

**Information Technology Supporting Documentation
Commonwealth of Pennsylvania
Governor's Office of Administration/Office for Information Technology**

Document Number:	OPD-ACC001A	
Document Title:	Manual Testing Strategies and Techniques for Web Site Accessibility Validation	
Issued by:	Deputy Secretary for Information Technology	
Date Issued:	March 16, 2006	Date Revised: November 18, 2010
Domain:	Access	
Discipline:	Accessibility	
Technology Area:	Usability	
Referenced by:	ITP-ACC001	
Revision History	Description:	
Date:		
11/18/2010	ITP Refresh	

Validation Overview:

Evaluating Web resources for accessibility can take many forms in terms of scope, method and reporting procedure. A single developer may just want to check their developed Web pages or interfaces for accessibility problems during development. Others may be called upon to do a quick, preliminary review of a small Web site, or a individual Web application or sub-domain during staging or post deployment to help determine if immediate problems need to be corrected. Other, more extensive reviews and reporting may be directed at conformance or compliance testing of entire Web domains to determine if Web resources conform to adopted accessible design standards. Although similar validation methods are applied in each, the manner in which they are applied; what they are applied to; and how the findings are recorded and reported can vary considerably.

This document describes just one scheme for conformance testing that combines automatic, semi-automatic, and manual testing of Web site accessibility. The technical methods described here can be used when developing a new site, or to evaluate an existing site. Although these methods do not need to involve users with disabilities, User Acceptance Testing by users of assistive technologies plays a critical role in proofing Web resources for accessibility and is to be incorporated into a validation process. This is especially true with regard to complex application interfaces, diverse content; or when technical reviewers do not have a very high degree of confidence in validation findings. Furthermore, involving assistive technology users with disabilities early on can frequently lead to better design solutions.

User Acceptance Testing alone may be more limited in scope and cannot determine if a Web site is accessible. It is to be combined with other testing methods and conform to an established review protocol in order to be effective. Specific strategies for conducting formal User Acceptance Testing are not addressed in this document.

In addition, this document does not address corrective measures for repairing inaccessible Web sites. Although validation findings often incorporate obvious implications for appropriate corrective actions, there is no attempt here to detail specific code requirements or design solutions.

Important Skills for Reviewers:

The ability to conduct effective accessibility evaluation depends on a wide range of skills. It is possible for individuals to evaluate Web site accessibility effectively, particularly if they have good training and experience in a variety of areas, including expertise with automated and semi-automated evaluation tools

for Web accessibility testing. However, it is less likely that one individual will have all the training and experience that a team approach can bring. For this reason, a team approach is recommended for comprehensive conformance level accessibility testing.

The following skill areas all play an important role:

- Thorough Knowledge of Web Content Accessibility Guidelines – Section 508 and WCAG 1.0
- Web Mark-up Languages (such as HTML, XML, DHTML)
- Accessible Web Design and Development Techniques, including Web Application Development
- Use of Computer-based Assistive Technologies and How People with Disabilities use the Web.
- Web Accessibility Evaluation Tools and Techniques
- Accessible Document Design Methods (PDF, PowerPoint)
- Building and Testing Synchronized Captioning for Streaming Media

The Role of Visual Inspection or Human Review:

Automated validation tools can significantly reduce the time and effort required to carry out evaluations. They are essential for testing large Web sites that include hundreds or even thousands of pages of content. However, automated tools alone cannot conclusively validate conformance to all the accessible design standards. Many accessibility checks require human judgment that are to be conducted manually using a visual inspection or semi-automatic tools. In some instances, automated tools cannot interpret markup conclusively and can only provide warnings that demand further visual inspection. Automated tools are also prone to producing false or misleading results and cannot identify workarounds or alternative solutions that have been incorporated to address specific accessibility requirements. The results from evaluation tools alone are not to be used to determine conformance levels unless used by experienced evaluators who understand the capabilities and limitations of the tools and can compensate for those shortcomings by conducting further visual inspection of representative content.

Determining the Scope of the Human Review:

Although an automated validation tool like AccVerify is to be used to automatically test or crawl nearly every page of a large Web site, it is not practical for reviewers to visually inspect this much content. Instead, when automated testing is completed, reviewers are to select a *representative sampling* of pages for manual evaluation according to the following criteria:

- Include a representative sample of pages that were identified with errors or warnings during the automated testing phase.
- Include all pages on which people are more likely to enter your site ("welcome page," etc.)
- Include a variety of pages with different layouts and functionality, for example: Web pages with complex data tables, forms, maps, graphics, multimedia, frames, animations or Flash.
- Include a plan for testing a sample of dynamically generated Web content if present on the site.
- Include online applications that are essential for public transactions and business.
- Include the most visited content or online resources based on visitor data.
- Include representative content from different sub-domains or program areas.
- Include samples from locations where different layout and design principles were applied, or different development tools or developers were used.
- Include "contact us" pages and feedback forms that are essential for communication with the public.

When the conformance report is finally published, specific details about the validation methodology and scope are to be included along with a reference to the adopted standards or conformance requirements.

Pre-Assessment Preparation:

Another important reason for determining scope prior to testing is because it may be necessary to acquire secured user access to some applications or protected content. In some instances, database administrators will need to be involved in establishing temporary user accounts so that access to the necessary application interfaces can be obtained. Other options might include the arrangement of access to staging or development environments where duplicate or *mirrored* content and functionality is provided for testing. Accessibility validation is to be an integral part of the development process and is to incorporate real data whenever possible.

Several online assessment tools available through browser extensions and plug-ins (such as [AIS - Accessible Information Solutions Toolbar for Internet Explorer](#) or the [Web Developer Extension for Firefox](#)) may not be able to access and validate content behind protective firewalls. Some arrangements may be required to export these *complete pages* to servers that are accessible to the public Internet while testing is underway. Both the Internet Explorer and Firefox browsers provide the means to save a complete Web page that includes all page code and image files for that page. These pages can be reopened in any browser and tested with online tools as long as they are accessible via the public Internet.

Selecting Validation Tools:

All validation tools provide results that are to undergo interpretation by qualified developers or reviewers before definitive conclusions can be reached. Use of a *general* automated validation tool like AccVerify that is capable of evaluating a wide range of accessibility requirements across multiple domains is an invaluable resource and can serve as a starting point for establishing a baseline of irregularities. It can also help to target specific areas that will require urgent, closer examination with more *focused* tools designed to test a limited aspect of accessibility.

These more focused tools are sometimes referred to as semi-automatic tools since the tools do not necessarily identify errors, but reveal markup variations, the absence of required design elements, reading order anomalies and other design flaws and characteristics that the knowledgeable reviewer can use to make a qualified judgment about the accessibility of Web content or functions. The reviewer may need to choose the most appropriate tool and manage the tool to capture the design elements or objects necessary to make a valid assessment.

Still other assessment tools are used to mimic assistive technologies or specific user or user agent characteristics and settings. This class of *simulation* tools helps to predict how design features may impact information exchanged with diverse audiences with special needs or adaptive technologies.

More skilled reviewers and Web coders may rely predominantly on markup validation tools that focus exclusively on syntax issues in HTML, CSS or JavaScript. These automated tools report code errors and reviewers are to be able to prioritize those that have more immediate impact on accessibility and/or site performance in general.

At times, visual inspection does not rely on tools at all, but simply the reviewer's knowledge of accessible design requirements. For instance, judgments about the meaningfulness of alternative text, descriptive link text, clarity of language, or the consistency of site navigation may require little technical skill, but a basic understanding of layout and design along with how assistive technology users access the Web can be invaluable.

Part of any usability assessment is to ideally include the examination of sample pages with different Web browsers run on multiple operating systems and computer models, using different configurations, screen resolutions, monitor types and systems. All this can be difficult to coordinate and arrange, so a centralized testing resource within an enterprise can be invaluable. The W3C recommends that the following tests be part of any visual inspection scheme:

1. Turn off images, and check whether appropriate alternative text for the images is available.
2. Turn off the sound, and check whether audio content is still available through text equivalents.

3. Use browser controls to vary font-size: verify that the font size changes on the screen accordingly; and that the page is still usable at larger font sizes.
4. Test with different screen resolution, and/or by resizing the application window to less than maximum, to verify that horizontal scrolling is not required (caution: test with different browsers, or examine code for absolute sizing, to ensure that it is a content problem not a browser problem).
5. Change the display color to gray scale (or print out page in gray scale or black and white) and observe whether the color contrast is adequate.
6. Without using the mouse, use the keyboard to navigate through the links and form controls on a page (for example, using the "Tab" key), making sure that you can access all links and form controls, and that the links clearly indicate what they lead to.

Finally, in some instances, the only foolproof method for determining how a Web design will interact with assistive technology is to access the Web resource with that technology. This is especially true with regard to Web forms, complex user interfaces, large data tables, dynamically generated content and visually intensive material. There are tools and validation techniques that can be used to inspect all these requirements; however, designers and reviewers gain a higher degree of confidence when they can witness a successfully working design. That's one reason early User Acceptance Testing is so critical to the successful development of accessible information management systems.

There is no single approach for evaluating a Web site for accessibility. Many methods, tools and skills are applicable. The tools chosen can vary depending on the skill of the examiner(s), adopted compliance requirements, and the nature and size of the Web resource being examined. The validation approach selected is to be able to meet these requirements. The specific tools and visual inspection techniques described in this document represent just one group or series of methods for conducting a conformance evaluation.

Software Requirements:

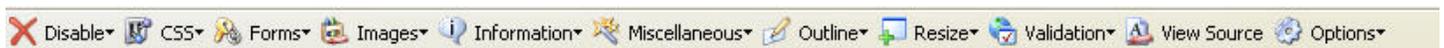
The visual inspection techniques that will be described require installation or access to the following tools:

[AIS - Accessible Information Solutions Toolbar for Internet Explorer](#)



The Web Accessibility Toolbar is a free extension for Internet Explorer (version 5 and above, Windows); it contains many features and links to online resources and tools that can help in the assessment of the accessibility of Web pages.

The [Firefox Web Browser](#) and the [Web Developer Toolbar Extension for Firefox](#)



The Web Developer extension for Firefox adds a menu and a toolbar to the Firefox browser with various Web developer tools that can also be extremely useful in evaluating accessibility. After installing the Firefox browser install the Web Developer Toolbar Extension.

