active Accessibility [MSAA] . It is being re-written to take advantage of the improved

accessibility support possible in Gnome (it currently uses Motif) in the Unix version. The Documentation is all available online as HTML and has been reviewed to ensure it conforms to [WAI-WEBCONTENT] .

7.2 Allow the author to change the presentation within editing views without affecting the document markup. [Priority 1]

This allows the author to edit the document according to personal requirements, without changing the way the document is rendered when published.

- In representing the source structure of a document mark elements with textual brackets rather than purely graphic representations. For example "</>" is regarded as a textual bracket, since it is made of character elements.
- Allow the user to create audio style sheets using a graphical representation rather than an audio one (with accessible representation, of course).
- An authoring tool that offers a "rendered view" of a document, such as a browser preview mode, may provide an editing view whose presentation can be controlled independently of the rendered view.
- A WYSIWYG editor may allow an author to specify a local style sheet, that will override the "published" style of the document in the editing view.
- Amaya allows the user to create local style sheets, and to enable or disable each style sheet that is linked to a document.

7.3 Allow the author to edit all properties of each element and object in an accessible fashion. [Priority 1]

- An authoring tool may offer several editing views of the same document, such as a source mode that allows direct editing of all properties.
- Allow the author to individually edit each attribute of the elements in an HTML or XML document, for example through a menu. **Note.** This must include the ability to add values for attributes that are not present, as well as changing current values of attributes.
- Amaya allows each attribute to be edited through the menu or through the structure view. Element types can be assigned through the menu system.
- For a site management tool, allow the author to render a site map in text form (e.g., as a structured tree file).
- Allow the author to specify that alternative information (or identifiers such as a URI or filename) are rendered in place of images or other multimedia content while editing.
- Include attributes / properties of elements in a view of the structure.
- Provide access to a list of properties via a "context menu" for each element.
- Graphically represented elements cannot be identified by assistive technologies that translate text to braille, speech, or large print, unless there is appropriate information available as text. For example, some HTML authoring tools render start and end tags as graphics.

7.4 Ensure the editing view allows navigation via the structure of the document in an accessible fashion. [Priority 1]

- Some tools do not have an editing view.
- Allow the author to navigate via an "outline" or "structure" of the document

being edited. This is particularly important for people who are using a slow interface such as a small braille device, or speech output, or a single switch input device. It is equivalent to the ability provided by a mouse interface to move rapidly around the document.

- To minimally satisfy this checkpoint, allow navigation from element to element.
- In a hypertext document allow the author to navigate among links and active elements of a document.
- For SMIL and other time-based presentations allow the author to navigate through the presentation in time.
- Allow the author to navigate regions of an image, or the document tree for an image expressed in a structured language such as Scalable Vector Graphics [SVG]
- Amaya provides a structure view, that can be navigated element by element, a Table of Contents view, that allows navigation via the headings, and a links view, that allows sequential navigation via the links in the document. It also provides configurable keyboard navigation of the HTML structure - parent, child, next and previous sibling elements.

7.5 Enable editing of the structure of the document in an accessible fashion. [Priority 2]

- An authoring tool may offer a structured tree view of the document, allowing the author to move among, select and cut, copy or paste elements of the document.
- A WYSIWYG tool may allow elements to be selected, and copied or moved while retaining their structure.
- A tool may allow transformation from one element type to another, such as
  1. HTML paragraphs to lists and back
  2. HTML `br` to `p`
  3. SMIL transformations between `switch`, `excl` and `par`
  4. HTML (deprecated) `FONT` into heuristically determined structure
  5. Lists of lists to tables and back
  6. MathML transformations between semantic and presentation markup
  7. Transforming SVG `g` elements to `symbol`
  8. Giving a structural role to a part of an element, such as an SVG `g` or an HTML `p`
- Amaya allows the author to select elements (including containers) and cut, copy and paste them with their attributes and properties in any of the formatted, structure and alternate views.

7.6 Allow the author to search within editing views . [Priority 2]

- Search functions are already present in almost every text and hypertext editing tools. The simplest allow searching for a sequence of characters, while more powerful searches can include the ability to perform searches that are case sensitive or case-insensitive, the ability to replace a search string, the ability to repeat a previous search to find the next or previous occurrence, or to select multiple occurrences with a single search.

