W3C WD-WAI-AUTOOLS-19981112

WAI Authoring Tool Guidelines

W3C Working Draft *12-Nov-1998*

This version:
http://www.w3.org/TR/1998/WD-WAI-AUTOOLS-19981112
Latest version:
http://www.w3.org/TR/WD-WAI-AUTOOLS

Editors:
Jutta Treviranus <jutta.treviranus@utoronto.ca>
Jan Richards <jan.richards@utoronto.ca>
Nancy Sicchia <nancy.sicchia@utoronto.ca>
Ian Jacobs <ij@w3.org>

Copyright © 1998 W3C (MIT, INRIA, Keio), All Rights Reserved.

Abstract

This document provides guidelines to authoring tool manufacturers or developers. The purpose of this document is two-fold: to assist developers in designing authoring tools that generate accessible Web content and to assist developers in creating an accessible authoring tool user interface. Accessible Web content is achieved by encouraging authoring tool users to create accessible Web content (through mechanisms such as prompts, alerts, checking and repair functions, help files and automated tools), but also by ensuring that the automatic processes of the authoring tool generate accessible content. This will result in authoring tools that can be used by a broader range of users and in the proliferation of Web pages that can be read by a broader range of readers.

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

Status of this document

This is the first public Working Draft of the W3C WAI Authoring Tool Guidelines. W3C Members, the general public, and all interested parties are invited to review this document. It is a draft document and may be updated, replaced or obsoleted 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, or the consensus of, either W3C or members of the WAI Authoring Tool (AU) Working Group.

The goals of the WAI AU Working Group are discussed in the WAI AU charter. A list of the current AU Working Group members is available.

Available formats

This document is available in the following formats:

HTML:
http://www.w3.org/TR/1998/WD-WAI-AUTOOLS-19981112/wai-autools.html
A plain text file:
http://www.w3.org/TR/1998/WD-WAI-AUTOOLS-19981112/wai-autools.txt,
HTML as a gzip'ed tar file:
http://www.w3.org/TR/1998/WD-WAI-AUTOOLS-19981112/wai-autools.tgz,
HTML as a zip file (this is a '.zip' file not an '.exe'):
http://www.w3.org/TR/1998/WD-WAI-AUTOOLS-19981112/wai-autools.zip,
A PostScript file:
http://www.w3.org/TR/1998/WD-WAI-AUTOOLS-19981112/wai-autools.ps,
A PDF file:
http://www.w3.org/TR/1998/WD-WAI-AUTOOLS-19981112/wai-autools.pdf.

In case of a discrepancy between the various formats of the specification, http://www.w3.org/TR/1998/WD-WAI-AUTOOLS-19981112/wai-autools.html is considered the definitive version.

Comments

Please send comments about this document to the public mailing list: w3c-wai-au@w3.org.

Table of Contents

1 Introduction 5
1.1 Guideline Priorities 5
2 Integrate Accessibility Awareness 5
2.1 Generate standard markup [Priority 1] 5
2.2 Support all accessible content recommendations [Priority 1] . . .5
2.3 Ensure that users may configure accessibility mechanisms [Priority 1] .6
2.4 Emphasize accessible authoring practices [Priority 1] 6
2.5 Identify all inaccessible markup [Priority 1] 7
2.6 Provide mechanisms for authors to add missing alternative representations for the content of converted documents [Priority 1] 7
2.7 Promote accessibility awareness in tool suites [Priority 2] 8
2.8 Integrate accessibility solutions naturally [Priority 2] 8
3 Provide Accessibility Information 9
3.1 Provide comprehensive accessibility help to authors [Priority 1] . . .9
3.2 Provide rationales that stress Universal Design [Priority 1] 9
3.3 Package multimedia files with pre-written descriptions [Priority 2] . 10
3.4 Promote accessibility in all Help examples [Priority 3] 11
3.5 Provide the author with progress feedback [Priority 3] 11
4 Automate Accessibility Tasks 11
4.1 Integrate accessibility solutions into relevant automated tools and wizards [Priority 1] 11
4.2 Allow the user to check for and correct accessibility problems automatically [Priority 1] 12
4.3 Ensure that all markup inserted by the authoring tool is accessible [Priority 1] 12
4.4 Ensure that conversion tools produce and retain accessible markup and content [Priority 1] 12
5 Ensure an Accessible Authoring Environment 13
5.1 Provide optional views of the edited document 13
5.2 Offer text representations of elements 13
5.3 Offer text representations of site maps 14
6 Accessible Authoring Support Mechanisms 14
6.1 Integrated Author Guidance and Prompting 14
6.2 Alert Techniques 14
6.3 Automated Tools and Wizards 15
6.4 Integrated Checking, Verification and Repair Mechanisms . . . 15
6.5 Context Sensitive Help and Documentation 15
7 Sample Implementations 15
7.1 Alt-Text for the HTML 4.0 IMG Element 16
7.2 Tool: "Alt"-text registry 17
8 Terms and Definitions 18
8.1 Markup Editing Tools and Functions 18
8.2 Documents, Elements, and Attributes 19
8.3 Accessibility Terms 19
8.4 Alternative Representation of Content 20