After the Web Developer Toolbar is installed, open the Firefox Browser and go to the [Mozilla Update Site](#) to acquire and install these additional Firefox Extensions. Some of these extensions may not work with the latest version of the Firefox browser (version 1.5). Until updates occur, consideration is to be given to installing version 1.0.7 of Firefox. Multiple versions of Firefox can be found at this location: http://www.filehippo.com/download_firefox/?390

Multiple versions of Firefox can be installed the same PC by installing each version in a different folder location.

Colorzilla
Image Toolbar
Image Show-Hide
[Fangs Screen Reader Emulator](#)

[Checky](#)
Table Inspector

A Screen Reader may be required for foolproof testing of Web applications, online forms, and complex data tables with numerous row and column headers. The following voice output tools have all been used and endorsed by organizations engaged in accessibility validation.

The [JAWS for Windows Screen Reader](#) or,
The [HAL Screen Reader](#) or,
The [IBM Home Page Reader 3.04](#)

Full functioning demo versions of both JAWS and HAL can be installed. Demonstration time is limited to 30 minutes but can be used over again each time you restart Windows. JAWS is the predominant screen reader in the U.S. market and is used in combination with the Windows Operating System. It works best with the Internet Explorer browser, although Firefox functionality for JAWS has recently arrived. Although expensive, JAWS is the recommended screen reader of choice due to its broad popularity and user base. Most Commonwealth of Pennsylvania employees that use screen readers use JAWS. So the experience of JAWS users may be a resource at some Commonwealth locations. HAL does not offer as many features as the JAWS screen reader, but some developers report finding it easy to use, and believe it is a useful assessment tool.

The IBM Home Page Reader 3.04 is still considered a *talking browser*, but functionality for accessing the Windows Desktop and Internet Explorer was recently added. IBM offers a demo version that expires after a limited time period. However, it is far less expensive than the mainstream screen readers and an individual license can be acquired for about \$142.

VISUAL INSPECTION TECHNIQUES

IMAGES AND NON-TEXT ELEMENTS

Section 508:

[1194.22 \(a\)](#) A text equivalent for every non-text element is to be provided (e.g., via "alt," "longdesc," or in element content).

Relevant WCAG Checkpoints:

[1.1](#) Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes:* images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. (Priority 1)

OVERVIEW

When a user reads or views a Web page with a text browser, screen reader, or a browser with the image loading function turned off, images are displayed/read as "IMAGE" or as the name of the image file unless some additional steps have been taken to describe that image. The following techniques are designed to help the user know what information the image is trying to convey.

What is meant by the term, non-text element?

This term applies to more than just images and graphics. Non-text elements also include audio clips, or other programmatic objects and features that require a text equivalent so that their purpose and meaning can be conveyed to the user that is otherwise unable to access these non-text elements. Examples include buttons, images, pictures, graphical representations of text, image maps, applets, scripts, spacer images, and embedded or streaming audio or video. The proper use of text equivalents for many of these elements is also addressed under other guidelines.

TECHNIQUES FOR IMAGES

The **alt-text** attribute provides a description phrase for each image. Short for alternative text, alt-text is simply a way to provide text equivalents to non-text elements on a page. The alt-text attribute is a text string of up to 256 characters, (10 words or less is often recommended) enclosed in quotation marks. Although including alt-text is an essential part of building accessible sites, what the alt-text says is equally important. Several guidelines:

- Keep the wording simple.
- Alternative text is to describe the purpose or function of the graphic or non-text element rather than what it is or looks like. (e.g., The function of a navigation button is more important than what it looks like)
- Imagine that you are describing this image or its function to a friend over the telephone. This technique may be helpful in constructing proper alt-text.

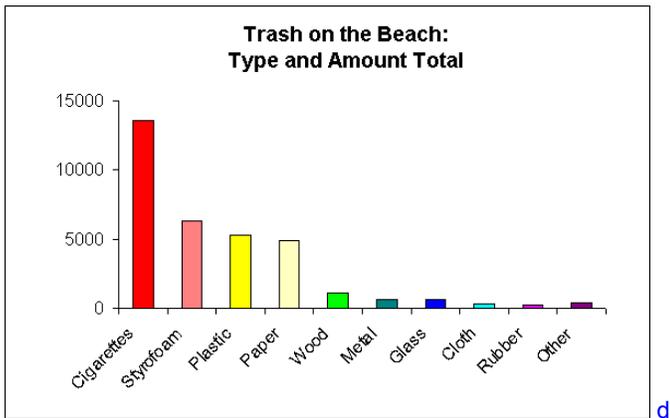
A couple of key concepts for the alt-text are:

1. Alt-text gives the individual using the Web page a quick description of the image's function or purpose.
2. Alt-text is most helpful when what is being described is stated as clearly and concisely as possible. As alt-text gets longer, it could actually become more confusing or take more time than is helpful.
3. If the content the image conveys is lengthy (more than 256 characters) consider using either the d-link or longdesc techniques.

Descriptive Link or "D-link"

There are instances where alt-text just can't convey enough information about an image such as a bar graph or chart. A D-link can be used to more clearly convey information about an image.

A D-link is used to provide longer descriptions or an in-depth description of something the author of the page wants to include more details about. The D-link, is only a small, very unobtrusive D (lower or upper case is the choice of the designer) on the graphical page. This is the only instance where the graphical version of the page will be altered.



Creation of a D-link takes several steps:

1. The letter "d" is placed next to an image.
2. The designer creates a new Web page and types the detailed description of the image. This page is saved with an appropriate name.
3. Make the "d" into a link connecting the user to the descriptive file created in step #2.

A fourth step may be to add a link to the file created in step #2 that returns the user back to the original file.

Long Description Attribute

A third option that may be considered for images is the long description HTML attribute. (This technique is also known as the longdesc attribute.) This technique is comparable in functionality to the d-link, but a bit more complicated.

Adding longdesc to HTML Code

Understand that all images in a Web page are encoded within the HTML using an IMG tag. For example:

```
<IMG src="bargraph.gif" alt="Trash Report Chart">
```

If you can find the IMG tag that refers to your image, you will be able to add the longdesc code. Note the bold code in the HTML below.

```
<IMG src="bargraph.gif" alt="Trash Report Chart" longdesc="trash_report.htm">
```

In this example, **longdesc="trash_report.htm"** is a message to screen reader technology only. The user is able to hear the alt-text and is also presented the opportunity to link to the description file. This link is transparent to individuals using visual browsers (Internet Explorer, Netscape Composer, Opera, Mozilla, etc.).

Decorative and Spacer Images

Alt-text, D-link, and longdesc are techniques to use for images that deliver content. Not all images on a Web page have meaning. The designer to take up space places some images on a page. These are referred to in the Web design community as spacer images. Designers may also develop a graphic that they use for bullets in a list. Again, it can be argued that these images don't have meaning. Decorative and spacer images still require an alt-text assignment. However, the alt-text is assigned an empty value. This is a signal to a screen reader that the image is decorative. The screen reader bypasses such an image. Empty alt-text can be assigned in this way:

Adding an Empty Alt-Text Value to HTML Code

An example of empty alt-text is below. Note alt = "". ``

The Need for Visual Inspection:

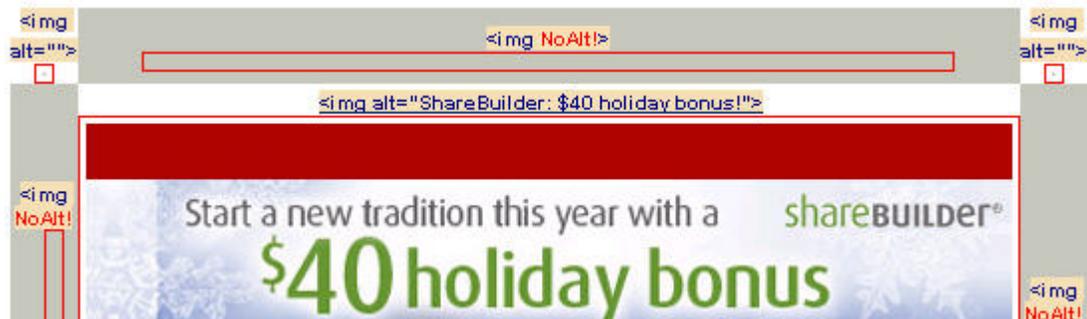
Automated validation tools are largely limited to testing for the presence of text equivalents in the markup. If configured properly, AccVerify can also test for the presence of empty Alt attributes for spacer images. Unfortunately, this presumes that developers conform to a standard naming convention for spacer images. For instance, all spacer images are named "spacer.gif" and are resized as required. However, the following files names (and others) are also frequently used as naming conventions for spacer images - clearpixel.gif, nothing.gif, transparent.gif.

Although automated validation tools can rapidly determine when Alt attributes are missing, they cannot effectively determine if text equivalents adequately describe the purpose and meaning of the non-text element. A visual inspection of all non-text elements along with an evaluation of accompanying text equivalents is to be conducted to conclusively determine if text equivalents are complete and appropriate.

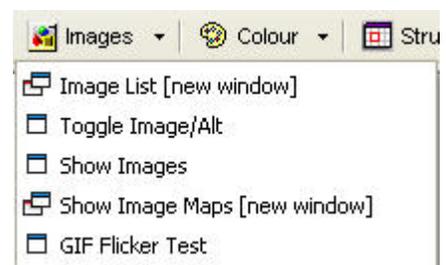
VALIDATION TECHNIQUES

The AIS Web Accessibility Toolbar provides a means of visually identifying many of the structural elements on a Web page. The AIS Toolbar reveals the properties and attributes for these components, if they have been incorporated in the markup.

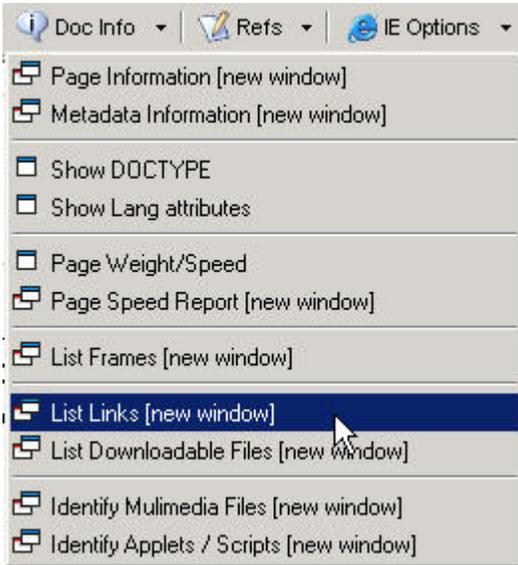
The **Show Images** function on the Images Menu of the Toolbar shows an img element next to each image on a Web page (along with the images' Alt attribute. If the img does not have an Alt attribute the text 'NoAlt!' is displayed) and puts a red border around each image displayed in the browser. Spacer images are also identified and empty alt attributes are identified as Alt="".



