W3C

Authoring Tool Accessibility Guidelines

W3C Working Draft 3-May-1999

This version:
http://www.w3.org/TR/1999/WAI-AUTOOLS-19990503
(plain text, HTML gzip archive, HTML zip archive, postscript, pdf)
Latest version:
http://www.w3.org/TR/WAI-AUTOOLS
Previous version:
http://www.w3.org/TR/1999/WD-WAI-AUTOOLS-19990301

Editors:
Jutta Treviranus <jutta.treviranus@utoronto.ca>
Jan Richards <jan.richards@utoronto.ca>
Ian Jacobs <ij@w3.org>
Charles McCathieNevile <charles@w3.org>

Abstract

This document provides guidelines for Web authoring tool manufacturers and 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 ("authors") to create accessible Web content (through mechanisms such as prompts, alerts, checking and repair functions, help files and automated tools), and by ensuring that the automatic processes of the authoring tool generate accessible content. This will result in the proliferation of Web pages that can be read by a broader range of readers and in authoring tools which can be used by a broader range of users.

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

Status of this document

This is a Public Working Draft of the Authoring Tool Accessibility Guidelines. It is a draft document and may be updated, replaced or rendered obsolete by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by either W3C or members of the WAI Authoring Tool (AU) Working Group. This draft is made available for public review and comment.

The Techniques listed in this document are intended to be informative only, and although a final form of the document will make them available, they will not be present in the final "normative" version.

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

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

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

Table of Contents

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. 16] , generation tools [p. 16] , and conversion tools [p. 16] . 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 Web Content Accessibility Guidelines ([WAI-WEBCONTENT]) [p. 20] .

1.1 Guidelines, Checkpoints, and Techniques

The guidelines documents have been organized to address readers seeking abstract principles of accessible authoring tool design and readers seeking concrete solutions. The guidelines documents define three terms for different levels of abstraction:

Guideline
: A guideline is a general principle of accessible authoring tool design. A guideline addresses the question "What accessibility issues should I be aware of?"

Checkpoint
: A checkpoint is a specific way of satisfying one or more guidelines. While checkpoints describe verifiable actions that may be carried out by the authoring tool developer, implementation details are described elsewhere. A checkpoint answers the question "What must/should/may I do to make an authoring tool (and the content it produces) accessible?"

Technique
: A technique is an implementation of one or more checkpoints in a given language (e.g., HTML, XML, CSS, ...). A technique answers the question "How do I implement that in an authoring tool?"

1.2 Checkpoint priorities

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

[Priority 1]
: This checkpoint must be implemented by authoring tools, otherwise one or more groups of users with disabilities will find it impossible to access some function of the tool, or some content produced by it. Satisfying this checkpoint is a basic requirement for some individuals to be able to use the authoring tool or its output.

[Priority 2]
: This checkpoint should be implemented by authoring tools, otherwise one or more groups of users will find it difficult to use the tool or content produced by it. Satisfying this checkpoint will remove significant barriers to using the authoring tool or its output for some individuals.

[Priority 3]
: This checkpoint may be implemented by authoring tools, to make it easier for one or more groups of users to author or access content. Satisfying this checkpoint will improve the accessibility of the authoring tool or its output for some individuals.

Conformance to other specifications

These guidelines require conformance to the Web Content Accessibility Guidelines and the User Agent Accessibility Guidelines. The priority for such conformance is analagous to that required in those guidelines, as follows:

- It is a **Priority 1** requirement to be level-A conformant or to meet all the Priority 1 requirements in the relevant document
- It is a **Priority 2** requirement to be double-A conformant or to meet all the Priority 2 requirements in the relevant document
- It is a **Priority 3** requirement to be triple-A conformant or to meet all the Priority 3 requirements in the relevant document

2 Ensure that content produced by the tool is accessible

The authoring tools used to generate Web content play a critical role in determining the form and accessibility of Web markup. It is imperative that authoring tools generate content that is accessible, and that they handle the accessible authoring practices applicable to the language/format being edited. This section contains guidelines and checkpoints to ensure that the authoring tool generates accessible content.

Accessible markup differs between languages and versions, but some general principles of accessible markup are:

- Separate structure and content from presentation;
- Ensure that accessible equivalents are available for all objects that may not otherwise be accessible (e.g. text, audio descriptions for video);
- Provide consistent structure and navigation.