8.5 Inserting and Editing 20
8.6 Alert Techniques 20
8.7 Selection, Focus, and Events 21
9 Acknowledgments 21
10 References 21

1 Introduction

The guidelines in this document are meant to help authoring tool developers and vendors design products that encourage authors to adopt accessible authoring practices. For the purposes of this document the term "authoring tool" will refer to authoring tools [p. 18] , generation tools [p. 18] and conversion tools [p. 18] . These guidelines emphasize the role of the user interface in informing, supporting, correcting and motivating authors during the editing process. For a more detailed discussion of accessible Web authoring practices, see the WAI Page Author Guidelines [p. 21] document.

1.1 Guideline Priorities

To assist in implementing the guidelines, each guideline in this document is assigned a priority.

[Priority 1]
: This guideline is fundamental to the creation of accessible documents by authoring tools.

[Priority 2]
: This guideline is important to the creation of accessible documents by authoring tools.

[Priority 3]
: This guideline promotes the creation of accessible documents by authoring tools.

This document also refers to guidelines and techniques defined in the WAI Page Author Guidelines [p. 21] and to priorities assigned to them (indicated, for example, by [Page-Author-Priority 1] [p. 6]).

2 Integrate Accessibility Awareness

Guiding Principle: The acceptance of accessible authoring practices requires that accessibility become a fundamental part of authoring tools.

2.1 Generate standard markup [Priority 1]

The first step towards accessibility is interoperability. Authoring tools should ensure that content is created in accordance with W3C specifications (or other standards when applicable).

Example:

HTML4: "alt" Attribute for IMG [p. 16]

2.2 Support all accessible content recommendations [Priority 1]

Methods for ensuring accessible markup vary with different markup languages. An authoring tool must support all accessibility features that have been defined for the markup language(s) supported by the tool. Listing the accessibility features of specific languages lies beyond the scope of this document. However, an informative list of documents that address accessible Web authoring practices follows.

Page Author Accessibility Features: (The actual accessible markup solutions)

- General: WAI Page Authoring Guidelines: Techniques [p. 22]
- HTML4: HTML4 Accessibility Improvements [p. 22]
- CSS2: CSS2 Accessibility Improvements [p. 22]
- SMIL, MathML

Page Author Implementation Priorities: (The priorities placed on the accessibility markup solutions)

- General: WAI Page Author Guidelines [p. 21]

Mechanisms that can be employed by authoring tools to support accessible authoring practices are discussed in Section 5 [p. 14] .

Example:

HTML4: "alt" Attribute for IMG [p. 16]

2.3 Ensure that users may configure accessibility mechanisms [Priority 1]

In supporting the creation of accessible Web content, authoring tools must take into account the differing authoring styles of their users. Some users may prefer to be alerted to problems when they occur, whereas others may prefer to perform a check after the document is completed. This is analogous to programming environments that allow users to decide whether to check for correct code during editing or at compile time.

Techniques:

[Technique: 2.3.1] [Priority 1]
: Authoring tools must be designed so that users can indicate their preferences regarding both the nature and timing of accessibility alerts.

[Technique: 2.3.2] [Priority 1]
: The priority level given to accessibility recommendations for a given language should be taken into account when determining the level of user control. Specifically, the user should have the option of determining the extent of alerts for [Page-Author-Priority 2] [p. 6] and [Page-Author-Priority 3] [p. 6] recommendation items.