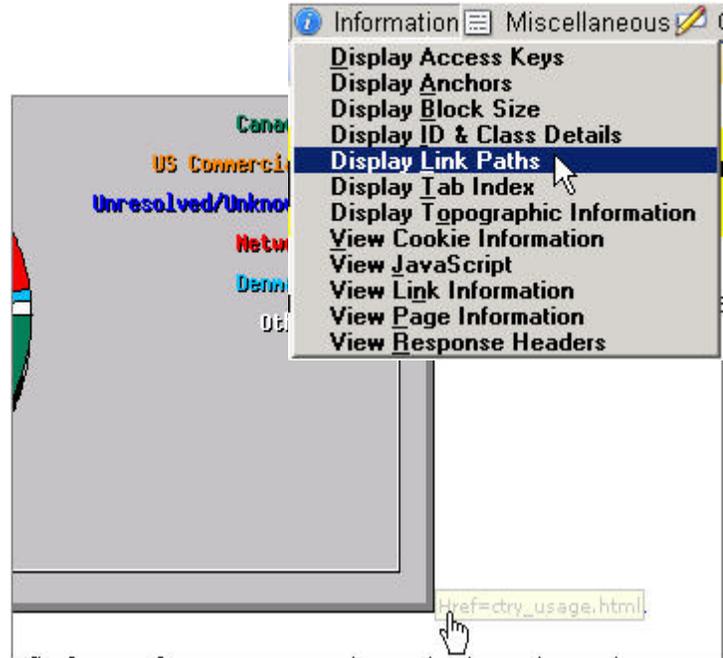
The **Toggle Image/Alt** function replaces all img elements on the current page with the content of their alt attributes within quotation marks. If an img element does not have an alt attribute the image is replaced with the text 'NoAlt!' Missing Alt text is easily identified and there is also a means of evaluating whether equivalent text is complete and appropriate within the context of the page. →



Evaluating “d” Link or the <Longdesc> attribute:



Since the “d” link is just a text link to another page that describes the image, using the visible link to access the descriptive content can easily complete the validation. The **List Links** function on the **Doc Info** Menu of the AIS TOOLBAR can all be used to list all the links on the page, the respective URLs and Title Attributes.



Since the <Longdesc> attribute is designed to be transparent to visual users, the page source code is to be examined to validate the correct implementation of <Longdesc>. An easier option may be to use the Web Developer’s Toolbar of the Firefox browser to **Display Link Paths** for every linked object on the page. When this is selected the Href attribute and document link is revealed along side each linked object. →

[Equivalent text for other types of Non-Text Elements will be addressed later in this document.]

MULTIMEDIA

Section 508:

[1194.22 \(b\)](#) Equivalent alternatives for any multi-media presentation are to be synchronized with the presentation.

[1194.22 \(m\)](#) When a Web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page is to provide a link to a plug-in or applet that complies with §1194.21(a) through (l).

Relevant WCAG Checkpoint:

[1.4](#) For any time-based multi-media presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation. (Priority 1)

OVERVIEW

Multimedia includes video, audio, and animations. Multimedia that isn't accompanied by an accessible alternative can be a major obstacle to an individual that is not able to see or hear.

Alternatives for Multimedia

Animations produced by software such as Macromedia Flash, are similar to video in that understanding the output relies upon vision, hearing, or both. It is important to provide alternatives for individuals that are blind or have partial vision and for people that are deaf or have hearing loss.

Video - Alternative Content

Alternative content for video will include both captions for spoken word and auditory descriptions of relevant action taking place on the screen. These alternatives are to be synchronized with the actions taking place on the screen. Modern video players like Windows Media Player support captioning and allow users to turn the caption on or off.

Audio - Alternative Content

The alternative for audio is a text transcription. The text transcription is to be in HTML form or plain text. Rather than listening to the content in the audio file, the user will read it as they would any other content. This text transcription needs to be placed in the near vicinity of the audio file. Ideally, the link to the audio file is accompanied by a second link that transports users to a Web page where they can read the text transcription.

Techniques for Creating Accessible Alternatives

Video - Accessible Alternatives

Video files with Audio require synchronized captioning. Software such as HiSoftware's HiCaption can assist the developer in creating the caption file. Captioning can also be completed using the National Center for Accessible Media's (NCAM) MagPie application. MagPie is free and can be [downloaded from the NCAM Web site](#). Directions for its use are also available at the NCAM site. One of the hurdles to clear in developing a caption is to generate the text. Some speakers may script their content prior to delivery. There are organizations that will generate text transcriptions for captioning from video. These services are usually not free.

Proper Notification and Access for Plug-Ins

If a plug-in is required to view the captioned video, the user needs to be able to acquire the necessary plug-in. Plug-ins include the Quicktime player, Real player, Macromedia's Flash player, etc. Brief directions for plug-in download specifications and a link to the necessary plug-in are to be provided.

VALIDATION TECHNIQUES

Locate and view all video clips and evaluate whether captioning is embedded and is synchronized with the video.

The AIS Accessibility Toolbar is capable of identifying most popular multimedia file formats. The **Identify Multimedia Files** function displays (in a new window) a list of the title, link and meta elements, and their content, found on the current page. Files are to be reviewed with the appropriate media player that is configured to display video captions. →

[AIS Toolbar Findings:](#)

Multimedia file information for 'Video Files' page

Object and Embed elements

No 'object' and/or 'embed' elements found.

Links to multimedia files

File type: .wmv

- [Canoe Tripping in Algonquin Provincial Park](#)
- [Canoe Tripping in Algonquin Provincial Park](#)
- [Backpacking in the Hammersley Fork Wild Area](#)

Locate and evaluate the equivalent text for audio files.

The **Display Link Paths** feature of the Firefox Web Developer's Toolbar can be helpful in identifying links to audio files. A link to a text transcript is to be provided in the immediate vicinity of the audio file link. The transcript or descriptive summary is to be evaluated to determine if accurately describes the audio material.

Confirm that links to necessary plug-ins are provided.

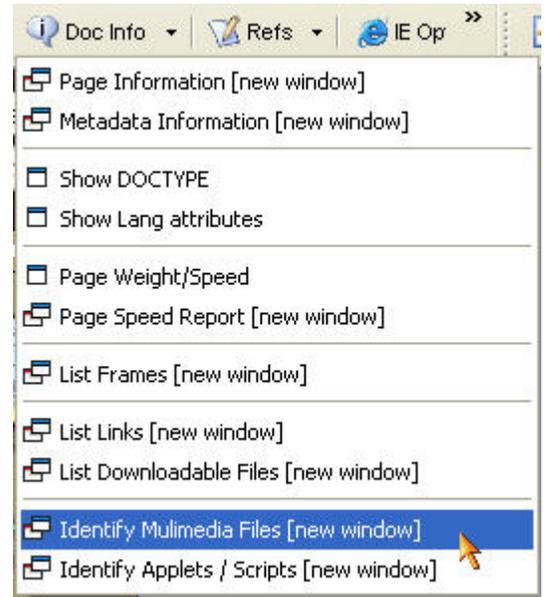
USE OF COLOR

Section 508:

[1194.22 \(c\)](#) Web pages are to be designed so that all information conveyed with color is also available without color, for example from context or markup.

Related WCAG Checkpoints:

[2.1](#) Ensure that all information conveyed with color is also available without color, for example from context or markup. (Priority 1)



[2.2](#) Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen. [Priority 2 is for images; Priority 3 is for text.]

OVERVIEW

Some people are unable to perceive color accurately or are not able to distinguish colors at all. Make sure that there is sufficient contrast between the text color and background color. The use of color to convey a meaning creates problems when those colors cannot be displayed by a browser or seen by the user. A user may be using a computer monitor with few colors, or a text-only reader. In addition, many people are colorblind and may not be able to see the colors used to convey meaning.

TECHNIQUES

- An alternative to color could be as simple as an asterisk that precedes the offering or a graphic that is accompanied by alt-text with the user alert or notification.
- If color is used to emphasize text, it may be prefaced by text such as the word – NOTE:
- If directions or a question is included that relate to images that are colored, provide appropriate alt-text for those images that describes their color with text.

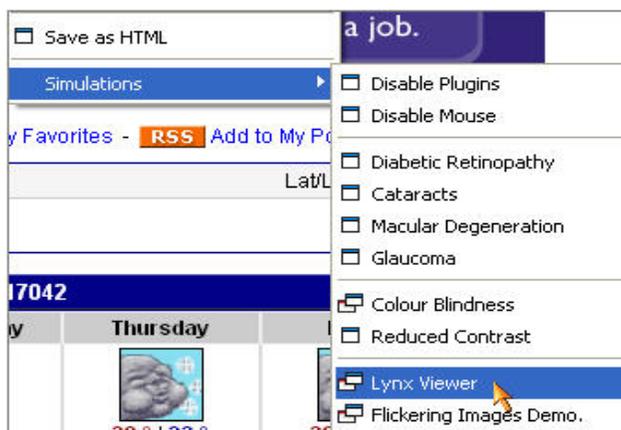
Choosing colors that contrast effectively is especially important of navigation elements and text links. Color combinations may not degrade gracefully when viewed by individuals with varied types of colorblindness. Simulation studies can be performed to provide some degree of measure with regard to how colors will contrast under various conditions. Choosing Web safe colors can help to minimize the amount of testing and simulation studies that may be necessary, since these colors generally render consistently across multiple platforms and operating systems. Establishing a well tested and consistent color scheme for a Web site can also assist in avoiding contrast issues.

VALIDATION TECHNIQUES

To determine if any page elements are color dependent, use the AIS Toolbar to view the page with colors turned off or **grey scaled**. This will also help to identify potential contrast problems as well as predict how printed pages would appear if printed on a black and white printer. Viewing pages in a text based browser like LYNX that does not support any color can also be used to verify color independence.