- The ability to search for a particular type of structure is useful in a structured document, structured image such as a complex SVG image, etc.
- In an image editor the ability to select an area by properties (such as color, or closeness of color) is useful. This is common in middle range and high end image processing software.
- The ability to search a database for particular content, or to search a collection of files at once (a simple implementation of the latter is the Unix function "grep") is an important tool in managing large collections, especially those that are dynamically converted into Web content.
- The use of metadata (as per [WAI-WEBCONTENT]) can allow for very complex searching of large collections, or of timed presentations. Refer also to the paper "A Comparison of Schemas for Dublin Core-based Video Metadata Representation" [SEARCHABLE] for discussion specifically addressing timed multimedia presentations.
- Amaya provides a search function. Because all editing views are synchronized, any search text found will be selected in each of the available views.

3 Appendix - Sample Implementations

The Sample Implementations are collections of the above techniques for a specific type of tool. They have been developed to illustrate how the design principles embodied in the guidelines sections can be applied in various types of authoring tool.

3.1 Amaya

Amaya [AMAYA] is the W3C's testbed Web authoring/browsing platform. Its default editing view is WYSIWYG-style. The sample implementation [AMAYA-SAMPLE] outlines how Amaya Release version 2.1 conforms to the 3 September 1999 draft of the guidelines, and plans for improving conformance. **Note.** Amaya is developed as a proof of concept for a number of specifications, not a product for market.

3.2 Sketch

Sketch [SKETCH] is an open-source image editor. The version tested is 0.6.2, which provides an experimental SVG import/export functionality, although it only implements a few SVG elements as a proof of concept. It is written in python to enable easy user extension (and how to do this is well-documented).

3.3 The A-prompt Tool

The A-prompt tool [APROMPT] is an example tool that allows for checking of many accessibility features in HTML pages, and incorporates an "Alternative Information Management Mechanism" Refer also to checkpoint 3.5. to manage equivalent alternative information for known resources. The tool is built in such a way that the

functions can be incorporated into an authoring tool.

3.4 Alt-Text for the HTML 4.0 IMG Element

"Alt-text" is generally considered the most important aid to HTML accessibility. For this reason, the issue of "alt-text" has been chosen as the subject for an extended technique based on a hypothetical implementation.

7 Ensure that the authoring tool is accessible to authors with disabilities
: *Implementation*: The author can edit the document using the alternative information of the image in its place, and can access all the properties of the image (height, width, etc)

2 Generate standard markup
: *Implementation*: In any markup produced, the IMG element is always properly formed as defined in the HTML4 specification. This means that the element contains both a "src" attribute and an "alt" attribute.

1 Support accessible authoring practices
: *Implementation*: Due to the [WEB-CONTENT-PRIORITY] recommendation status of "alt-text" in the Web Content Accessibility Guidelines, special attention will be devoted to prompting and guiding the user toward full "alt" coverage. The authoring tool has the capability of opening and converting word processor documents into HTML. If an image is encountered during this process, the user will be prompted for "alt-text". The authoring tool sometimes makes changes to the HTML it works with to allow more efficient manipulation. These changes never result in the removal or modification of "alt-text" entries.

3 Support the creation of accessible content
: *Implementation*: The authoring tool is shipped with many ready-to-use clip art and other images. For each of these images a short "alt-text" string and a longer description have been pre-written and stored in an "alt-text" registry. When the user selects one of these images for insertion, the alternative text and long description are offered for editing and approval. Whenever the user includes another image, the tool keeps the reference to that image and the associated "alt-text" and long description in the "alt-text registry". When a text alternative offered by the tool is edited, the tool adds the new text to the registry, and offers both entries when the image is used again. There is an option to mark any entry as the default.

5 Integrate accessibility solutions into the overall "look and feel"
: *Implementation*: At no point do "alt-text" requests appear *on their own* or in a non-standard manner. Instead "alt-text" notices and emphasis appear as integrated and necessary as the "src" attribute.

