WAI Authoring Tool Guidelines

W3C Working Draft *4-Nov-1998*

This version:
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-19981104
Latest version:
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS
Previous version:
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-19981013

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

Abstract

This document provides guidelines to authoring tool manufacturers or developers. The purpose of this document is to provide guidance on accessibility of the authoring tool user interface, on implementation of accessibility improvements in Web technologies, and on making accessible Web design more "automatic" through prompting, alerts, and verification of features related to accessibility. 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 *[not yet]* a W3C Working Draft for review by W3C members and other interested parties. 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/WAI/AU/WD-WAI-AUTOOLS-19981104/wai-autools.html
A plain text file:
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-19981104/wai-autools.txt,
HTML as a gzip'ed tar file:
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-19981104/wai-autools.tgz,
HTML as a zip file (this is a '.zip' file not an '.exe'):
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-19981104/wai-autools.zip,
A PostScript file:
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-19981104/wai-autools.ps,
A PDF file:
http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-19981104/wai-autools.pdf.

In case of a discrepancy between the various formats of the specification, http://www.w3.org/WAI/AU/WD-WAI-AUTOOLS-19981104/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 6
1.1 Overview: Recommendations for Authoring Tool Developers . . .6
1.2 Guideline Priorities 6
2 Terms and Definitions 6
2.1 Markup Editing Tools and Functions 6
2.2 Documents, Elements, and Attributes 7
2.3 Accessibility Terms 8
2.4 Alternative Representation of Content 8
2.5 Inserting and Editing 9
2.6 Alert Techniques 18
2.7 Selection, Focus, and Events 9
3 Guidelines for Development of Web Authoring Tools that Support Accessibility 10
3.1 Integration of accessibility awareness 10
3.1.1 Support all accessibility features of relevant languages [Priority 1] 10
3.1.2 Ensure that authoring tools may be configured by users [Priority 1] 10
3.1.3 Emphasize accessible authoring practices [Priority 1] . . 11
3.1.4 Identify all inaccessible markup [Priority 1] 11
3.1.5 Provide mechanisms for authors to add missing alternative representations for the content of converted documents [Priority 1] . 11
3.1.6 Promote accessibility awareness in tool suites [Priority 2] . 12
3.1.7 Integrate accessibility solutions naturally [Priority 2] . . 12
3.2 Provision of accessibility information 12
3.2.1 Provide context-sensitive accessibility help to authors [Priority 1] 13
3.2.2 Provide rationales that stress Universal Design [Priority 1] . 13
3.2.3 Provide the author with positive feedback [Priority 2] . . 13
3.2.4 Package multimedia files with pre-written descriptions [Priority 2] 14
3.2.5 Promote accessibility in all Help examples [Priority 3] . . 14
3.3 Accessibility Task automation 15
3.3.1 Integrate accessibility solutions into relevant automated tools and wizards [Priority 1] 15
3.3.2 Allow the user to check for and correct accessibility problems automatically [Priority 1] 15
3.3.3 Ensure that all markup inserted through the user interface is accessible [Priority 1] 15
3.3.4 Ensure that conversion tools produce accessible markup [Priority 1] 16
3.3.5 Avoid removing or modifying any of the existing structure or descriptive content of documents [Priority 1] 16
4 Guidelines for Ensuring an Accessible Authoring Environment . . . 16
4.1 Optional Views of the Web Page to be Edited 17
4.2 Text Representation of Elements 17
4.3 Text representation of Site Map 17
4.4 Control of Cursor movement and Block Selection 17
4.5 Dialog Box and Control Palette Access 18
5 Authoring Tool Techniques 18
5.1 Alert Techniques 18

6 Sample Implementation: Alt-Text for the HTML 4.0 IMG Element . . 19
6.1 Tools: "alt"-text Registry 21
7 Acknowledgments 21
8 References 21

1 Introduction

The guidelines in this document are meant to guide authoring tool developers and vendors as they design their products. For the purposes of this document the term "authoring tool" will refer to authoring tools [p. 6] , generation tools [p. 7] and conversion tools [p. 7] . The authoring tool guidelines are designed to help developers guide users towards accessible authoring practices. This approach emphasizes the role of the user interface in informing, supporting, correcting and motivating authors during the editing process. For more detailed discussion of specific accessibility issues of particular languages, see the WAI Page Author Guidelines [p. 22] document.

1.1 Overview: Recommendations for Authoring Tool Developers

This document contains two guideline sections:

Guidelines for Development of Web Authoring Tools that Support Accessibility [p. 10]
: These guidelines outline design goals and techniques for incorporating accessible document creation awareness into Web authoring tools.