[Technique: 2.3.3] [Priority 1]
: Users should be prevented from completely disabling alerts for [Page-Author-Priority 1] [p. 6] items. The author should at least be notified using a non-intrusive alert.

Example:

HTML4: "alt" Attribute for IMG [p. 16]

2.4 Emphasize accessible authoring practices [Priority 1]

Recommended accessible authoring practices [p. 6] (and their priorities [p. 6]) must be taken into account during the design of relevant user interface components and program functionality.

Techniques:

[Technique: 2.4.1] [Priority 1]
When more than one means of performing a particular authoring task is available, the most accessible means of performing that task should be the most visible and easily initiated.

[Technique: 2.4.2] [Priority 1]
In cases where user input is ambiguous, authoring tools should always assume that the author intends to maintain accessibility.

[Technique: 2.4.3] [Priority 1]
When a [Page-Author-Priority 1] [p. 6] guideline advises against an authoring practice, authoring tools must not recommend or otherwise encourage its use.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

2.5 Identify all inaccessible markup [Priority 1]

Authoring tools must be designed so that all processes that manipulate markup text are sensitive to the existence of inaccessible markup [p. 19] . **Note.** the term *checking* below does not necessarily imply that the user must be *alerted* immediately.

Techniques:

[Technique: 2.5.1] [Priority 1]
Check existing documents when they are opened for editing.

[Technique: 2.5.2] [Priority 1]
Check documents during all types of editing [p. 20] (including hand-coding, paste operations, and code insertions).

[Technique: 2.5.3] [Priority 1]
When problems are detected, the author should be alerted according to a user-configurable schedule. See the section on ensuring that users may configure accessibility mechanisms [p. 6] and Alert Techniques. [p. 14]

Example:
HTML4: "alt" Attribute for IMG [p. 16]

2.6 Provide mechanisms for authors to add missing alternative representations for the content of converted documents [Priority 1]

When an application converts documents from other formats into a markup format such as HTML, author intervention may be required to make the resulting documents accessible.

Techniques:

[Technique: 2.6.1] [Priority 1]
The author must be prompted, on a configurable schedule, to provide "alt"-text for images, image maps, and image map links.

[Technique: 2.6.2] [Priority 1]
The authoring tool must never insert rule-generated description text into the document (default "alt"-text) or a properties field (place-holder "alt"-text).

Automated processes may place pre-authored (by a person) text when the meaning or function of the described object is known with certainty.

[Technique: 2.6.3] [Priority 1]

The author must be prompted to provide captioning or transcriptions for any video or audio segments.

[Technique: 2.6.4] [Priority 3]

The author should be given the option of providing a long description for any graphic element.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

2.7 Promote accessibility awareness in tool suites [Priority 2]

Language markup authoring tools are often integrated into Web publishing suites that also include peripheral tools for handling non-coding tasks.

Techniques: (A sample list of peripheral tools appears below with accompanying sample techniques)

[Technique: 2.7.1] [Priority 2]

Site Management Tools: These tools already routinely perform site management checks, such as identifying broken links. The ability to check for and correct [Page-Author-Priority 1] [p. 6] accessibility problems should be added to these tools.

[Technique: 2.7.2] [Priority 2]

Image, Audio and Video Editors: Image editors should be modified to encourage people to write short "alt"-text and long description text for any images created or edited. Audio and video editors should encourage the user to create long descriptions, transcripts and captions. Once created, these associated descriptions might then be cataloged where they can be used to provide image/audio/video library search capabilities and default descriptive text, should the described object be added to a Web document in the future.

[Technique: 2.7.3] [Priority 2]

Publishing Tool: Publishing tools should be modified to prompt for and correct [Page-Author-Priority 1] [p. 6] accessibility problems before content is published.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

2.8 Integrate accessibility solutions naturally [Priority 2]

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.

Techniques:

[Technique: 2.8.1] [Priority 2]

In order to gain user acceptance of accessible authoring, it is imperative that any new functionality associated with accessibility be properly integrated into the overall "look and feel" of the authoring tool.