4 Provide ways of checking and correcting inaccessible content
: *Implementation*: If the user opens content or pastes in markup containing an IMG element that lacks "alt-text", the author is prompted to add them. The tool can be configured to prompt as soon as an error is detected, or to provide a highlight mark where these errors occur and to prompt when the author is saving or publishing a document. The default prompt includes prompting for a long description of each image.

6 Promote accessibility in help and documentation

Implementation: Whenever missing "alt-text" is flagged (anywhere in the tool suite) the same quick explanation, extended help, and examples are offered. The help documentation for inserting images and image maps includes providing alternative text as part of the necessary steps, and describes how to determine appropriate alternative text in the same section. Examples of images and image-maps all have alternative text included, and images have long descriptions.

4 Glossary of Terms and Definitions

Accessibility (Also: ***Accessible***)

Within these guidelines, "accessible Web content" and "accessible authoring tool" mean that the content and tool can be used by people regardless of disability.

To understand the accessibility issues relevant to authoring tool design, consider that many users may be creating content in contexts very different from your own:

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

Accessible design will benefit people in these different authoring scenarios and also many people who do not have a physical disability but who have similar needs. For example, someone may be working in a noisy environment and thus require an alternative representation of audio information. Similarly, someone may be working in an eyes-busy environment and thus require an audio equivalent to information they cannot view. Users of small mobile devices (with small screens, no keyboard, and no mouse) have similar functional needs as some users with disabilities.

Accessibility Awareness

An accessibility-aware application is one that has been designed to account for users' differing needs, abilities, and technologies. In the case of authoring tools, this means that (1) care has been taken to ensure that the content produced by user-authors is accessible and (2) that the user interface has been designed to be usable with a variety of display and control technologies.

Accessibility Information

Accessibility information is content, including information and markup, that is used to improve the accessibility of a document. Accessibility information includes, but is not limited to, equivalent alternative information .

Accessibility Problem (Also: ***Inaccessible Markup***)

Inaccessible Web content or authoring tools cannot be used by some people with disabilities. The Web Content Accessibility Guidelines [WAI-WEBCONTENT] describes how to create accessible Web content.

Accessible Authoring Practice

Practices that improve the accessibility of Web content. Both authors and tools engage in accessible authoring practices. For example, authors write clearly, structure their content, and provide navigation aids. Tools generate valid markup and assist authors in providing and managing appropriate equivalent alternatives.

Alert

An alert draws the author's attention to an event or situation. It may require a response from the author. An alert warns the author that there are problems that need to be addressed. Attracting the user's attention artfully can be challenging,

since user perceptions of alerts, prompts, and warnings can influence opinions of the tool and even of accessible authoring.

An ***Unintrusive Alert*** is an alert such as an icon, underlining, or gentle sound that can be presented to the user without necessitating immediate action. For example, in some word processors misspelled text is highlighted without forcing the user to make immediate corrections. These alerts allow users to continue editing with the knowledge that problems will be easy to identify at a later time. However, users may become annoyed at the extra formatting or may choose to ignore the alerts altogether.

An ***Interruptive Alert*** is an informative message that interrupts the editing process for the user. For example, interruptive alerts are often presented when a user's action could cause a loss of data. Interruptive alerts allow problems to be brought to the user's attention immediately. However, users may resent the constant delays and forced actions. Many people prefer to finish expressing an idea before returning to edit its format.

Alternative Information (Also: ***Equivalent Alternative***)

Content is "equivalent" to other content when both fulfill essentially the same function or purpose upon presentation to the user. Equivalent alternatives play an important role in accessible authoring practices since certain types of content may not be accessible to all users (e.g., video, images, audio, etc.). Authors are encouraged to provide text equivalents for non-text content since text may be rendered as synthesized speech for individuals who have visual or learning disabilities, as braille for individuals who are blind, or as graphical text for individuals who are deaf or do not have a disability. For more information about equivalent alternatives, please refer to [WAI-WEBCONTENT] .

Text equivalents for still images can be short ("Site Map Link") or long (e.g., "Figure 4 shows that the population of bacteria doubled approximately every twenty hours over the first one hundred hours, increasing from about 1000 per milliliter to about 32,000 per milliliter."). Text equivalents for audio clips are called "text transcripts". Captions are essential text equivalents for movie audio. Another essential text equivalent for a movie is a "collated text transcript ." An essential non-text equivalent for movies is "auditory description " of the key graphical elements of a presentation.