Guidelines for Ensuring an Accessible Authoring Environment [p. 16]
: These guidelines outline design goals and techniques for increasing the accessibility of authoring tools themselves for people with disabilities.

1.2 Guideline Priorities

In order to guide authoring tool developers more effectively, 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.

2 Terms and Definitions

2.1 Markup Editing Tools and Functions

Authoring Tool
: An *Authoring Tool* is any application that is specifically designed to aid users in the editing of 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 allow markup to be produced from a browsing perspective. 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 dedicated Web publishers as well as "save as HTML"(or other markup format) features in non-markup applications.

Generation Tool

A *Generation Tool* is a program or script that produces automatic markup "on the fly" following a template or set of rules.

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.

Image Editor

A graphics program that provides a variety of special features for altering bit-mapped images. The difference between image editors and paint programs is not always clear-cut, but in general image editors are specialized for modifying bit-mapped images, such as scanned photographs, whereas paint programs are specialized for creating images. In addition to offering a host of filters and image transformation algorithms, image editors also enable you to create and superimpose layers.

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 defines 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.

2.2 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.

2.3 Accessibility Terms

Accessibility Awareness

The term accessibility awareness is used to describe an application that 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 documents produced by user-authors are 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. 22] .

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.

2.4 Alternative Representation of Content

Alternate Textual Representations

Certain types of content may not be accessible to all users (e.g., images), so authoring told must ensure that *alternate textual representations* ("Alt-text") of information be 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.). D-links should be identified in the document source by giving the "rel" attribute the value "dlink".

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.

2.5 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.

2.6 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.

2.7 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 element (link, form control, element with associated scripts) in a view that will react when the user next interacts with the document.

3 Guidelines for Development of Web Authoring Tools that Support Accessibility

3.1 Integration of accessibility awareness

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

3.1.1 Support all accessibility features of relevant languages [Priority 1]

Different languages often contain different sets of accessibility features, the relative importance of which are determined by the capabilities of the languages themselves. Authoring tools must support all those accessibility features that have been defined for their language(s). This document will not list these solutions since they are stated elsewhere and are subject to clarification or change. Instead a partial list of links to language accessibility resources appears below:

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

- General: WAI Page Authoring Guidelines: Techniques [p. 22]
- HTML4: HTM4 Accessibility
- CSS2: CSS2 Accessibility
- SMIL, MathML, etc.

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

- General: WAI Page Author Guidelines [p. 22]

3.1.2 Ensure that authoring tools may be configured by users [Priority 1]

Accessibility-aware 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: 3.1.2.1] [Priority 1]
Authoring tools should be designed so that the scheduling and appearance of accessibility alerts is under some degree of user control.

[Technique: 3.1.2.2] [Priority 2]
The author should be given the freedom to customize when and how the system detects accessibility problems within the priority level constraints of the particular markup language. Specifically, the author should decide which of the [Page-Author-Priority 2] [p. 10] and [Page-Author-Priority 3] [p. 10] items in the guidelines should be flagged by the authoring tool.

[Technique: 3.1.2.3] [Priority 1]
The author should be prevented from completely disabling checking for [Page-Author-Priority 1] [p. 10] items.

[Technique: 3.1.2.4] [Priority 2]
The author should be given the option of disabling tooltips, input prediction, etc.
[Technique: 3.1.2.5] [Priority 2]
Tools should not publish inaccessible pages.

3.1.3 Emphasize accessible authoring practices [Priority 1]

All recommended page authoring accessibility features [p. 10] (and their priorities [p. 10]) must be taken into account during the design of relevant user interface components and program functionality.
Techniques:

[Technique: 3.1.3.1] [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: 3.1.3.2] [Priority 1]
Where a user input is ambiguous, authoring tools should always assume that the author intends maintain accessibility.
[Technique: 3.1.3.3] [Priority 1]
Authors must not be permitted to neglect accessibility solutions that are prioritized as [Page-Author-Priority 1] [p. 10] by a particular markup language.

3.1.4 Identify all inaccessible markup [Priority 1]

Authoring tools must be designed so that the all processes that manipulate markup text are sensitive to the existence of inaccessible markup. [p. 8]
Techniques:

[Technique: 3.1.4.1] [Priority 1]
Check existing documents when they are opened for editing.
[Technique: 3.1.4.2] [Priority 1]
Check documents during all types of editing [p. 9] (including: hand-coding, paste operations and code insertions).
[Technique: 3.1.4.3] [Priority 1]
When problems are detected the author should be notified according to a user-configurable schedule (See Alert Techniques [p. 9]).