[Technique: 2.8.2] [Priority 2]
Accessibility features should never interfere with any of the expected operations of an author's editing environment. For example, *fundamental* operations such as saving, closing, and pasting should not be canceled or postponed due to the existence of accessibility problems.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

3 Provide Accessibility Information

Guiding Principle: Help files, justifications and examples are critical if authors are to become more aware of and implement Web accessibility principles.

3.1 Provide comprehensive accessibility help to authors [Priority 1]

The issues surrounding Web accessibility are often unknown to Web authors. Providing convenient links to clear and concisely written help files will contribute to author acceptance of, and education about, markup accessibility.
Techniques:

[Technique: 3.1.1] [Priority 1]
Implement context sensitive help for all special accessibility terms, as well as tasks related to accessibility. Mechanisms used to identify accessibility problems such as icons, outlining or other emphasis within the user interface should be linked to help files as these may appear unfamiliar to the user.
[Technique: 3.1.2] [Priority 1]
The accessibility help files should explain the accessibility problem or accessibility feature quickly, with emphasis placed on the solutions available rather than placing emphasis on elements that have been incorrectly marked up.
[Technique: 3.1.3] [Priority 1]
The help files should include many examples as well as links to any automated correction utilities.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

3.2 Provide rationales that stress Universal Design [Priority 1]

Most users are unfamiliar with accessibility issues on the Web. By incorporating explanations of universal design benefits into authoring tools, authors will better understand the value of accessible page design.
Techniques:

[Technique: 3.2.1] [Priority 1]
The help system should explain the importance of utilizing accessibility features generally and for specific instances.

[Technique: 3.2.2] [Priority 1]

Explanations should emphasize the Universal Design principle of supporting flexible display and control choices, that are critical for:

- hands free, eyes-free, voice-activated browsing devices such as Web phones
- the large number of slow Web connections (not everyone has a fiber connection)
- Web users who prefer text-only browsing to avoid "image clutter"
- the aging population (with the accompanying decrease in visual, hearing, motor, and cognitive abilities)
- the relatively high Web presence of people with sensory and motor disabilities.

Example:
HTML4: "alt" Attribute for IMG [p. 16]
Information:
For more information on Universal Design, visit the Trace Center.

3.3 Package multimedia files with pre-written descriptions [Priority 2]

Textual descriptions, including "alt"-text, long descriptions, video captions, and transcripts are absolutely necessary for the accessibility of all images, applets, video, and audio files. However, the task of writing these descriptions is probably the most time-consuming accessibility recommendation made to the author.

Techniques:

[Technique: 3.3.1] [Priority 2]

Include professionally written descriptions for all multimedia files packaged with the authoring tool (e.g. clip art) so that:

1. users will be saved time and effort
2. a significant number of professionally written descriptions will begin to circulate
3. users will be provided with convenient models to emulate when they write their own descriptions
4. users will see evidence of the importance of description writing

[Technique: 3.3.2] [Priority 2]

The authoring tool should make use of the pre-written descriptions by suggesting them as default text whenever one of the associated files is inserted into the author's document.

[Technique: 3.3.3] [Priority 2]

Increase user acceptance of pre-written descriptions by allowing authors to make keyword searches of the description database, thus simplifying the task of finding relevant images.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

3.4 Promote accessibility in all Help examples [Priority 3]

In addition to a help section dedicated to accessibility [p. 9] , accessibility principles should be followed for *all* applicable markup examples in the rest of the help system. This will increase integration and help show authors that accessibility is a normal part of authoring, rather than a separate concern.

Techniques:

[Technique: 3.4.1] [Priority 3]
: All markup practices that require accessible solutions should appear with those solutions (ex. IMG elements should appear with "alt"-text)

[Technique: 3.4.2] [Priority 3]
: Lower priority accessibility solutions should also be included to increase the wider acceptance of these standards.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

3.5 Provide the author with progress feedback [Priority 3]

Achieving accessibility requires some extra effort and cooperation from the author. In order to maintain user goodwill and acceptance of accessible authoring practices, the user should receive progress feedback regarding satisfied accessibility objectives.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

4 Automate Accessibility Tasks

Guiding Principle: The power of automation should be utilized to free authors to concentrate on description writing and other higher-level aspects of Web accessibility.