Authoring tools are used to automate the mechanical tasks that are part of producing Web pages. The power of this automation can enhance the accessibility of the Web if it is used to ensure that the code produced promotes accessibility, and frees the author to concentrate on the higher level problems of overall design, content, description, etc. Authoring tools can provide this support for authors in several ways:

- Producing and handling accessible content;
- Encouraging the author to adopt accessible authoring practices;
- Prompting the author for necessary information;
- Checking, validating and where necessary repairing markup;
- Providing documentation regarding accessible authoring practices;
- Integrating accessibility into the general look and feel of the tool, rather than separating it as an "optional extra".

Depending upon the design of the authoring tool, the process of creating accessible Web content can be either frustrating and onerous or easy and intuitive. It is up to the authoring tool to make accessible authoring practices an integral and efficient part of creating Web content.

Guideline 2.1: Generate standard markup

The first step towards accessibility is conformance with standards, which promotes interoperability.

Checkpoints:

2.1.1: [Priority 2]
Use applicable W3C Recommendations.

Techniques:

- When creating document types, make full use of W3C Recommendations [p. 19] (specifications which have been approved by the W3C). For example when creating mathematical content for the Web use MathML rather than another markup language.

2.1.2: [Priority 1]
Extensions to W3C Recommendations must not make content inaccessible.

Techniques:

- New document types are constantly being developed, and in many cases offer improvements to the structure and utility of Web content. In implementing a new or extended document type it is important to ensure that a tool does not remove access to information that had been inherent in the base document type.
  An HTML example of a document type that contravenes this checkpoint is a FRAMESET used without NOFRAMES - it precludes access to the underlying information, whereas NOFRAMES provides a means to access the information contained within the FRAMESET.
  The same can apply to a reduced DTD. For example, producing a DTD which did not include the "alt" attribute for IMG, or effectively working to such a DTD by not implementing a means to include the attribute, compromises the accessibility of any included IMG elements.

Guideline 2.2: Support all accessible authoring practices of W3C Recommendations

Methods for ensuring accessible markup vary with different markup languages. If markup is automatically generated, many authors will be unaware of the accessibility status of the final product unless they expend extra effort to make appropriate corrections by hand. Since many authors are unfamiliar with accessibility, these problems are likely to remain.

Checkpoints:

2.2.1: [Priority 1]
Implement all accessible authoring practices that have been defined for the markup language(s) supported by the tool.

Techniques:

- General: Checkpoints for Web Content Accessibility Guidelines [p. 20]
- Techniques for Web Content Accessibility Guidelines [p. 20]
- HTML4: HTML4 Accessibility Improvements [p. 20]
- CSS2: CSS2 Accessibility Improvements [p. 20]

2.2.2: [Priority 1]
Produce content that conforms to the W3C's Web Content Accessibility Guidelines [p. 20] .

2.2.3: [Priority 1]
Ensure that templates to be inserted in the document conform to W3C Web Content Accessibility Guidelines [p. 20] .

Techniques:

- Produce accessible representations for site maps generated by the authoring tool.

Guideline 2.3: Ensure that no accessibility content is missing

Textual equivalents, 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 producing these equivalents is probably the most time-consuming accessibility recommendation made to the author.

The authoring tool can provide various mechanisms to assist the author in generating textual equivalents while ensuring that the author can determine whether the textual equivalent accurately reflects the information conveyed by the multimedia object.

Including professionally written descriptions for all multimedia files (e.g. clip-art) packaged with the tool will:

- Save users time and effort;
- Cause a significant number of professionally written descriptions to circulate on the Web;
- Provide users with convenient models to emulate when they write their own descriptions;
- Show authors the importance of description writing.

This will lead to an increase in the average quality of descriptions used.

Checkpoints:

2.3.1: [Priority 1]
Prompt the author to provide alternative content (e.g. captions, descriptive video).

Techniques:

- Provide an author with the option of specifying alternative content, or electing to insert null alternative content. Default to an accessibility error such as no "alt" attribute for images.

2.3.2: [Priority 1]
Prompt the author for all missing structural information (e.g. TABLE scope, LABEL, FIELDSET and LEGEND for form controls in HTML).

2.3.3: [Priority 2]
Provide pre-written alternative content for all multimedia files packaged with the authoring tool.

Techniques

- Use formats which allow for accessible annotation, such as PNG.
- Provide files for longdescs, and associated text files with appropriate alt text in clip-art collections.
- Provide video description files with prepackaged video.
- Provide text caption files for prepackaged audio, or video with audio track(s).
- See also checkpoint 2.3.4

2.3.4: [Priority 3]
Provide a mechanism to manage alternative content for multimedia objects, which retains and offers for editing pre-written or previously linked alternative content.