Attribute

This document uses the term "attribute" as used in SGML and XML ([XML]): Element types may be defined as having any number of attributes. In the following example, the attributes of the `beverage` element type are `"flavour"`, which has the value "lots", and `"colour"`, which has the value "red":

```
<beverage flavour="lots" colour="red">my favourite</beverage>
```

Some attributes are integral to document accessibility (e.g., the `"alt"`, `"title"`, and `"longdesc"` attributes in HTML).

Auditory Description

An auditory description provides information about actions, body language, graphics, and scene changes in a video. They are commonly used by people

who are blind or have low vision, although they may also be used as a low-bandwidth equivalent on the Web. An auditory description is either a pre-recorded human voice or a synthesized voice (recorded or generated on the fly). The auditory description must be synchronized with the audio track of a video presentation, usually during natural pauses in the audio track.

Authoring Tool

An authoring tool is any software that is used to generate content for publishing on the Web. Authoring tools include:

- Editing tools specifically designed to produce Web content (e.g., WYSIWYG HTML and XML editors);
- Tools that offer the option of saving material in a Web format (e.g., word processors or desktop publishing packages);
- Tools that translate documents into Web formats (e.g., filters to translate desktop publishing formats to HTML);
- Tools that produce multimedia, especially where it is intended for use on the Web (e.g., video production and editing suites, SMIL authoring packages);
- Tools for site management or site publication, including tools that generate Web sites dynamically from a database, on-the-fly conversion and Web site publishing tools;
- Tools for management of layout (e.g., CSS formatting tools).

Automated Markup Insertion Function

Automated markup insertion functions are the features of an authoring tool that allow the user to produce markup without directly typing it. This includes a wide range of tools from simple markup insertion aids (such as a bold button on a toolbar) to markup managers (such as table makers that include powerful tools such as "split cells" that can make multiple changes) to high level site building wizards that produce almost complete documents on the basis of a series of user preferences.

Captions

Captions are essential text equivalents for movie audio. Captions consist of a text transcript of the audio track of the movie (or other video presentation) that is synchronized with the video and audio tracks. Captions are generally rendered graphically and benefit people who can see but are deaf, hard-of-hearing, or cannot hear the audio.

Conversion Tool

A conversion tool is any application or application feature that allows content in some other format (proprietary or not) to be converted automatically into a particular markup language. This includes software whose primary function is to convert documents to a particular markup language as well as "save as HTML" (or other markup language) features in non-markup applications.

Current User Selection

When several views co-exist, each may have a user selection, but only one is active, called the current user selection. The selections may be rendered specially (e.g., graphically highlighted).

Description Link (D-link)

A description link, or D-Link, is an author-supplied link to additional information about a piece of content that might otherwise be difficult to access (image, applet, video, etc.).

Document

A document is a series of elements that are defined by a markup language (e.g., HTML 4.0 or an XML application).

Editing an element

Editing an element involves making changes to one or more of an element's attributes or properties. This applies to all editing, including, but not limited to, direct coding in a text editing mode, making changes to a property dialog or direct User Interface manipulation.

Editing View

A view provided by the authoring tool that allows editing. Some authoring tools will have several different types of view, and some allow views of several documents at once.

Element

An element is any identifiable object within a document, for example a character, word, image, paragraph or spreadsheet cell. In [HTML40] and [XML] , an element refers to a pair of tags and their content, or an "empty" tag - one that requires no closing tag or content.

Focus

The focus designates the active element (e.g., link, form control, element with associated scripts, etc.) in a view that will react when the user next interacts with the document.

Generation Tool

A Generation Tool is a program or script that produces automatic markup "on the fly" by following a template or set of rules. The generation may be performed on either the server or client side.

Image Editor

A graphics program that provides a variety of options for altering images of different formats.

Inserting an element

Inserting an element involves placing that element's markup within the markup of the file. This applies to all insertions, including, but not limited to, direct coding in a text editing mode, choosing an automated insertion from a pull-down menu or tool bar button, "drag-and-drop" style insertions, or "paste" operations.