4.1 Integrate accessibility solutions into relevant automated tools and wizards [Priority 1]

Accessibility issues often arise in complex markup tasks. Many authoring tools provide automated tools or wizards to guide authors in creating the relatively complex syntax needed to insert items such as objects, tables, or frames.

Techniques:

[Technique: 4.1.1] [Priority 1]
: When tasks for which page author accessibility guidelines exist are automated, the relevant accessibility solutions must be incorporated into the tool or wizard.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

4.2 Allow the user to check for and correct accessibility problems automatically [Priority 1]

Many authoring tools allow their users to create documents with little or no knowledge about the underlying markup.

Techniques:

[Technique: 4.2.1] [Priority 1]
In order to ensure that documents are made accessible according to the WAI Page Author Guidelines, [p. 21] authoring tools must include an automated tool that identifies and helps correct accessibility problems.

[Technique: 4.2.2] [Priority 1]
The correcting tool must be designed in such a way that authors can correct accessibility problems without necessarily understanding the underlying structure of the language or the details of markup accessibility.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

4.3 Ensure that all markup inserted by the authoring tool is accessible [Priority 1]

If markup is automatically generated, the author will be unaware of the accessibility status of the final product unless they expend extra effort to make appropriate corrections by hand. Since most authors are unfamiliar with accessibility, these problems are likely to remain.

Techniques:

[Technique: 4.3.1] [Priority 1]
Automated markup insertion functions [p. 19] must make use of appropriate accessible solutions, even if this means presenting the author with extra prompts [p. 20] for necessary information or structure during or following the process.

[Technique: 4.3.2] [Priority 1]
Automatic tools must make use of algorithms that produce accessible markup [p. 19] .

Example:
HTML4: "alt" Attribute for IMG [p. 16]

4.4 Ensure that conversion tools produce and retain accessible markup and content [Priority 1]

Many applications feature the ability to convert documents from other formats (e.g., Rich Text Format) or proprietary formats into a markup format, such as HTML. Markup changes may also be made to facilitate efficient editing and manipulation. These processes are usually hidden from the user’s view and may create inaccessible content or cause inaccessible content to be produced.

Techniques:

[Technique: 4.4.1] [Priority 1]
Conversion utilities must generate documents that respect the WAI Page Author Guidelines [p. 21] .
[Technique: 4.4.2] [Priority 1]
Authoring tools must never remove or modify structure or content that is necessary for continued accessibility.
[Technique: 4.4.3] [Priority 1]
Authors must be provided with the option of a receiving a summary of all automated structural changes that may affect accessibility.

Example:
HTML4: "alt" Attribute for IMG [p. 16]

5 Ensure an Accessible Authoring Environment

Web authors have a broad range of skills and needs. Guidelines in this section address the accessibility of the authoring tools to Web authors. Authoring tools, like other software applications can be made accessible in two ways:

1. by building in a range of options for displaying information and controlling the application,
2. by making the tool compatible with third party assistive technology (e.g., text to speech devices or alternative keyboards).

Authoring tools are similar to and frequently include the functionality of user agents. Most of the guidelines for providing access to the authoring tool are covered in the WAI User Agent Guidelines. Authoring tools should therefore comply with the User Agent Guidelines. Issues not addressed by the User Agent Guidelines are related to the additional and unique functionality of authoring tools.

5.1 Provide optional views of the edited document

When creating or editing a Web page the desired ultimate rendering of the page may not be optimal for creating and editing.
Techniques:

[Technique: 5.1.1] [Priority 1]
The authoring tools should support at least two views:
1. an authoring/editing view
2. a publishing or browser view, (similar to the normal and page view or print preview of popular word processors).

[Technique: 5.1.2] [Priority 1]
In the authoring/editing view, the font size, letter and line spacing, and text and background color should be independent of the final format of the document.

5.2 Offer text representations of elements

Graphically represented elements cannot be identified by third-party assistive technologies that translate text to Braille, speech, or large print. Some authoring tools display the start and end tags as graphics.

Techniques:

[Technique: 5.2.1] [Priority 1]
Allow the author to display tags in a text format.
[Technique: 5.2.2] [Priority 1]
To help distinguish the start or end tag from the remainder of the document, the tags should be surrounded by text brackets.