The AIS Accessibility Toolbar has tools that can simulate both these conditions. However, it is also recommended that chosen color schemes are evaluated using various filter and color simulation tools.



The Color Laboratory is an online tool that enables developers to construct color schemes using Web safe colors and then test those color schemes with color filters that simulate various types of colorblindness.

<http://colorlab.wickline.org/colorblind/colorlab/>

The **Colorblind Web Page Filter** provides similar capability for evaluating color on an existing Web page.

<http://colorfilter.wickline.org/>

USE OF STYLE SHEETS AND MARKUP

Section 508:

[1194.22 \(d\)](#) Documents are to be organized so they are readable without requiring an associated style sheet.

Relative WCAG Checkpoints:

[6.1](#) Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it is to be possible to read the document. (Priority 1)

[3.4](#) Use relative rather than absolute units in markup language attribute values and style sheet property values. (Priority 2)

[3.5](#) Use header elements to convey document structure and use them according to specification. (Priority 2) [4.2](#) Specify the expansion of each abbreviation or acronym in a document where it first occurs. (Priority 3)

OVERVIEW

There are in fact three sources from which style sheets can be taken when a document is rendered:

- The User Agent (often a browser) has a built-in default.
- The author can attach style sheets to a document, via links in the document or embedded within the document (HTML Page).
- However, the user can also install one or more local style sheets and direct his UA to use them for all or for certain documents.

How structural markup and style sheets benefit accessibility:

With CSS-styled pages, users can easily apply personalized formatting to Web documents. Some browsers have a feature that allows users to override author-defined style sheets with their own style sheet. A page designed using red text against a green background, for example, presents a problem for people with red-green color blindness: the contrast between text and background may not be great enough for the text to be readable. If text color is set using `` and headings are set using `` and `` for emphasis, the user-defined style sheet will have nothing to apply itself to (no paragraph or heading tags). However, if the developer controls these elements via a style sheet, users can set their browser preferences to override your settings and can apply their own style sheet to the page instead. With CSS-styled pages, users can transform Web content into a format that meets their requirements for accessibility.

Style sheets were designed to allow the author to *influence* the presentation, but *not* to control it. Users who may choose--or require--their own style sheet, can override style sheets. For these reasons, authors who depend on a certain style will find their pages inaccessible to a portion of users.

Use HTML headings to structure the Web page. It's best to format the appearance of the headings from the style sheet. Don't use headers for font effects. Screen reader users can readily identify and group headers. This helps the screen reader user to understand the organizational structure of the page.

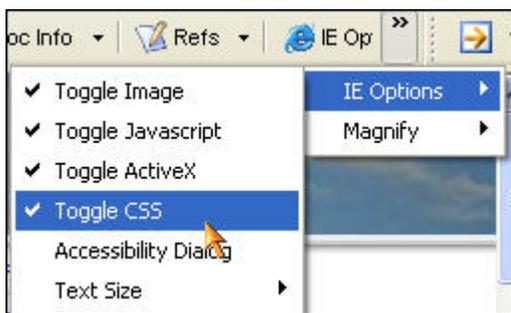
Restrain from formatting text color, size, and font in the HTML. All of these characteristics are better assigned from a style sheet. Web browsers allow users to increase or decrease the size of the font on a Web page. Style sheets can take advantage of this function or eliminate it. Style sheets provide the option for either absolute or relative font sizing.

Absolute sizing locks the font into a size that a user can't modify from their browser. Sizing units such as pixels and points are absolute. Relative sizing allows the user to adjust the text to their comfort level. With style sheet the relative sizing units are ems and %. For low vision users there is an obvious advantage to being able to scale the size of text to a level that they can read easily, even if it has an unsightly effect on the aesthetic appearance of the Web page.

Abbreviations and Acronyms are to also be expanded in the markup. These are especially troublesome for Screen Readers to voice.

Since some users may elect to turn your style sheets off or impose their own, the requirement is that your pages remain readable with styles sheets turned off. If style sheets are used to control structure and positioning, the page can become significantly disorganized when the associated style sheet is turned off. Care is to be taken to ensure that correct reading order is preserved and the page remains usable when style sheets are turned off.

VALIDATION TECHNIQUES



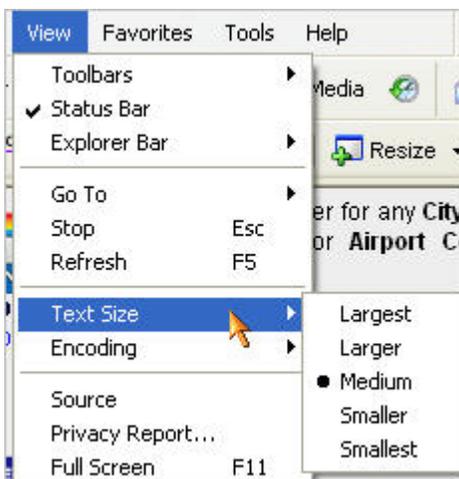
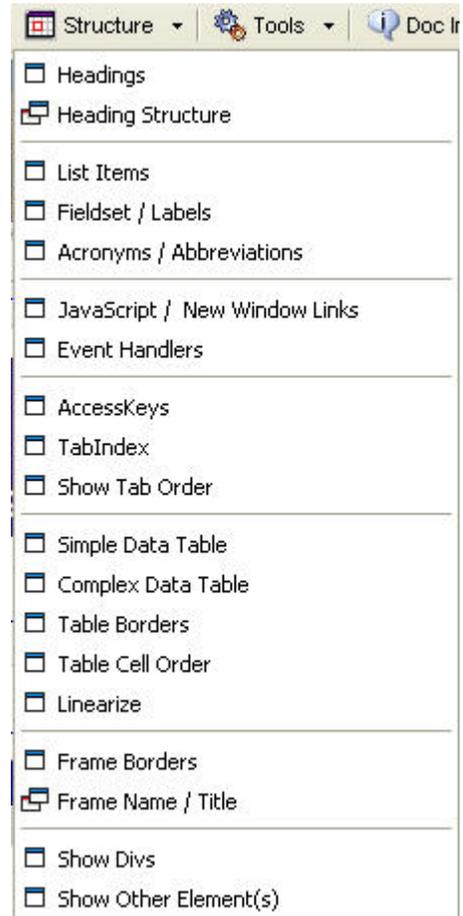
To determine how your page will function without style sheets disconnect your style sheet from the Web page by turning off support for styles in the browser. Support for style sheets can be easily restored. This can be done easily with the AIS Accessibility Toolbar. The Toggle CSS function can be used to visually examine how the content of a Web page is displayed without the presentational effects of Cascading Style Sheets. After styles are disabled, evaluate the page to ascertain that reading order is not affected and the page remains usable, despite some visible

structural differences.

Use items on the Structure Menu in the AIS Web Accessibility Toolbar to reveal if **Header Structure** is present and implemented correctly.

If they are present, identify **Acronyms and Abbreviations** in the markup using the Structure Menu of the AIS Accessibility Toolbar. →

To determine if **relative font sizing** was implemented correctly, enlarge the text size using the Web browser controls. More precise keyboard control is available in the Netscape or Firefox browser. If fonts scale up and down accordingly, relative text sizing was utilized.



IMAGE

MAPS

Section 508:

- [1194.22 \(e\)](#) Redundant text links are to be provided for each active region of a server-side image map.
- [1194.22 \(f\)](#) Client-side image maps are to be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

Relevant WCAG Checkpoints:

- [1.1](#) Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content).
- [1.2](#) Provide redundant text links for each active region of a server-side image map.
- [9.1](#) Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

OVERVIEW / TECHNIQUES

Unlike server-side image maps, the client-side image map allows an author to assign text to each image map "hot spot." This feature means that someone using a screen reader can easily identify and activate regions of the map. People that are unable to use a mouse cannot activate the active regions of a server-side image map. Therefore, redundant text links are required for server-side image maps.

Image map information for 'The Africa Guide - Map of Africa' page

Server-side image maps

Server-side image maps were **not found**.

Client-side image maps



```
<MAP name=map2>  
<AREA shape=POLY alt=Morocco coords=121,38,155,20,172,31,127,82,101,62,128,32,129,32 href="country/morocco/index.htm">  
<AREA shape=RECT coords=4,125,48,158 href="country/cverde/index.htm">No Alt!  
<AREA shape=RECT alt=gambia coords=21,173,64,185 href="country/gambia/index.htm">  
<AREA shape=RECT alt=senegal coords=18,160,74,175 href="country/senegal/index.htm">
```

VALIDATION TECHNIQUE

The **Show Image Maps** option of the AIS Accessibility Toolbar checks for the presence of client-side and server-side image maps. If none are found it is indicated by an alert message. If image maps are found, they are displayed in a new window along with the associated area elements. Redundant text links are not necessary for client-side image maps that provide equivalent text for the active regions of the image map.



DATA AND LAYOUT TABLES

Section 508:

[1194.22 \(g\)](#) Row and column headers are to be identified for data tables.

[1194.22 \(h\)](#) Markup is to be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

Relative WCAG Checkpoints:

[5.1](#) For data tables, identify row and column headers.

[5.2](#) For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells.

[5.5](#) Provide summaries for tables.

OVERVIEW

Section 508 Standards (g) and (h) permit the use of tables, but require that the tables be coded according to the rules of the markup language being used for creating tables. Large tables of data can be difficult to interpret if a person is using a non-visual means of accessing the Web. Users of screen readers can easily get "lost" inside a table because it may be impossible to associate a particular cell that a screen reader is reading with the corresponding column headings and row names.

Section 508 - Standard (g) provides guidance for simple data tables. Section 508 - Standard (h) addresses more complex data tables. Coding complex data tables is a tedious and challenging task. Fortunately, not all data tables are complex and it is sometimes possible to break Complex Tables into Simple Tables. Any accessibility solution that avoids the need to use id and headers is a good one to consider from a standpoint of development time requirements.

This document is not designed to be a tutorial about coding complex data tables. There are many online resources dedicated to that task. One that is recommended is WebAIM and their useful section devoted to complex data tables: <http://www.Webaim.org/techniques/tables/2> Another is from Jim Thatcher's 508 Tutorials: <http://www.jimthatcher.com/Webcourse9.htm>.