3.1.5 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: 3.1.5.1] [Priority 1]
The author should be requested to provide missing "alt"-text for inserted images, image maps, and image map links.
[Technique: 3.1.5.2] [Priority 3]
The author should be given the option of providing a long description for any graphic element.

[Technique: 3.1.5.3] [Priority 1]

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

3.1.6 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 along with recommendations for how they may assist in facilitating accessibility.)

[Technique: 3.1.6.1] [Priority 2]

Site Management Tools: These tools already routinely perform checks for site issues, such as broken links. The ability to check for and correct [Page-Author-Priority 1] [p. 10] accessibility problems should be added to these tools. Documents missing required accessibility solutions, such as "alt"-text for HTML4.0, should not be published on the Web until all [Page-Author-Priority 1] [p. 10] accessibility problems are corrected.

[Technique: 3.1.6.2] [Priority 2]

Image, Audio and Video Editors: Image editors should be modified to encourage the writing of 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 should then be cataloged where they can be used to provide image/audio/video library search capability and default descriptive text, should the described object be added to a Web document in the future.

3.1.7 Integrate accessibility solutions naturally [Priority 2]

When a new feature is added to an established 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: 3.1.7.1] [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: 3.1.7.2] [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 document saves, closes, and pastes should not be canceled or postponed due to the existence of accessibility problems.

3.2 Provision of accessibility information

Guiding Principle: Help files, justifications and examples are critical if authoring styles and attitudes are to become more aware of Web accessibility.

3.2.1 Provide context-sensitive accessibility help to authors [Priority 1]

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

Techniques:

[Technique: 3.2.1.1] [Priority 1]

When the user activates the help system, the context for the help response should be provided by the user's point of regard within the authoring environment.

[Technique: 3.2.1.2] [Priority 1]

Implement context sensitive help for any special accessibility icons, outlining or other emphasis within the user interface (initially these may appear unfamiliar to new users).

[Technique: 3.2.1.3] [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: 3.2.1.4] [Priority 1]

The help files should include many examples as well as links to any automated correction utilities.

3.2.2 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: 3.2.2.1] [Priority 1]

The help system should provide some explanations as to the importance of utilizing accessibility features.

[Technique: 3.2.2.2] [Priority 1]

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

- hands free, eyes-free, voice-activated browsing devices such as web phone
- 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.

For more information on Universal Design, see the Trace Center.

3.2.3 Provide the author with positive feedback [Priority 2]

Achieving accessibility requires some extra effort and cooperation from the author. In order to maintain user good will and acceptance of accessible authoring practices, the user should receive positive reinforcement when accessibility objectives are satisfied.

Techniques:

[Technique: 3.2.3.1] [Priority 2]
Users should be notified of the successful implementation of accessibility objectives (ex. all images have "alt"-text in HTML4 document) as they work towards full compliance.
[Technique: 3.2.3.2] [Priority 2]
Avoid presenting long lists of remaining inaccessible items.

3.2.4 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: 3.2.4.1] [Priority 2]
Include professionally written descriptions for all multimedia files packaged with the authoring tool, in order to satisfy the following important objectives:
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: 3.2.4.2] [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: 3.2.4.3] [Priority 2]
Allow authors to make keyword searches of the description database in order to find relevant images.

3.2.5 Promote accessibility in all Help examples [Priority 3]

In addition to a help section dedicated to accessibility [p. 13] , 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: 3.2.5.1] [Priority 3]
All markup practices that require accessible solutions should appear with those solutions (ex. IMG elements should appear with "alt"-text)
[Technique: 3.2.5.2] [Priority 3]
Lower priority accessibility solutions should also be included to increase the wider acceptance of these standards.

3.3 Accessibility Task automation

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.

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

Accessibility issues often arise in complex markup tasks for which pre-existing user guidance automation exists. For example, authoring tasks such as the insertion of objects and the design of tables or frames involve relatively complex syntax that is automated to some extend by many authoring tools.

Techniques:

[Technique: 3.3.1.1] [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.

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

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

Techniques:

[Technique: 3.3.2.1] [Priority 1]

In order to ensure that documents are made accessible according to the WAI Page Author Guidelines [p. 22] every authoring tool must include an automated tool which identifies and helps correct accessibility problems.

[Technique: 3.3.2.2] [Priority 1]

The correcting tool must be designed in such a way that authors can correct accessibility problems without necessarily understanding either the underlying structure of the language or the details of markup accessibility.