5.3 Offer text representations of site maps

Graphic representation of Web pages or Web site elements in site management tools cannot be identified by third-party assistive technologies that translate text to Braille, speech, or large print.
Techniques:

[Technique: 5.3.1] [Priority 1]
Allow the author to display the site map in text form (e.g., as a structured tree file).

6 Accessible Authoring Support Mechanisms

The authoring tool should guide and assist the author in creating accessible content. This can be done at various stages of the authoring process and in various ways. This section describes several user interface mechanisms that can be used to support accessible authoring practices. If implemented, these support mechanisms should be an integral part of the authoring tool user interface ("Integrate accessibility features naturally" [p. 8]) and the author should have the freedom to decide when and how accessibility will be addressed ("Allow authors to configure the level of accessibility awareness" [p. 6]). These descriptions are meant only as a guide, since the actual implementation will depend on the "look and feel" of the authoring tool itself.

6.1 Integrated Author Guidance and Prompting

Interface mechanisms such as dialogs, menus, toolbars, and palettes can be structured so that markup or elements that are accessible are given as the first and easiest choice.

Prompts can be used to encourage authors to provide information needed to make the content accessible (such as alternative textual representations). Prompts are simple requests for information before a markup structure has been finalized. 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.

6.2 Alert Techniques

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 in which users are alerted, prompted, or warned will influence their view of the tool as well as their opinion of accessible authoring.

The following are sample alert possibilities with a short definition and a brief discussion of their advantages and disadvantages.

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.

6.3 Automated Tools and Wizards

Just as some more tedious, repetitive, or complex authoring processes are automated in many authoring tools, processes required to provide accessible content can be automated or simplified through wizards. Examples include an "alt"-text registry, a captioning tool, a transcription tool, or a table tool.

6.4 Integrated Checking, Verification and Repair Mechanisms

Integrated checking and repair mechanisms allow the author to check the content once the content is created or upon importing content (using a batch detection process). Detected problems would be highlighted and linked to tools or dialogs that can be used to repair or address the problem. The timing of the batch detection might be determined by the user as in the case of a spell checker or as in the case of syntax checking prior to compiling code. Although the author is allowed full freedom to create and edit their content, if access problems are not addressed until the end of the authoring process, the user may be presented with an extensive list of problems that might include important organizational changes.

6.5 Context Sensitive Help and Documentation

Help files and documentation are essential mechanisms for informing authors and promoting accessible authoring practices. Please consult the section on providing accessibility information [p. 9] for details.

7 Sample Implementations

The Sample Implementations are *not* guidelines. The section has been included to illustrate how the design principles embodied in the guidelines sections can be applied to concrete issues. The specific ideas discussed in this section are meant to be used only as clarification.

7.1 Alt-Text for the HTML 4.0 IMG Element

"Alt"-text is generally considered the most important aid to accessibility. For this reason, the issue of "alt"-text has been chosen as the subject for the first sample implementation.

2.1 Generate standard markup [Priority 1] [p. 5]
Implementation: In any content 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.

2.2 Support all accessible content recommendations [Priority 1] [p. 5]
Implementation: Due the [Page-Author-Priority 1] [p. 6] recommendation status of "alt"-text in the WAI Page Author Guidelines, special attention will be devoted to prompting and guiding the user toward full "alt" coverage.

2.3 Ensure that users may configure accessibility mechanisms [Priority 1] [p. 6]
Implementation: A configuration system allows the user to decide whether they wish to be reminded each time they place an IMG element without "alt"-text or if they will complete the "alt"-text entry task at a later time. The configuration system does not contain the option of disabling "alt"-text checking completely. Other options allow the user to specify the behavior of the "alt"-text registry [p. 17] .

2.4 Emphasize accessible authoring practices [Priority 1] [p. 6]
Implementation: The "alt" attribute appears immediately below the "src" attribute in the image properties listing. Whenever the properties for an image without "alt"-text are examined, visual highlighting of the "alt" entry field remind the user that "alt"-text should not be left empty. In addition, when an image without "alt"-text is selected, *Insert "alt"-text* is one of the options presented to the user.

2.5 Identify all inaccessible markup [Priority 1] [p. 7]
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 (unless they have configured the tool to postpone this task).