Markup Language

Authors encode information using a markup language such as HTML ([HTML40]), SVG ([SVG]), or MathML ([MATHML]).

Multimedia Authoring Tool

Software that facilitates integration of diverse media elements into an comprehensive presentation format. Multimedia includes video, audio, images, animations, simulations, and other interactive components.

Prompt

A prompt is a request for user input, either information or a decision. A prompt

requires author response. For example, an "alt-text" entry field prominently displayed in an image insertion dialog would constitute a prompt. Prompts can be used to encourage authors to provide information needed to make content accessible (such as alternative text equivalents).

Property

A property is a piece of information about an element, for example structural information (e.g., it is item number 7 in a list, or plain text) or presentation information (e.g., that it is marked as bold, its font size is 14). In XML and HTML, properties of an element include the name of the element (e.g., `IMG` or `DL`), the values of its attributes , and information associated by means of a style sheet. In a database, properties of a particular element may include values of the entry, and acceptable data types for that element.

Publishing Tool

A tool that allows content to be uploaded in an integrated fashion. Sometimes these tools makes changes such as local hyper-reference modifications. Although these tools sometimes stand alone, they may also be integrated into site management tools.

Rendered Content

The rendered content is that which an element actually causes to be rendered by the user agent. This may differ from the element's structural content. For example, some elements cause external data to be rendered (e.g., the `IMG` element in [HTML40]), and in some cases, browsers may render the value of an attribute (e.g., `"alt"`, `"title"`) in place of the element's content.

Rendered View, Preview

What is rendered by the authoring tool to the author as a means of simulating how a user of the document being edited will interact with the document currently being edited as a published document.

Selection

A selection is a set of elements identified for a particular operation. The user selection identifies a set of elements for certain types of user interaction (e.g., cut, copy, and paste operations). The user selection may be established by the user (e.g., by a pointing device or the keyboard) or via an accessibility Application Programmatic Interface (API). A view may have several selections, but only one user selection.

Site Management Tool

A tool that provides an overview of an entire Web site indicating hierarchical structure. It will facilitate management through functions that may include automatic index creation, automatic link updating, and broken link checking.

Transcript

A transcript is a line by line record of sounds within an audio clip, or an audio track from a video clip. A collated text transcript for a video combines (collates) caption text with text descriptions of video information (descriptions of the actions, body language, graphics, and scene changes of the video track). Collated text transcripts are essential for individuals who are deaf-blind and rely on braille for access to movies and other content.

Transformation

A process that changes a document or object into another, equivalent, object according to a discrete set of rules. This includes any application or application feature that allows content which is marked up in a particular markup language to be transformed into another markup language, such as a conversion tool, software that allows the author to change the DTD defined for the original document to another DTD, and the ability to change the markup of lists and convert them into tables.

User Agent

An application that retrieves and renders Web content. User agents include browsers, plug-ins for a particular media type, and some assistive technologies.

User-Configurable Schedule

A user-configurable schedule allows the user to determine the type of prompts and alerts that are used, including when they are presented. For example, a user may wish to include multiple images without being prompted for alternative information, and then provide the alternative information in a batch process, or may wish to be reminded each time they add an image. If the prompting is done on a user-configurable schedule they will be able to make that decision themselves. This technique allows a tool to suit the needs a wide range of authors.

Video Editor

A tool that facilitates the process of manipulating video images. Video editing includes cutting segments (trimming), re-sequencing clips, and adding transitions and other special effects.

View

Authoring tools may render the same content in a variety of ways; each rendering is called a view. For instance, one view may show raw markup, a second may show a structured tree, a third may show markup with rendered objects while a final view shows an example of how the document may appear if it were to be rendered by a particular browser. A typical way to distinguish views in a graphic environment is to place each in a separate window.