Use of Summary and Caption

Summary is transparent to sighted users. It is an attribute of the <table> tag. Proper use of summary provides an overview of the content in the table and possibly the layout of the table. It is useful in describing complex data tables, but is not needed for simple tables. While not a requirement of Section 508, it is a quick implementation to make that assistive technology users can benefit from hearing. Summaries for Data Tables are to clarify the relationship among cells.

Caption is a tag nested within the <table> markup. It provides a standard means of adding a title to a table. Many developers precede tables with a title within the document text without using the Caption tag.

When validating complex data tables for accessibility, two things are to be established -- is the appropriate table markup present and is it implemented correctly?

Use of Layout Tables:

Web developers also use tables to control the layout of their Web pages. This may not be the best approach, but is a common practice. Today's assistive technology devices are able to handle tables that

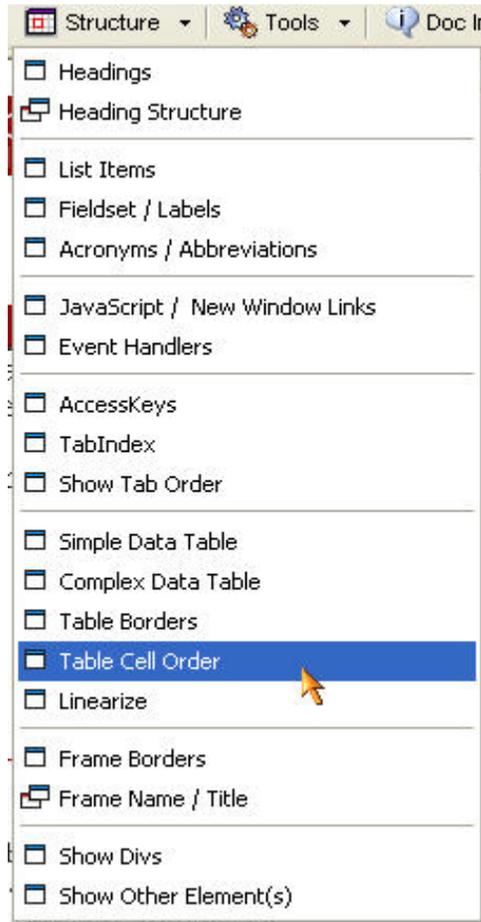
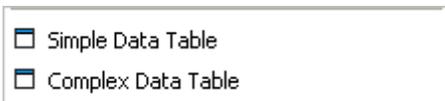


Table Validation:

Both Simple and Complex Data Tables are to be checked to determine if required table headers and markup are present. Use the AIS Web Accessibility Toolbar to reveal pertinent table markup.



Simple Data Table Shows Table, th td & caption elements on the current page along with recommended attributes for marking up simple data tables (summary, scope). A warning is given if summary ["No summary!"] or scope ["No scope!"] attributes are not present on the appropriate elements. **Complex Data Table** Shows Table, th td & caption elements on the current page along with recommended attributes for marking up Complex data tables (summary, scope, id & headers). A warning is given if summary ["No summary!"], scope ["No scope!"], id ["No id!"], headers ["No headers!"] attributes are not present on the appropriate elements.

Simple Data Table Example:

Complex Data Table Example:

[table NO summary!]

[caption] Enrollment - University of Wisconsin-Stevens Point [/caption]

[th NO scope!]	[th NO scope!]	[th NO scope!]	[th NO scope!]
	2000	2001	2002
[th NO scope!] Freshman	[td] 2200	[td] 2150	[td] 2000
[th NO scope!] Sophomore	[td] 2000	[td] 2050	[td] 2010
[th NO scope!] Junior	[td] 1992	[td] 1987	[td] 2000
[th NO scope!] Senior	[td] 1875	[td] 1990	[td] 1900
[th NO scope!] Graduate	[td] 500	[td] 475	[td] 510
[th NO scope!] Total	[td] 8566	[td] 8652	[td] 8420

[/table]

[table summary="Comparison of Enrollment between the University of Wisconsin-Stevens Point and the University of Wisconsin-Platteville. Enrollment numbers are presented vertically by the calendar years of 2000, 2001, and 2002 and horizontally as Freshman, Sophomore, Junior, Senior, Graduate, and Total. Stevens Point is presented first, Platteville second."]

[caption] Enrollment Comparison [/caption]

[th NO scope! & No id!]	[th NO scope! id="r1_c2"] 2000	[th NO scope! id="r1_c3"] 2001	[th NO scope! id="r1_c4"] 2002
[th NO scope! id="r2_c1"] University of Wisconsin-Stevens Point	[td NO headers!]	[td NO headers!]	[td NO headers!]
[th NO scope! id="r3_c1"] Freshman	[td headers="r2_c1 r1_c2 r3_c1"] 2200	[td headers="r2_c1 r1_c3 r3_c1"] 2150	[td headers="r2_c1 r1_c4 r3_c1"] 2000
[th NO scope! id="r4_c1"] Sophomore	[td headers="r2_c1 r1_c2 r4_c1"] 2000	[td headers="r2_c1 r1_c3 r4_c1"] 2050	[td headers="r2_c1 r1_c4 r4_c1"] 2010
[th NO scope! id="r5_c1"] Junior	[td headers="r2_c1 r1_c2 r5_c1"] 1992	[td headers="r2_c1 r1_c3 r5_c1"] 1987	[td headers="r2_c1 r1_c4 r5_c1"] 2000
[th NO scope!]	[td headers="r2_c1"]	[td headers="r2_c1"]	[td headers="r2_c1"]

Table Validation:

Large Complex Data Tables are to be further validated with a screen reader such as JAWS or with a tool such as the Fangs screen reader emulator. Fangs is an extension for the Firefox browser and [can be found at this Web location](#).

Reading Tables with JAWS:

JAWS automatically reformats tables so that the information is easier to read. JAWS will indicate when a table is present by indicating the number of columns and rows in the current table; it will also read the summary attribute if present. JAWS will also announce, "Table End", when it reaches the end of the table. In properly developed tables, JAWS users can also get the "cross reference" or cell association information (the associated <th> information), essential in comprehending large or complicated tables. Tables are always read in a linear fashion, from top left cell to bottom right cell, row by row, using normal JAWS reading keystrokes (Up Arrow, Down Arrow, Insert-Down Arrow, etc.) JAWS also offers these keystroke controls to read information in a table:

Read the current cell	ALT - CTRL - CENTER
Table Cell Association	CONTROL - ALT - NUMPAD 5
Read Row	R
Next row	WINDOWS KEY + DOWN ARROW
Prior row	WINDOWS KEY + UP ARROW
Read the next cell	ALT - CTRL – RIGHT ARROW
Read the prior cell	ALT - CTRL - LEFT ARROW
Read Column	C
Read the cell below the current cell	ALT - CTRL – DOWN ARROW
Read the cell above the current cell	ALT - CTRL - UP ARROW
Move to the first cell in the table	ALT - CTRL – HOME
Move to the last cell in the table	ALT - CTRL – END

USE OF FRAMES

Section 508:

[1194.22 \(i\)](#) Frames are to be titled with text that facilitates frame identification and navigation.

Relevant WCAG Checkpoints:

[12.1](#) Title each frame to facilitate frame identification and navigation. (Priority 1)

[14.3](#) Create a style of presentation that is consistent across pages. (Priority 3)

OVERVIEW

Many designers discourage the use of frames. Frames pose unique challenges to site usability; however, they can be accessible if they are coded correctly and conform to a simple and consistent presentation style for the site.

TECHNIQUE

Proper technique for building a frameset involves three steps:

- Assigning a frame name using the HTML frame name attribute
- Assigning a frame title using the HTML frame title attribute
- Title all Web pages that will become part of a frameset using the HTML <title> tag

The three steps are required as different assistive technology devices look in different places for the identification of a frame. By completing all three components, your frames will be properly labeled.

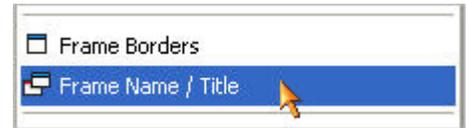
OTHER CONSIDERATIONS

Use of the <Noframes> Tag

Not all graphical Web browser versions support frames. If this is a concern, the <noframes> tag can help. The <noframes> tag is placed at the end of the frameset code. Browsers that are unable to use frames read any content that is placed within the <noframes> tag. Appropriate content would point users to a site navigation page where they can navigate the site as well as a reminder that if they are able, they may wish to upgrade their Web browser to a current version.

VALIDATION TECHNIQUE

The **Frame Name / Title** option of the AIS Accessibility Toolbar displays (in a new window) a list of framed pages along with their frame elements name and title attribute content. The title is to adequately describe the general purpose of each structure. A link to a no frames version or functionality is to be provided via <NOFRAMES> for browsers that do not support frames. In this example, no title was specified.



Frame Information

- FRAME-1
 - Name="contents"
 - Title **not specified**
 - Src="<http://www.usahistory.com/menu.htm>"
- FRAME-2
 - Name="main"
 - Title **not specified**
 - Src="<http://www.usahistory.com/main.htm>"

NOFRAMES

<NOFRAMES> <p>No frames version: Menu</p> </NOFRAMES>

To further improve usability, Framesets for a Web site are to conform to a similar style and structure across a Web site. This enables users to more readily adapt to the unique structure and navigation associated with the frame design.

CONTROL OF SCREEN FLICKER

SECTION 508:

[1194.22 \(j\)](#) Pages are to be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz

Related WCAG Checkpoint:

[7.1](#) Until user agents allow users to control flickering, avoid causing the screen to flicker. (Priority 1)

OVERVIEW

Elements that flicker between the rate of 2 Hz and 55 Hz may cause seizures in individuals that have photosensitive epilepsy. Moving content may be difficult to view for individuals that use screen magnifying techniques. Flashing Images and animations generally annoy people.

Elements that Flicker Include:

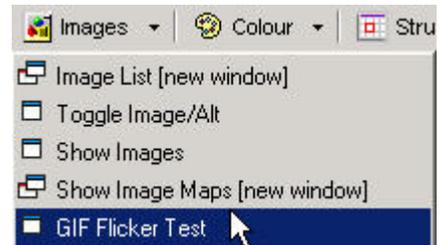
- Animated gifs
- Text that has been formatted to blink using the HTML <blink> tag
- Text that has been formatted to scroll using the HTML <marquee> tag
- Macromedia Flash animations

TECHNIQUES

There isn't an easy method for measuring the flicker rate of a flickering element. The best solution is to limit use of such elements. If an animation has some practical value other than decoration, choose one with a low flicker rate; or limit animation looping to a short period of time.