2.6 Provide mechanisms for authors to add missing alternative representations for the content of converted documents [Priority 1] [p. 7]
Implementation: The authoring tool makes the task of entering "alt"-text as easy as possible for the user. Specifically, the authoring tool handles the markup syntax (as it does for other attributes). In addition, an "alt"-text registry [p. 17] is provided to store and retrieve the "alt"-text previously used for a user's images.

2.7 Promote accessibility awareness in tool suites [Priority 2] [p. 8]
Implementation: The image editing tool that is shipped with the authoring tool prompts users to enter alt-text after the first save and after close requests. By making use of the "alt"-text registry [p. 17] , the tool offers the user the choice of using this text as the "alt"-text anywhere the image appears on the site. The site management tool checks each constituent document and flags those missing "alt"-text in the user's site view. Quick links are provided to correct the problems.

2.8 Integrate accessibility solutions naturally [Priority 2] [p. 8]
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.

3.1 Provide comprehensive accessibility help to authors [Priority 1] [p. 9]
Implementation: Whenever missing "alt"-text is flagged (anywhere in the tool suite) the same quick explanation, extended help, and examples are offered.

3.2 Provide rationales that stress Universal Design [Priority 1] [p. 9]
Implementation: In addition to describing the need for "alt"-text for access by people with visual disabilities, the rationales mention how "alt"-text allows users of Web phones and other non-visual browsing technologies to access the content of the image.
3.3 Package multimedia files with pre-written descriptions [Priority 2] [p. 10]
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 the "alt"-text registry [p. 17] .
3.5 Provide the author with progress feedback [Priority 3] [p. 11]
Implementation: Whenever an accessibility checker completes a run, a summary list of accessibility issues is presented. When the user has entered "alt"-text for all the images in a document, the "alt"-text completed box will be checked in the summary. This box will remain checked as long as no images without "alt"-text are added.
3.4 Promote accessibility in all Help examples [Priority 3] [p. 11]
Implementation: Whenever the IMG element appears in the help system, the "alt" attribute is always present. Links to "alt"-text specific help and rationale are provided.
4.1 Integrate accessibility solutions into relevant automated tools and wizards [Priority 1] [p. 11]
Implementation: The authoring tool includes an in-line image editor that allows the user to manipulate images on their page. If an IMG element does not include "alt"-text, then the user prompted (unless they have postponed this task) to enter this text as they click elsewhere in the document to close the editor.
4.2 Allow the user to check for and correct accessibility problems automatically [Priority 1] [p. 12]
Implementation: An accessibility checker utility scans the active document like a spell checker. If an IMG element is found without "alt"-text, a prompt is displayed with "alt"-text registry [p. 17] default text, if it is available.
4.3 Ensure that all markup inserted by the authoring tool is accessible [Priority 1] [p. 12]
Implementation: If the user drags an image from the desktop into the authoring tool, the user will be prompted for "alt"-text for the IMG element (unless the user has postponed this task).
4.4 Ensure that conversion tools produce and retain accessible markup and content [Priority 1] [p. 12]
Implementation: 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.

7.2 Tool: "Alt"-text registry

This tool does not have a visual window presence as far as the user is concerned. It works by saving an association between every "alt"-text label that a user writes with the name of the image, applet, image map, or image map link. Then, whenever one of these elements is inserted, the file name information of the object is checked against the registry association file. If a match is found, then the pre-written "alt" text is displayed as a default choice, allowing users to avoid the repetition of writing multiple descriptions for

the same image. The ability to store several descriptions in different languages might also be supported. In more sophisticated implementations, the tool may include a prediction algorithm that takes into account the recency of the "alt"-text, name similarity, and target similarity when searching for matches. This tool has the curb-cut advantage that the descriptions (especially the professionally written ones that come with bundled images) will allow users to search images using keyword searches, thereby simplifying the task of finding appropriate images.

8 Terms and Definitions

8.1 Markup Editing Tools and Functions

Authoring Tool
: An *Authoring Tool* is any application that is specifically designed to aid users in editing markup and presentation language documents. The editing processes covered by this definition may range from direct hand coding (with automated syntax support or other markup specific features) to WYSIWYG editors that do not present the actual underlying markup to the author for editing. This definition does *not* include text editors and word processors that also allow HTML to be hand produced.

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.

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.

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.

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.

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

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.