Techniques:

- Allow authors to add objects and alternative content to a database maintained by the authoring tool. Whenever an object is used for which alternative content is provided, ask the author if they would like to add the object and the alternative content to the database. Allow multiple pieces of alternative content to be associated with a single object.
- Allow authors to make keyword searches of a description database (to simplify the task of finding relevant images, sound files, etc.). A paper describing a method to create searchable databases for video and audio files [p. 20] is available.[SEARCHABLE]
- Suggest pre-written descriptions as default text whenever one of the associated files is inserted into the author's document.

2.3.5: [Priority 1]
Do not insert automatically generated (e.g. the filename) or place-holder (e.g. "image") equivalent text, except in cases where human-authored text has been written for an object whose function is known with certainty.

Techniques

- Extensive examples that cover a number of these checkpoints are provided in the Appendix "Sample implementations" [p. 14]

Guideline 2.4: Integrate accessibility solutions into the overall "look and feel"

When a new feature is added to an existing software tool without proper integration, the result is often an obvious discontinuity. Differing color schemes, fonts, interaction styles and even application stability can be factors affecting user acceptance of the new feature.

Checkpoints:

2.4.1: [Priority 2]
Ensure that the highest-priority accessible authoring practices are the most visible and easily initiated by the author. Highlight the most accessible solutions when

presenting choices for the author.

Techniques:

- If there is more than one option for the author, and one option is more accessible than another, place the more accessible option first and make it the default. For example, when requesting alternative content for an image, offer an unchecked option for empty alternative (i.e., alt="", implying the image has no real function) with the cursor positioned in the text entry for an "alt" value, rather than offering the filename as a default suggestion, with the null "alt" value selected.

2.4.2: [Priority 1]
Make generation of accessible content a naturally integrated part of the authoring process

Techniques:

- Ensure that accessible authoring practices can be easily accessed by the author in a natural, intuitive fashion
- Include considerations for accessibility - such as the "alt" and "longdesc" attributes of the IMG element - right below the "src" attribute in a dialogue box, not buried behind an "Advanced..." button.
- Allow efficient and fast access to accessibility-related settings with as few steps as possible needed to make any changes that will generate accessible content.
- Do not set accessibility features off to the side as some optional "module"; rather, make them a part of the core operation of the authoring tool.
- The "factory settings" default configuration for the authoring tool should favor accessible solutions "out of the box", for the benefit of newer users.
- A help page that describes how to make an image map should include adding alternative content for each AREA in the MAP as part of the process. Any examples of code should give either block content with text links, or AREA elements that all have relevant ALT attribute values.
- When a user creates a frameset, suggest the main content page and a navigation bar as the content for NOFRAMES.

Guideline 2.5: Preserve existing accessible structure or content

Many applications feature the ability to convert documents from other formats (e.g., Rich Text Format) 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.

Checkpoints:

2.5.1: [Priority 1]
The tool must recognize accessibility markup for any language or format that it imports or converts.

2.5.2: [Priority 1]
Never remove markup supported by the tool that is known to promote accessibility.
2.5.3: [Priority 2]
When removing unrecognized markup, alert the author (according to a configurable schedule)

Techniques:

- Provide a summary of all automated structural changes that may affect accessibility.
- Do not change the DTD without notifying the author.

Guideline 2.6: Provide methods of checking and correcting inaccessible content

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

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.

Checkpoints:

2.6.1: [Priority 1]
Check for and alert the author of accessibility and validity problems.
2.6.2: [Priority 2]
Allow users to control both the nature and timing of accessibility alerts.

Techniques

- Allow users to choose different alert levels based on the priority of authoring accessibility recommendations.
- If interruptive warnings are used, provide a means for the author to quickly set the warning to non-obtrusive to avoid frustration.
- Include alerts for [Web-Content-Priority 1] [p. ??] checkpoints in the default configuration.
- Allow authors to control both the nature and timing of the correction process.

2.6.3: [Priority 1]
Assist authors in correcting accessibility and validity problems.

Techniques:

- Do this in a way that is consistent with the look and feel of the authoring tool.

2.6.4: [Priority 3]
Provide the author with a summary of the accessibility status on a configurable schedule.

2.6.5: [Priority 3]
Allow the author to perform element transformations. For example, to transform visually formatted elements to structure elements, or tables to lists.

Techniques

- Allow the user to define transformations for imported documents which have presentation, rather than structural, markup.
- Include pre-written transformations to rationalize multiple tables and to transform (deprecated) presentation HTML into style sheets.

Guideline 2.7: Promote accessibility in help and documentation

The issues surrounding Web accessibility are often unknown to Web authors. Help and documentation should explain accessibility problems and solutions, with examples.

Checkpoints:

2.7.1: [Priority 1]
Explain the use of accessible authoring practices supported by the authoring tool.