5 Acknowledgments

Many thanks to the following people who have contributed through review and comment: Jim Allan, Denis Anson, Kitch Barnicle, Kynn Bartlett, Harvey Bingham, Judy Brewer, Carl Brown, Dick Brown, Wendy Chisholm, Rob Cumming, Daniel Dardailler, Mark Day, BK Delong, Martin Dürst, Kelly Ford, Jamie Fox, Edna French, Sylvain Galineau, Al Gilman, Eric Hansen, Phill Jenkins, Len Kasday, Brian Kelly, Marja-Riitta Koivunen, Sho Kuwamoto, Jaap van Lelieveld, William Loughborough, Karen McCall, Charles Oppermann, Dave Pawson, Dave Poehlman, Bruce Roberts, Chris Ridpath, Gregory Rosmaita, Janina Sajka, John Slatin, Jim Thatcher, Irène Vatton, Gregg Vanderheiden, Pawan Vora, Jason White, and Lauren Wood.

6 References

For the latest version of any W3C specification please consult the list of W3C Technical Reports.

[ACCESS-AWARE]
"The Three-tions of Accessibility-Aware HTML Authoring Tools," J. Richards.

[AMAYA]
Amaya W3C's own browser/authoring tool, used to demonstrate and test many of the new developments in Web protocols and data formats. Amaya has a WYSIWYG style of interface. Source code, binaries, and further information are all available at http://www.w3.org/Amaya/.

[AMAYA-HELP-IMG]
"Images and Client-side Image Maps" Amaya's Help page for images and image maps.

[AMAYA-SAMPLE]
Amaya - Authoring Tool Accessibility Guidelines example" Describes how Amaya, W3C's WYSIWYG browser/authoring tool, implements the guidelines.

[APPLE-HI]
"Macintosh Human Interface Guidelines," Apple Computer Inc.

[APROMPT]
A-prompt tool is a freely available example tool developed by the Adaptive Technology Resource Center at the University of Toronto, and the TRACE center at the University of Wisconsin. The source code for the tool is also available at http://aprompt.snow.utoronto.ca

[AUTO-TOOL]
"Techniques For Evaluation And Implementation Of Web Content Accessibility Guidelines," C. Ridpath.

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

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

[CSS2-ACCESS]
"Accessibility Features of CSS," I. Jacobs and J. Brewer, eds., 4 August 1999. This version is http://www.w3.org/1999/08/NOTE-CSS-access-19990804. The latest version of Accessibility Features of CSS is available at http://www.w3.org/TR/CSS-access.

[ED-DEPT]
"Requirements for Accessible Software Design," US Department of Education, version 1.1 March 6, 1997.

[EITAAC]
"EITACC Desktop Software standards," Electronic Information Technology Access Advisory (EITACC) Committee.

[HTML4-ACCESS]
""WAI Resources: HTML 4.0 Accessibility Improvements," I. Jacobs, J. Brewer, and D. Dardailler, eds. This document describes accessibility features in HTML 4.0.

[HTML40]
"HTML 4.0 Recommendation," D. Raggett, A. Le Hors, and I. Jacobs, eds., 17 December 1997, revised 24 April 1998. This HTML 4.0 Recommendation is http://www.w3.org/TR/1998/REC-html40-19980424. The latest version of HTML 4.0 is available at http://www.w3.org/TR/REC-html40.

[IBM-ACCESS]
"Software Accessibility," IBM Special Needs Systems.

[ICCCM]
"The Inter-Client communication conventions manual." A protocol for communication between clients in the X Window system.

[ICE-RAP]
"An ICE Rendezvous Mechanism for X Window System Clients," W. Walker. A description of how to use the ICE and RAP protocols for X Window clients.

[JAVA-ACCESS]
"IBM Guidelines for Writing Accessible Applications Using 100% Pure Java," R. Schwerdtfeger, IBM Special Needs Systems.

[JAVA-CHECKLIST]
"Java Accessibility Guidelines and Checklist," IBM Special Needs Systems.

[JAVA-TUT]
"The Java Tutorial. Trail: Creating a GUI with JFC/Swing." An online tutorial that describes how to use the Swing Java Foundation Class to build an accessible User Interface.

[MATHML]
"Mathematical Markup Language," P. Ion and R. Miner, eds., 7 April 1998, revised 7 July 1999. This MathML 1.0 Recommendation is http://www.w3.org/TR/1998/REC-MathML-19990707. The latest version of MathML 1.0 is available at http://www.w3.org/TR/REC-MathML.