VALIDATION TECHNIQUES

These are normally just identified by inspecting the Web page. If an individual image is in question and there is a desire to retain it, the AIS Web Accessibility Toolbar provides a link to the **GIF Flicker Test**. This online service offers a rating system for identifying potentially troublesome animated images. Macromedia Flash can also produce the Flicker effect. However, it also provides enhanced controls for diminishing the problems.



TEXT ONLY PAGES

Section 508:

[1194.22 \(k\)](#) A text-only page, with equivalent information or functionality, is to be provided to make a Web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page is to be updated whenever the primary page changes.

Relevant WCAG Checkpoint:

[11.4](#) If, after best efforts, you cannot create an accessible page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent information (or functionality), and is updated as often as the inaccessible (original) page. (Priority 1)

OVERVIEW/TECHNIQUE

Designers familiar with the accessible Web design standards may argue that text only pages rarely make sense, since adequate flexibility and technique make it possible to make nearly any Web page accessible. Since the text only alternative is to convey the same content as its graphical counterpart, decisions to offer them are often based on decorative features. The text only page conveys the same content as the graphical page in a pure text form. This means that descriptions or text equivalents are to be provided for all informational components. The key concept is that the content on the graphical version and the text only version are to be the same, and are to be updated together.

VALIDATION TECHNIQUE

Assessment is based solely on a visual inspection of the content. Read both the original version and the text only version to verify they provide equivalent information and function.

SCRIPTS AND APPLETS

Section 508:

[1194.22 \(I\)](#) When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script is to be identified with functional text that can be read by assistive technology.

Relevant WCAG Checkpoint:

[6.2](#) Ensure that equivalents for dynamic content are updated when the dynamic content changes. (Priority 1)

[6.3](#) Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page. (Priority 1)

[6.4](#) For scripts and applets, ensure that event handlers are input device-independent. (Priority 2)

[6.5](#) Ensure that dynamic content is accessible or provide an alternative presentation or page. (Priority 2)

[8.1](#) Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important and not presented elsewhere, otherwise Priority 2.]

[9.3](#) For scripts, specify logical event handlers rather than device-dependent event handlers. (Priority 2)

OVERVIEW

The subject of client-side scripting and accessibility remain confusing and controversial. A client-side script is a script that is embedded within the HTML. JavaScript is the commonly used client-side script. Client-side scripts are dependent upon a Web browser's capabilities. A client-side script and a browser that doesn't process scripts equate to an inaccessible element in a Web page. Modern assistive technologies do support client-side scripting. However, they are not universally supported and still others disable the JavaScripting capabilities of their browser. In particular, assistive technology software is most likely unable to support client-side scripting, because of the other client-side functions that are to occur to re-render the display in the particular format required by the assistive technology user.

JavaScript allows developers to add increased interaction, information processing, and control in Web based content. However, JavaScript can also introduce accessibility issues.

The JavaScript may be device or mouse dependent and unusable with just a keyboard or assistive technology. The script may trigger events or *visual* effects that the assistive technology user has no awareness of and is unable to respond to. These could include things such as changes to on-screen content that are vision dependent. Most importantly, JavaScript support may be unavailable or disabled in the individual's user agent or browser.

Another relevant factor that is to also be evaluated is that not all JavaScripts are tied to the delivery of real content. Often they are used just to introduce non-essential, decorative effects or page enhancements. These may not even need to be made accessible as long they do not interfere with access to the actual content.

TECHNIQUES

Developers are to use device-independent scripts that do not depend on a mouse to function. Secondly, they are to avoid or modify scripts that produce visual effects or changes that some users may not be able to perceive or comprehend. However, after these issues are resolved, the issue of Web browsers that do not support scripting is to still be addressed. Given this problem, the developer is left with three options:

- Provide an equivalent alternative with the <NOSCRIPT> element.

- Use a server-side script instead of or in addition to a client-side script to provide equivalent functionality.
- Provide an equivalent alternative that doesn't require scripting.

The <NOSCRIPT> Tag

NOSCRIPT is an HTML tag. The content placed within a NOSCRIPT tag (<NOSCRIPT></NOSCRIPT>) is visible only to browsers or assistive technology devices that don't support scripting. The exact placement of the NOSCRIPT tag within the HTML code is open to argument. Feel good about placement immediately prior to or after the actual script element the user will encounter. Here is a very simple NOSCRIPT example:

<NOSCRIPT>

This page requires a JavaScript enabled browser to fully function.

</NOSCRIPT>

Almost anything that a designer can create as an alternative or can go in a Web page can go between <NOSCRIPT> Tags. Hyperlinks, paragraphs of text, a data table, or even an accessible audio file are all possibilities. In some cases, an alternative is as simple as a phone number or email address. Use of a human contact as an alternative is not only acceptable, but often encouraged. This requires planning to effectively accommodate individuals that may use this alternative.

Other Considerations

Given dependence on client-side scripting, generating effective NOSCRIPT content may be the real challenge. Some experts use the wording "comparable alternative." Think through the types of accommodations that may be possible. At the very least, provide users with contact information at which they may receive assistance if needed. Unfortunately, there is no easy fix that can be applied to solve all accessibility problems associated with JavaScript. The only way to ensure JavaScript accessibility is by evaluating each individual script and devising a unique solution to the accessibility problem it poses.

VALIDATION TECHNIQUES:

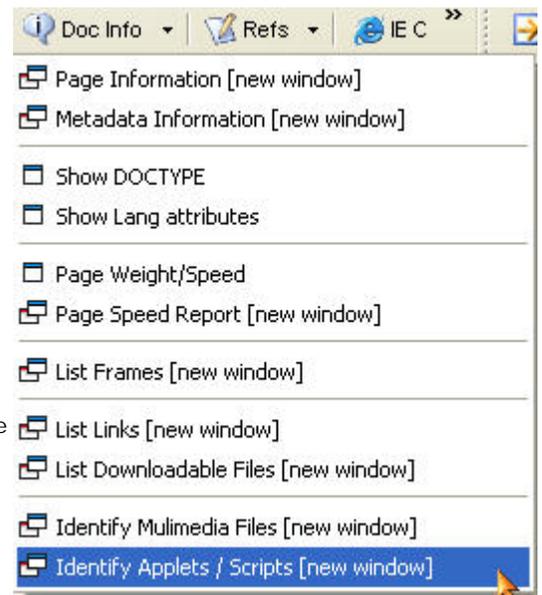
The compliance requirements for scripting can be confusing. Validation is to answer a few important questions. But before we can do that, we are to first determine if client-side scripting is present and is it being utilized to deliver essential content? Remember, some may argue that all information is essential. If the content is deemed essential, we are to then determine:

- Is the script implemented correctly so that it is device-independent and does not utilize cues or visual effects that may not be usable by some assistive technology users?
- Is a "comparable alternative" provided either by <NOSCRIPT> or another accessible solution?
- Is the alternative content or solution updated or changed when the main information is updated or changed?

Is scripting present?

The AIS Accessibility Toolbar has a tool that can help you identify scripts that are present in a Web document or page.

The Identify Applets / Scripts function displays (in a new window) any applets found and the associated code. It also lists



embedded and external scripts found and in most cases allows the viewing of external scripts linked from current page.

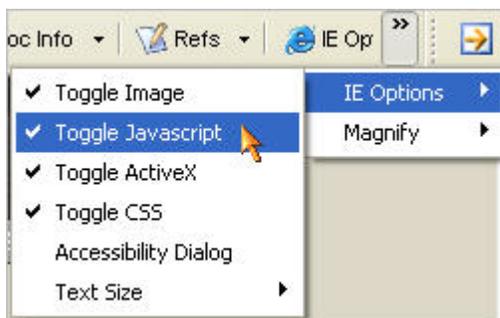
For the trained observer, viewing the script code can provide clues as to whether it is essential for the function of the content. For instance, a Web form may not be usable by non-supporting browsers, if it uses validation scripts for completing the form.

After scripting is identified, a more thorough review of the page with scripting support turned off in the browser can help further determine if the scripts or applets are essential to the function of the content.

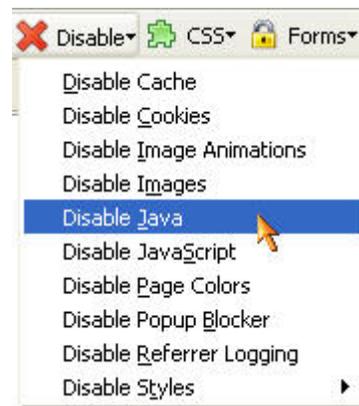
Disabling JavaScript and Java Support in the Browser

The AIS Toolbar provides the means to toggle JavaScript on and off in the Web browser. The Web Developers Toolbar in the Firefox browser provides similar on and off capabilities for JAVA.

AIS Accessibility Toolbar

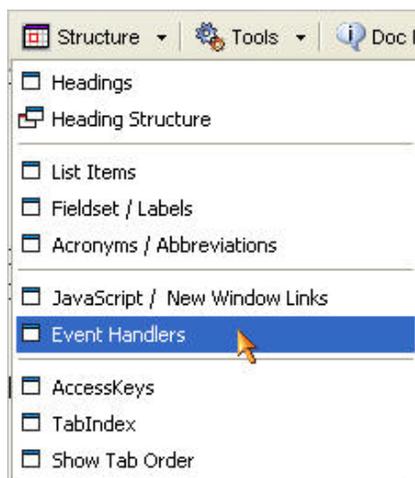


Firefox – Web Developers Toolbar



If scripts are present, are they designed to work with assistive technology?

Of course, the foolproof test is to use the scripts with the assistive technology. However, if that resource is not available, we can use the AIS Accessibility Toolbar to identify JavaScript event handlers that do not support device independence.