2.7.2: [Priority 2]
Integrate accessible authoring practices in all applicable help topics.

Techniques:

- Ensure that accessibility solutions are present in all help text descriptions of markup practices (e.g., IMG elements should appear with "alt"-text and a "longdesc" attribute wherever appropriate).
- Provide examples of all accessibility solutions in help text, including those of lower Web-Content-Priority.
- Link from help text to any automated correction utilities.
- Implement context-sensitive help for all special accessibility terms as well as tasks related to accessibility.
- Link those mechanisms used to identify accessibility problems (e.g., icons, outlining or other emphasis within the user interface) to help files.

2.7.3: [Priority 1]
Examples must not use inaccessible markup.

2.7.4: [Priority 3]
Emphasize the universal benefit of accessible design.

Techniques:

- In help text, when explaining the accessibility barriers of non-deprecated elements, emphasize appropriate solutions rather than explicitly discouraging the use of the element.
- Explain the importance of utilizing accessibility features generally and for specific instances.
- In help text, emphasize accessibility features that benefit multiple groups.

3 Ensure that the Authoring Tool is Accessible to Authors with Disabilities

Web authors have a broad range of skills and needs. Guidelines in this section address the accessibility of the authoring tools to Web authors.

Principles to consider in making the authoring tool accessible to authors with disabilities relate to three classes of functionality:

1. The authoring tool is a software program with standard user interface elements and as such should follow relevant user interface accessibility guidelines.
2. The authoring tool frequently encompasses the functionality of a user agent or browser and as such should follow the User Agent Accessibility Guidelines [p. 20] .
3. The authoring tool has unique functionality as a Web content editor.

Software can be made accessible by building in a range of options for displaying information and controlling the application, and by making the tool compatible with third party assistive technology (e.g., text to speech devices or alternative keyboards). Although implementation requirements and techniques vary from platform to platform, the following general principles should be applied:

- Use system standards.
- Support device independence.
- Enable user configurability.
- Provide appropriate programmatic interfaces.

Guideline 3.1: Follow principles of accessible design

The authoring tool is a software program with standard user interface elements and as such should follow relevant user interface accessibility guidelines.

Checkpoints:

3.1.1: [Priority 1]

Use operating system and accessibility standards and conventions for the platform(s) the tool runs on.

Techniques:

- Guidelines for specific platforms include
  - Microsoft accessibility guidelines
  - IBM accessibility guidelines
  - Apple accessibility guidelines
  - Java accessibility guidelines
- General guidelines for producing accessible software include:
  - TRACE guidelines

3.1.2: [Priority 1]

Ensure that user agent functionality offered by the tool (e.g. in a preview mode) conforms to the W3C’s User Agent Accessibility Guidelines [p. 20] .

Guideline 3.2: Ensure independence of authoring and publishing environments.

The author may need a different presentation to edit the Web content than the one they wish ultimately to be displayed. This implies display preferences that do not manifest themselves in the ultimate markup or style declarations.

Checkpoints:

3.2.1: [Priority 1]
Ensure the rendering used while authoring is independent of styles used for the published document (e.g., the font size, letter and line spacing, and text and background color, etc.).

Techniques:

- In representing the source structure of a document, mark elements with textual brackets rather than purely graphic representations. For example "</>" is regarded as a textual bracket, since it is made of character elements.

3.2.2: [Priority 1]
Allow the author to display a textual equivalent of content while editing.

Techniques:

- For a site management tool, allow the author to display a site map in text form (e.g., as a structured tree file).
- Allow the author to specify that filenames or alternative content are rendered in place of images or other multimedia content while editing.

Guideline 3.3: Provide accessible navigation

[Editors' note: The name of this guideline will be revised to reflect dealing with the structure of a document]

Authoring Web content requires editing a potentially large and complex document. In order to edit a document the author must be able to locate and select specific blocks of text, efficiently traverse the document and quickly find and mark insertion points. Authors who use screen readers, refreshable braille displays, or screen magnifiers can make limited use (if at all) of visual artifacts that communicate the structure of the document and act as sign posts when traversing the document. There are strategies that make it easier to navigate and manipulate a marked up document . A compressed view of the document allows the author to both get a good sense of the overall structure and to navigate that structure more easily.

Checkpoints:

3.3.1: [Priority 1]
Enable navigation and editing via the structure of the document.

Techniques:

- Allow the author to navigate via an "outline" or "structure" of the document being edited. This is particularly important for people who are using a slow interface such as a small braille device, or speech output, or a single switch input device. It is equivalent to the ability provided by a mouse interface to move rapidly around the document.
- To minimally satisfy this checkpoint, allow navigation from element to element.