[MS-ENABLE]
"Accessibility for Applications Designers," Microsoft Corporation.

[MS-SOFTWARE]
"The Microsoft Windows Guidelines for Accessible Software Design." **Warning!** This is a "self-extracting archive", an application that will probably only run on MS-Windows systems.

[MSAA]
"Information for Developers About Microsoft Active Accessibility," Microsoft Corporation.

[NOTES-ACCESS]
"Lotus Notes Accessibility Guidelines," IBM Special Needs Systems.

[SEARCHABLE]
"A Comparison of Schemas for Dublin Core-based Video Metadata Representation," J Hunter.

[SKETCH]
The Sketch open source image editor home page.

[SMIL-ACCESS]
"Accessibility of SMIL", M.-R. Koivunen, I. Jacobs eds. The latest version is available at http://www.w3.org/TR/SMIL-access

[SUN-DESIGN]
"Designing for Accessibility," Eric Bergman and Earl Johnson. This paper discusses specific disabilities including those related to hearing, vision, and cognitive function.

[SUN-HCI]
"Towards Accessible Human-Computer Interaction," Eric Bergman, Earl Johnson, Sun Microsytems 1995. A substantial paper, with a valuable print bibliography.

[SVG]
"Scalable Vector Graphics (SVG) 1.0 Specification" (Working Draft), J. Ferraiolo, ed. The latest version of the SVG specification is available at http://www.w3.org/TR/SVG

[SVG-ACCESS]
"Accessibility of Scalable Vector Graphics" (Working Draft), C. McCathieNevile, M.-R. Koivunen eds. The latest version is available at http://www.w3.org/1999/09/SVG-access

[TRACE-REF]
"Application Software Design Guidelines," compiled by G. Vanderheiden. A thorough reference work.

[WAI-AUTOOLS]
"Techniques for Authoring Tool Accessibility," J. Treviranus, J. Richards, I. Jacobs, and C. McCathieNevile eds. The latest version is available at http://www.w3.org/WAI/AU/WAI-AUTOOLS-TECHS.

[WAI-ER]
The Web Accessibility Initiative Evaluation and Repair Tools Working Group tracks and develops tools that can help repair accessibility errors.

[WAI-USERAGENT]
"User Agent Accessibility Guidelines," J. Gunderson and I. Jacobs, eds. The latest version of the User Agent Accessibility Guidelines is available at http://www.w3.org/WAI/UA/WAI-USERAGENT.

[WAI-WEBCONTENT]
"Web Content Accessibility Guidelines 1.0," W. Chisholm, G. Vanderheiden, and I. Jacobs, eds., 5 May 1999. This Recommendation is http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505. The latest version of the Web Content Accessibility Guidelines 1.0" is available at http://www.w3.org/TR/WAI-WEBCONTENT/.

[WAI-WEBCONTENT-TECHS]
"Techniques for Web Content Accessibility Guidelines 1.0," W. Chisholm, G.

Vanderheiden, and I. Jacobs, eds. The latest version of Techniques for Web Content Accessibility Guidelines 1.0 is available at http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/.

[WEB-CONTENT-PRIORITY]
Priorities defined by [WAI-WEBCONTENT] .

[WHAT-IS]
"What is Accessible Software," James W. Thatcher, Ph.D., IBM, 1997. This paper gives a short example-based introduction to the difference between software that is accessible, and software that can be used by some assistive technologies.

[XHTML10]
"XHTML(TM) 1.0: The Extensible HyperText Markup Language (Working Draft)," S. Pemberton et al. The latest version of XHTML 1.0 is available at http://www.w3.org/TR/xhtml1.

[XML]
"The Extensible Markup Language (XML) 1.0," T. Bray, J. Paoli, C. M. Sperberg-McQueen eds. The latest version of The XML Specification is available at http://www.w3.org/TR/REC-xml.

[XMLGL]
"XML Accessibility Guidelines (Draft Note)," D. Dardailler ed. Draft notes for producing accessible XML document types. The latest version of the XML Accessibility Guidelines is available at http://www.w3.org/WAI/PF/xmlgl.