Multi-media Authoring Tool
Software that facilitates integration of diverse media elements into an comprehensive presentation format. May incorporate video, audio, images, animations, simulations, and other interactive components.

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.

8.2 Documents, Elements, and Attributes

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

Element
Each *element* consists of a name that identifies the type of element, optional attributes that take values, and (possibly empty) content.

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

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 HTML), and in some cases, browsers may render the value of an attribute (e.g., "alt", "title") in place of the element's content.

8.3 Accessibility Terms

Accessibility Awareness
The term accessibility awareness is used to describe an application that has been designed to maximize the ease of use of the interface and its products for people with 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.

Inaccessible Markup, Inaccessible Element, Inaccessible Attribute, Inaccessible Authoring Practice and Access Barrier
All these terms are used in the context of inaccessibility as defined by the WAI Page Author Guidelines [p. 21] .

Accessibility Solution, Accessible Authoring Practice
These terms refer to markup techniques than can be used to eliminate or reduce accessibility problems as they are defined above.

8.4 Alternative Representation of Content

Alternate Textual Representations
Certain types of content may not be accessible to all users (e.g., images), so authoring tools must ensure that *alternate textual representations* ("Alt-text") of information is available to the user. Alternate text can come from element content (e.g., the OBJECT element) or attributes (e.g., "alt" or "title").

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.).

Transcripts
A transcript is a line by line record of all dialog and action within a video or audio clip.

Video Captions
A video caption is a textual message that is stored in the text track of a video file. The video caption describes the action and dialog for the scene in which it is displayed.

8.5 Inserting and Editing

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.

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 UI manipulation.

8.6 Alert Techniques

Prompts
Prompts are simple requests for information before a markup structure has been finalized.

Interruptive Alerts
Interruptive alerts are informative messages that interrupt the edit process for the user.

Unintrusive Alerts
Unintrusive alerts are alerts such as icons, underlines, and gentle sounds that can be presented to the user without necessitating immediate action.

Alert Tools
Alert tools allow a batch detection process to address all problems at a given time.

8.7 Selection, Focus, and Events

Views

An authoring tool may offer several *views* of the same document. For instance, one view may show raw markup, a second may show a structured tree view, 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.

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 API. A view may have several selections, but only one user selection.

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., visually highlighted).

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.

9 Acknowledgments

Many thanks to the following people who have contributed through review and comment: Harvey Bingham, Judy Brewer, Carl Brown, Daniel Dardailler, Phill Jenkins, William Loughborough, Charles McCathieNevile, Charles Oppermann, and Gregg Vanderheiden.

If you have contributed to the AU guidelines and your name does not appear please contact the editors to add your name to the list.

10 References

[HTML40]

"HTML 4.0 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds. The HTML 4.0 Recommendation is available at: http://www.w3.org/TR/REC-html40/

[CSS1]

"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds. The CSS1 Recommendation 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. The CSS2 Recommendation is available at: http://www.w3.org/TR/REC-CSS2/

[WAI-PAGEAUTH]

"WAI Page Authoring Guidelines", G. Vanderheiden, W. Chisholm, and I. Jacobs, eds. These guidelines for designing accessible documents are available at: http://www.w3.org/TR/WD-WAI-PAGEAUTH/

[Page Authoring Techniques]
"Techniques for WAI Page Authoring Guidelines", G. Vanderheiden, W. Chisholm, and I. Jacobs, eds. These guidelines for designing accessible documents are available at:
http://www.w3.org/WAI/wai-gl-techniques

[CSS2-ACCESS]
"WAI Resources: CSS2 Accessibility Improvements", I. Jacobs and J. Brewer, eds. This document, which describes accessibility features in CSS2, is available at:
http://www.w3.org/WAI/References/CSS2-access

[HTML4-ACCESS]
"WAI Resources: HTML 4.0 Accessibility Improvements", I. Jacobs, J. Brewer, and D. Dardailler, eds. This document, which describes accessibility features in HTML 4.0, is available at:
http://www.w3.org/WAI/References/HTML4-access

[Access Aware Authoring Tools]
"The Three-tions of Accessibility-Aware HTML Authoring Tools", J. Richards. Available at:
http://www.utoronto.ca/atrc/rd/hm/3tions.htm