3.3.2: [Priority 2]
Enable editing of the structure of the document.

Guideline 3.4: Ensure accessible representation of elements

Graphically represented elements cannot be identified by assistive technologies that translate text to braille, speech, or large print, unless there is appropriate information available as text. For example, some HTML authoring tools display start and end tags as graphics.

Checkpoints:

3.4.1: [Priority 1]
For all elements of a document, the properties of that element must be accessible to the author.

4 Appendix - Sample Implementations

The Sample Implementations are *not* Guidelines, they are Techniques. 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.

4.1 The A-prompt Tool

The A-prompt tool ([APROMPT] [p. 20]) is an example tool that allows for checking of many accessibility features in HTML pages, and incorporates an "alt text registry" to manage alternative content for known resources. The tool is built in such a way that the functions can be incorporated into an authoring tool.

4.2 Alt-Text for the HTML 4.0 IMG Element

[Editors' note: This section has not kept pace with the development of the guidelines. It will be updated in future drafts.]

"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 a sample implementation.

2.1 Generate standard markup [p. 6]
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 authoring practices of W3C Recommendations [p. 6]
Implementation: Due to the [Web-Content-Priority 1] [p. 20] recommendation status of "alt"-text in the Web Content Accessibility Guidelines, special attention will be devoted to prompting and guiding the user toward full "alt" coverage.

2.3 Ensure that no accessibility content is missing [p. 7]
Implementation: The authoring tool is shipped with many ready-to-use clip art and other images. For each of these images a short "alt"-text string and a longer description have been pre-written and stored in an "alt"-text registry.

2.4 Integrate accessibility solutions into the overall "look and feel" [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.

2.5 Preserve existing accessible structure or content [p. 9]
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.

2.6 Provide methods of checking and correcting inaccessible content [p. 10]
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).

2.7 Promote accessibility in help and documentation [p. 11]
Implementation: Whenever missing "alt"-text is flagged (anywhere in the tool suite) the same quick explanation, extended help, and examples are offered.

5 Terms and Definitions

[Editors' note: This section will be reviewed by the group, and is expected to be updated in future drafts]

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 and Alerts

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.

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.

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

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

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

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

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

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.

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

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

Attributes
: in XML and HTML, an element may have any number of attributes. In the following example, the attributes of the beverage element are flavour, which has the value "lots", and colour, which has the value "red": <beverage flavour="lots" colour="red">my favorite</beverage> 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.

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 Web Content Accessibility Guidelines [p. 20] .

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

Alternative Representation of Content

Alternative Textual Representations
: Certain types of content may not be accessible to all users (e.g., images), so authoring tools must ensure that *alternative textual representations* ("Alt-text") of information is available to the user. Alternative 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.

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 User Interface manipulation.

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 Application Programmatic Interface (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.

6 Acknowledgments

Many thanks to the following people who have contributed through review and comment: Jim Allan, Kynn Bartlett, Harvey Bingham, Judy Brewer, Carl Brown, Wendy Chisholm, Rob Cumming, Daniel Dardailler, Mark Day, BK Delong, Jamie Fox, Sylvain Galineau, Phill Jenkins, William Loughborough, Charles Oppermann, Dave Pawson, Bruce Roberts, Gregory Rosmaita, Jim Thatcher, Irène Vatton, Gregg Vanderheiden and Lauren Wood.

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.

7 References

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

[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

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

[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/

[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

[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/

[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

[SEARCHABLE-VIDEO]
"A Comparison of Schemas for Dublin Core-based Video Metadata Representation", J Hunter. Available at:
http://www.dstc.edu.au/RDU/staff/jane-hunter/mpeg7/contribution.htm

[WAI-USERAGENT]
"User Agent Accessibility Guidelines", J. Gunderson and I. Jacobs, eds. These guidelines for designing accessible user agents are available at:
http://www.w3.org/TR/WAI-USERAGENT

[WAI-WEBCONTENT]
"Web Content Accessibility Guidelines", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. These guidelines for designing accessible documents are available at:
http://www.w3.org/TR/WAI-WEBCONTENT

[Web-Content-Priority]
"Techniques for Web Content Accessibility Guidelines", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. These guidelines for designing accessible documents are available at:
http://www.w3.org/TR/WAI-WEBCONTENT/#priorities

[Web-Content-Techniques]
"Techniques for Web Content Accessibility Guidelines", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds. These guidelines for designing accessible documents are available at:
http://www.w3.org/TR/WAI-WEBCONTENT/wai-pageauth-tech