Displays a warning ⚠ beside an element if the element

- has an onmouseover attribute without a corresponding onfocus
- has an onclick attribute without a corresponding onkeypress
- is a select element and has an onchange attribute
- has an onfocus attribute but cannot receive focus

Displays an information bubble ⓘ beside an element if the element

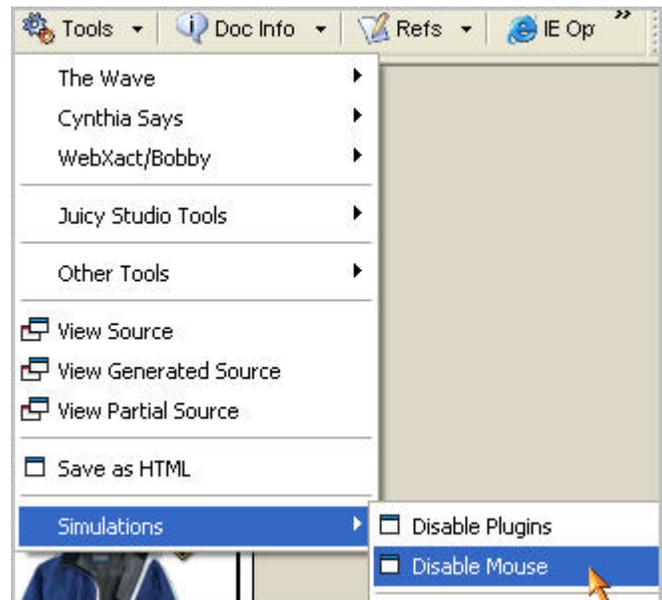
- has an onmouseover attribute with a corresponding onfocus
- has an onclick attribute with a corresponding onkeypress



The alt text of each icon contains information about why the warning/information bubble is present.

Another thorough test is to access and use all components of the document without a mouse. For assistance, the AIS Accessibility Toolbar provides a method for disabling the mouse for just the Web page functions.

This stops the user from accessing the functionality of the current page using the mouse left click/select events (currently, this feature does not stop mouseover/onmouse based events). When a user presses the left mouse button an alert box appears. **Note:** right click is not disabled.



NOTE: The evaluator is to review all JavaScript functions and also make a judgment as to whether any cues or prompts associated with the script functions are dependent on vision or a response to an audible cue that will impact accessibility. Consultation with an assistive technology user may be necessary to fully determine the impact of these types of cues.

Is a “comparable alternative” provided either by <NOSCRIPT> or another accessible solution?

JAVA Applets are to incorporate an alternative text attribute that provides descriptive information. Another alternative for providing textual descriptions is to simply include the alternative text between opening and closing <APPLET> or <OBJECT> tags. Yet another way of providing a textual description is to include it in the page in the surrounding context.

If JavaScript support is turned off in the browser, content contained between the <NOSCRIPT> tags is to be displayed in the browser when the page loads. Otherwise, <NOSCRIPT> is not present or implemented correctly, and this support for JavaScript is not provided.

Another possible option is to provide direct access via a link to a server-side solution or application that supports assistive technology requirements. However, this type of redundancy is rare for obvious reasons

Is the alternative content or solution updated or changed when the main information is updated or changed?

It is presumed that the alternative resource is accessible. This final check triggers a review of the alternative resource to determine that it is timely and provides the same information and similar functionality. When equivalents are provided for dynamic content, they are to be updated when the dynamic content changes.

FORMS

Section 508:

[1194.22 \(n\)](#) When electronic forms are designed to be completed on-line, the form is to allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

Relevant WCAG Checkpoints:

[6.3](#) Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page. (Priority 1)

[10.2](#) Until user agents support explicit associations between labels and form controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned. (Priority 2)

[5.3](#) Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version). (Priority 2)

[12.4](#) Associate labels explicitly with their controls. (Priority 2)

[9.3](#) For scripts, specify logical event handlers rather than device-dependent event handlers. (Priority 2)

[10.1](#) Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user. (Priority 2)

[5.1](#) For data tables, identify row and column headers. (Priority 1)

[5.2](#) For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells. (Priority 1)

[12.3](#) Divide large blocks of information into more manageable groups where natural and appropriate. (Priority 2)

[9.4](#) Create a logical tab order through links, form controls, and objects. (Priority 3)

[1.1](#) Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes:* images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. (Priority 1)

[2.1](#) Ensure that all information conveyed with color is also available without color, for example from context or markup. (Priority 1)

OVERVIEW

Online forms and complex Web applications can be troublesome for screen reader users and individuals that do not use a mouse as an input device. Those that rely on the keyboard or some type of specialized pointing device that emulates keyboard function may not be able to complete the form transaction, if form controls are mouse dependent. Voice output users are to be able to *explicitly* associate form labels with the appropriate entry field or form control. A frequently confusing factor in evaluating the Section 508 requirements for online forms is that many seemingly unrelated accessibility factors often contribute to making forms unusable for assistive technology users. Linear reading order and structural elements are important to helping the user navigate the form. Likewise the user is to be able to ascertain and respond to error prompts (directions and cues). This is the basis for the Section 508 requirement on forms.

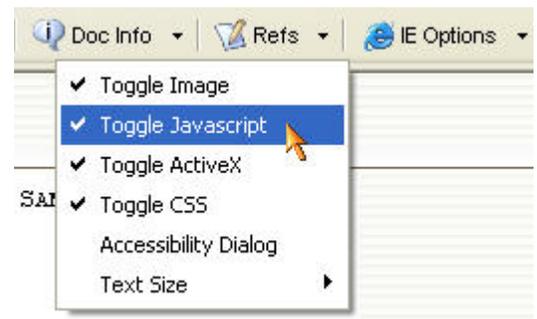
TECHNIQUE AND VALIDATION METHODS

Use of Scripting and <NOSCRIPT>

We are to first determine if a Web browser that supports scripting is required to complete the form. Although estimates vary significantly, as many as 1 in 10 users may still be using a browser that does not support JavaScript or has JavaScript disabled. Many Web forms depend on JavaScript event handlers to function. Form validation controls for prompting users to complete unfinished sections of the form are frequently dependent on client-side scripting as well.

The AIS Web Accessibility Toolbar for Internet Explorer makes it possible to easily toggle the JavaScript interpreter on and off in the browser. Turn off JavaScript and inspect the form to determine if it remains functional. Can form fields be completed and submitted? Do form validation prompts still report errors or omissions? Do confirmations still work, if applicable?

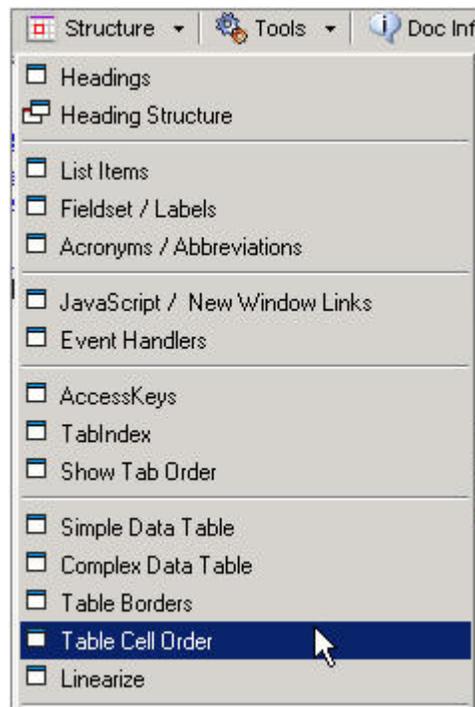
If the form requires JavaScript support, the user needs to be informed of that fact. This can be done via a visible message that precedes the form in the page reading order or through a <NOSCRIPT>message contained within the page markup. This message is displayed in browsers that don't



support scripting. The <NOSCRIPT> message is to ideally direct the user to another information resource. If appropriate <NOSCRIPT> is present in the markup, the <NOSCRIPT> message is to appear when the form is reloaded after JavaScript is toggled off. When JavaScript is found on a Web page, the failure to identify <NOSCRIPT> is also reported by AccVerify and other automated validation tools.

Positioning of Form Labels (Implicit Association)

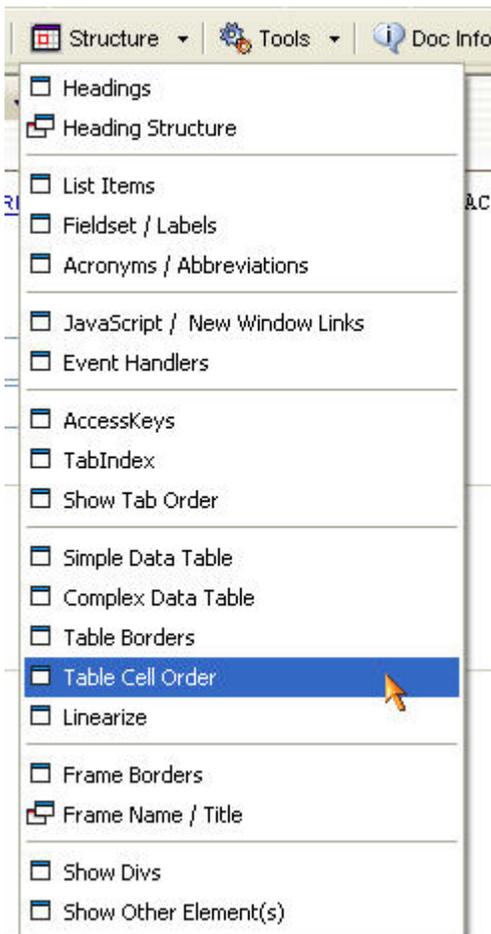
Web developers sometimes use tables to layout form elements on a page. Care is to be taken to ensure that correct reading order is preserved and the form labels are correctly positioned so they can be **implicitly** associated with the proper form control. The AIS Web Accessibility Toolbar provides controls that make it possible to reveal both *Table Cell Order* and the *Linear Order* of a Web page.



Input fields require implicit descriptions wherever possible for backward compatibility with older technologies. Text descriptions are to directly precede input fields (except for radio and checkbox types) and the label (within the markup) is to surround both description and input field, although they may appear on different lines.



The AIS Web Accessibility Toolbar can also reveal the linear order of all page content as well as the numerical reading order of Table Cells. This order dictates the positioning of descriptive text for form elements and can identify inconsistencies in the implied reading order.



114

Projects and Initiatives

[Search](#) [L&I Home](#) [Printable Version](#) [Text-Only](#) [Full-Screen](#) [Previous](#) [Next](#)

SWIF Modernization Phase 3

On September 26, 2005, the Department of Labor and Industry (DLI) issued Request for Proposals (RFP) # SWIF-2005-2 for the SWIF Modernization Phase 3. The scope of this RFP includes strategic planning regarding the PowerComp application, conducting requirements gathering, developing and implementing priority enhancements, and training and mentoring OIT staff in order to transfer responsibility for the PowerComp application from vendors to OIT staff.