3.3.3 Ensure that all markup inserted through the user interface is accessible [Priority 1]

If markup is automatically generated by an invisible process, 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: 3.3.3.1] [Priority 1]

Automated markup insertion functions [p. 7] should always make use of appropriate accessible solutions, even if this means presenting the author with extra prompts [p. 9] for necessary information or structure.

[Technique: 3.3.3.2] [Priority 1]

Automatic tools should *always* make use of algorithms that produce accessible markup [p. 8] .

3.3.4 Ensure that conversion tools produce accessible markup [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. This process is usually hidden from the user's view and often creates inaccessible documents. Note: It is likely that the creation of accessible pages will necessitate requesting information from the author during the conversion process. In addition, changes may have to be made to the appearance of the document.

Techniques:

[Technique: 3.3.4.1] [Priority 1]
Conversion utilities should generate documents that respect the WAI Page Author Guidelines [p. 22] .

3.3.5 Avoid removing or modifying any of the existing structure or descriptive content of documents [Priority 1]

When a document is created in one author tool and then opened in another, the markup is often changed to allow more efficient editing and manipulation. Unfortunately, these modifications often have the side-effect of removing markup required for accessibility.

Techniques:

[Technique: 3.3.5.1] [Priority 1]
Authoring tools should never remove or modify structure or content that is necessary for accessibility.
[Technique: 3.3.5.2] [Priority 1]
Authoring tools are allowed to change a document to make it more accessible.
[Technique: 3.3.5.3] [Priority 1]
Authoring tools must allow users to know when any changes are made.

4 Guidelines for Ensuring an Accessible Authoring Environment

Section Editors: J. Treviranus

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. Through built in adjustability (options regarding the method of displaying information and controlling the application),
2. Through compatibility 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 Guidelines: User Agent. 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.

4.1 Optional Views of the Web Page to be Edited

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

Techniques:

[Technique: 3.3.5.4] [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: 3.3.5.5] [Priority 1]

In the authoring/editing view the font size, letter and line spacing and font and background color should be independent of the final format of the document.

4.2 Text Representation of Elements

Graphically represented elements cannot be identified by third party assistive technologies that translate text to Braille, speech or large print.

Techniques:

[Technique: 3.3.5.6] [Priority 1]

Authors should be provided the option of displaying the elements in a text format.

[Technique: 3.3.5.7] [Priority 1]

The text format should include the greater than and less than brackets to help to distinguish the start or end tag from the remainder of the document.

4.3 Text representation of Site Map

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: 3.3.5.8] [Priority 1]

Authors should be given the option of displaying the site map in text form (as a structured tree file).

4.4 Control of Cursor movement and Block Selection

To edit a document, authors require accurate and efficient control of cursor movement. Many authors are unable to use a mouse or similar pointing device.

Techniques:

[Technique: 3.3.5.9] [Priority 1]

In addition to the standard cursor movement keys, authoring tools should provide keyboard commands to move the insertion cursor or begin and extend a selection, these keyboard commands should include movement or extension:

- word by word
- sentence by sentence
- element by element

- from tag to tag.

The common feature of automatically extending a selected block can be confusing to users who do not see the screen.

[Technique: 3.3.5.10] [Priority 1]
Authors should be given the option of turning this feature off.

[Technique: 3.3.5.11] [Priority 1]
To allow third party assistive technologies to find and follow the cursor, the cursor must be created using standard OS controls.

4.5 Dialog Box and Control Palette Access

Dialog boxes and palettes depend upon visual conventions and visual placement to communicate function and structure. For users who cannot see the dialog box or who cannot see the dialog box in its entirety, this structure and functionality is not communicated.

Techniques:

[Technique: 3.3.5.12] [Priority 1]
Be consistent in the placement of text labels relative to the associated text input fields, radio buttons, check boxes or other controls (i.e. always to the left, right, above or below). Also be consistent throughout the authoring tool.

[Technique: 3.3.5.13] [Priority 1]
Insure that the order of tabbing through the dialog or palette is logical.

[Technique: 3.3.5.14] [Priority 1]
Provide keyboard shortcuts to directly select controls or buttons.

[Technique: 3.3.5.15] [Priority 1]
Use structural tools available in SDKs and APIs to organize interface elements or controls.

[Technique: 3.3.5.16] [Priority 1]
Provide text labels for graphically represented controls or palette choices (this can be done through mechanisms such as tool tips).

5 Authoring Tool Techniques

5.1 Alert Techniques

The act of attracting the user's attention is an important issue for this document. The way in which a user is alerted, prompted, warned, etc. will influence their view of the tool as well as their opinion regarding the issue of accessibility itself. Within the document, emphasis is placed on integrating accessibility into the feel of the authoring tool's user interface (Integrate accessibility features naturally [p. 12]) and allowing the author the freedom to decide when and how accessibility will be addressed (Allow authors to configure the level of accessibility awareness [p. 10]).

The following is a partial list of sample alert possibilities with a short definition and a brief discussion of their advantages and disadvantages. These recommendations are meant only as a guide, since the actual implementation will depend on the look and feel of the authoring tool itself.

Prompts

Prompts are simple requests for information before a markup structure has been finalized. For example, an "alt"-text entry prompt in an image insertion dialog. 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.

Interruptive Alerts

Interruptive alerts are informative messages that interrupt the edit process for the user. For example, the 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 many 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.

Alert Tools

Alert tools allow a batch detection process to address all problems at a given time. The timing of the batch detection might be determined by the user as in the case of a word processor spell checker or by the necessity of ensuring proper syntax before a programming compile. Alert tools allow the user unfettered freedom to edit their document and a centralized detection ability within which correction tools are often integrated. However, allowing the user to leave all the accessibility problems to the end might mean that the user is presented with an extensive list that might include important organizational changes.

6 Sample Implementation: Alt-Text for the HTML 4.0 IMG Element

Section Editors: J. Richards

"Alt"-text is generally considered the most important aid to accessibility. For this reason, the "alt"-text Section of the WAI Page Author Guidelines [p. 22] should receive special emphasis both within the user interface and within the Help system.

Support all accessibility features of relevant languages [Priority 1] [p. 10]

Implementation: Every time an IMG element is created, the "alt" attribute is present.

Ensure that authoring tools may be configured by users [Priority 1] [p. 10]

Implementation: The author uses a configuration decide whether they wish to be reminded if they forget to enter "alt"-text or if they will complete the "alt"-text entry task at a later time. They are prevented from disabling "alt"-text completely.

Emphasize accessible authoring practices [Priority 1] [p. 11]

Implementation: The "alt" attribute appears below "src" in the image properties listing. Whenever the properties for an image without "alt"-text are examined, the user is reminded that "alt"-text should not be left empty.

Identify all inaccessible markup [Priority 1] [p. 11]

Implementation: If the user opens a document or pastes in markup with IMG elements lacking "alt"-text the author is prompted to add them (unless they have postponed this task).

Provide mechanisms for authors to add missing alternative representations for the content of converted documents [Priority 1] [p. 11]

Implementation: The authoring tool makes entering "alt"-text easy. Specifically, the text is added to the markup automatically and the "alt"-text Registry [p. 21] attempts to locate previous text that can be used as a default.

Promote accessibility awareness in tool suites [Priority 2] [p. 12]

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. The tool then offers to use 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.

Integrate accessibility solutions naturally [Priority 2] [p. 12]

Implementation: At no point does "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.

Provide context-sensitive accessibility help to authors [Priority 1] [p. 13]

Implementation: Whenever missing "alt"-text is flagged (anywhere in the tool suite) the same quick explanation, extended help and examples are offered.

Provide rationales that stress Universal Design [Priority 1] [p. 13]

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.

Provide the author with positive feedback [Priority 2] [p. 13]

Implementation: When the user has entered "alt"-text for all the images in a document an "alt"-text completed box will be checked as the checker begins.

Package multimedia files with pre-written descriptions [Priority 2] [p. 14]

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 sting and a longer description has been pre-written and stored in the "alt"-text registry [p. 21] .

Promote accessibility in all Help examples [Priority 3] [p. 14]

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.

Integrate accessibility solutions into relevant automated tools and wizards [Priority 1] [p. 15]

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.

Allow the user to check for and correct accessibility problems automatically [Priority 1] [p. 15]

Implementation: An accessibility checker utility exists that 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. 21] default text, if it is available.

Ensure that all markup inserted through the user interface is accessible [Priority 1] [p. 15]

Implementation: If the user drags an image from the desktop into the authoring tool, they will be prompted for "alt"-text for the IMG element (unless they have postponed this task).

Ensure that conversion tools produce accessible markup [Priority 1] [p. 16]

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.

Avoid removing or modifying any of the existing structure or descriptive content of documents [Priority 1] [p. 16]

Implementation: The authoring tool sometimes makes changes tio the HTML it works with to allow more efficient manipulation. These changes *never* result in the removal or modification of "alt"-text entries.

6.1 Tools: "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.

7 Acknowledgments

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

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.

8 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-CSS1-96121 7/.

[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.