RFP Number: RFP SWIF-2005-2

115 RFP Issuance	116 Monday, September 26, 2005
117 RFP Questions Due	118 Friday, September 30, 2005, 2:00 p.m. EDT
119 Revised Pre-Proposal Conference	120 Wednesday, October 19, 2005, 10:00a.m. EDT Room 100, Labor and Industry Building
121 Revised Response to Questions	122 Wednesday, October 26, 2005, 5:00 p.m. EDT
123 Revised Proposal Response	124 Thursday, November 10, 2005, 2:00p.m. EST
125 Oral Presentations	126

Associate labels explicitly with their controls

Edit boxes and other form controls are to also incorporate **explicit** descriptions. The FOR attribute in the label element is to be associated with the input ID in the markup. This is essential for voice output devices to identify what needs to be entered in an edit box. There are two ways to visually inspect whether this association has been correctly established in the markup.

The AIS Web Accessibility Toolbar is able to reveal the presence of the FOR attribute in the label tag if it is properly constructed in the markup. Use the **Fieldset / Label** control of the toolbar to reveal these attributes.

```
<label for="description">Description:
<input type="text" id="description" />
</label>
```



Sample Form: Ask a Question

`<label for="first">` First Name: `[id="first"]` `<label for="last">` Last Name:
`[id="last"]`

`<label for="email">` E-mail Address: `[id="email"]`

`<fieldset>`

`<legend>`Are you a current student?

`<label for="grad">` `[id="grad"]` Current Graduate Student

`<label for="ugrad">` `[id="ugrad"]` Current Undergraduate Student

`<label for="none">` `[id="none"]` None: not a current student

`<label for="topic">` What is the topic of your question (choose best):
`[id="topic"]` Choose topic

`<label for="question">` Tell us your question:
`[id="question"]`

In rare instances, labels can also be hidden from display in the browser by using style sheets and ***display:none***. This has the effect of not rendering the label on the screen but, because it's in the form markup, screen readers will still read the label aloud. This can make the visual inspection of labels more difficult for the accessibility reviewer. Note: This technique is only recommended for form elements where there may be insufficient space to place descriptive text on the form, but the screen reader still needs a label in the markup to make the correct label association with the edit box or form control. Under normal circumstances, labels are to be visible. However, this particular method may hide them from view. This technique is sometimes used in conjunction with telephone number labels – Area Code, Prefix, and Number. The on screen text may not correspond to the labels in the markup. On the screen it might look something like this:

Telephone Number * () -

Proper label usage can also be verified by using the mouse. Click on the text description associated with the form element. If the markup is optimally constructed the cursor will be placed in the edit box when the "explicit" label is clicked. If not, the markup may be incorrectly constructed and further validation may be required. However, the label can still meet accessibility requirements if it is a simple form and the text label on the page can be "implicitly" associated with the form element.

[HTML](#)

Sample Form: Ask a Question

First Name:

E-mail Address:

Device-Independence

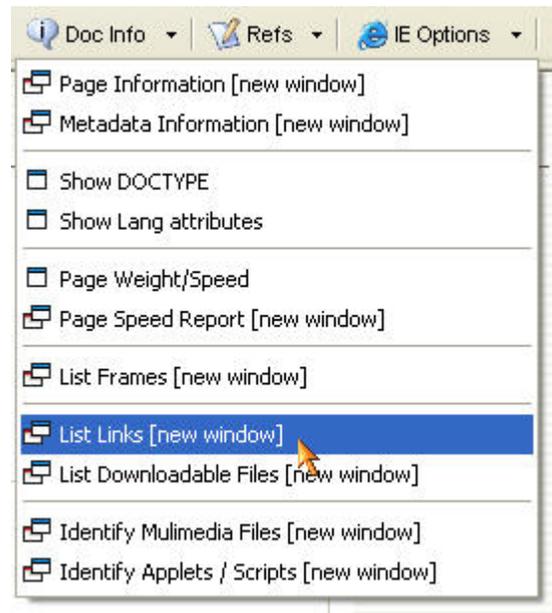
Forms are to be device-independent and not dependent on a mouse. AccVerify and other automated validation tools are able to effectively identify device-dependent event handlers in the markup and provide

appropriate warnings. (Note: A method for evaluating device-dependent event handlers with the AIS Accessibility Toolbar was addressed previously in this document under Scripts and Applets.) However, a manual approach for detecting issues with Web forms is to simply complete the form transaction by using just the keyboard. The user is to be able to tab logically through all the form controls, complete the form, submit the data and respond effectively to validation error prompts with just the keyboard. If this is not possible, device-dependent event handlers, or poorly associated form labels, are likely the problem.

Pop-ups and Spawned Windows

Spawned windows or new browser sessions are often necessary when working in online forms. Additional help and other information are sometimes offered in a new window to prevent the user from losing a secured session or entered information on a partially completed form. However, under less demanding circumstances, new windows are to be avoided unless absolutely necessary. When an activated link opens a new window, the non-sighted user is to be alerted or informed that the link opens a new window. This can be accomplished by adding a title attribute to the link in the markup. The title is to be something like: "Link Opens in New Window."

Title attributes for all links can be revealed using the List Links function of the AIS Web Accessibility Toolbar. This can be used to determine if the Title attributes applied to the new window link provide the appropriate warning.



AIS Web Accessibility Toolbar – Links List Screen:

Links List

[close window]

Title:Forms: Sample Accessible Form

Source:<http://www.wac.ohio-state.edu/tutorials/forms/examples/>

Links:31

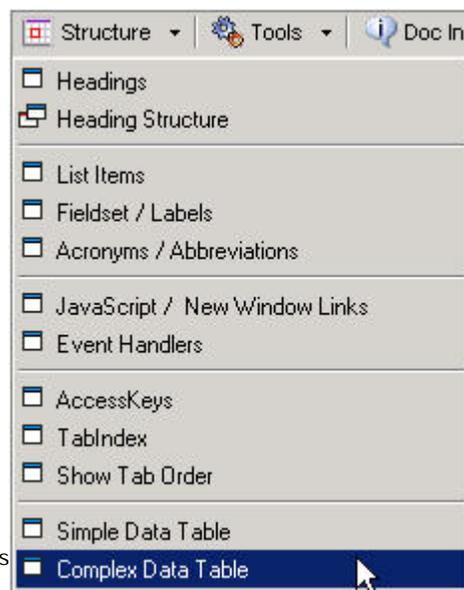
Link Content	URL	Title
1.	http://www.wac.ohio-state.edu/tutorials/forms/examples/#content	
2. The Ohio State University	http://www.osu.edu/	The Ohio State University
3. Help	http://www.osu.edu/help.php	Help
4. Campus map	http://www.osu.edu/map/	Campus Map
5. Find people	http://www.osu.edu/findpeople.php	Find people
6. Search OSU	http://www.osu.edu/search.php	Search OSU site

Using Forms with Table Rows and Columns:

Form elements within complex tables can capture data and sometimes are laid out in a grid or tabular fashion. When this occurs the rigid markup requirements that apply to complex data tables also apply to the various row and column headers for the grid. Please ensure as a minimum:

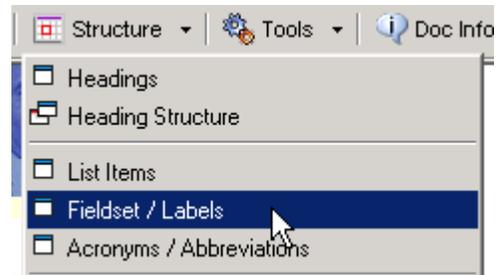
1. A table summary is stated. Outlining the purpose of the table contents.
2. Use a caption to state the purpose of the table contents.
3. Define column and or row headers along with their scope.
4. Define, where required, colgroups and their scope.
5. Use <thead> and <tbody> to separate content from headers where applicable.
6. Consider table linearization carefully before separating a label from its associated input.

The existence of these data tables and the required markup for row and column headers can be identified in the source code by using the AIS Web Accessibility Toolbar. [**Note:** This was addressed more fully in the previous section regarding Data and Layout Tables.]



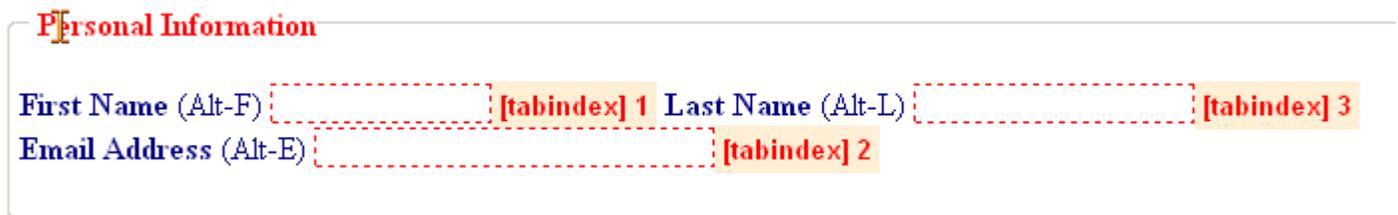
Use of <Legend> and <Fieldset>:

Large blocks of information collected by a form can be organized into more manageable groups according to data type and other characteristics. The Legend, Fieldset and Label tags are sufficient to layout a form in most instances. AIS Web Accessibility Toolbar can also be used to identify these structural components.



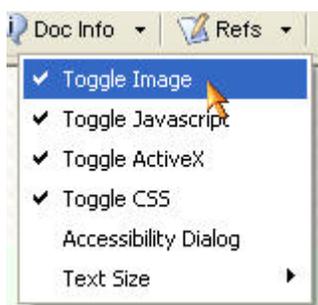
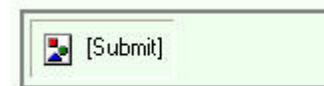
Logical Tabbing Order

Form elements are to follow a logical tabbing order. This can be tediously tested by manually tabbing through the form using the Tab Key. However, the AIS Web Accessibility Checker is able to reveal the numerical Tabbing order using the **Show Tab Order** function in the Structure Menu. If the author has utilized tab indexing, the <tabindex> order will also be noted on the screen. Tab Indexing is typically used to circumvent layout problems and is seldom necessary when the reading and tabbing order remain logical and have been verified.



Using Graphics for Form Buttons

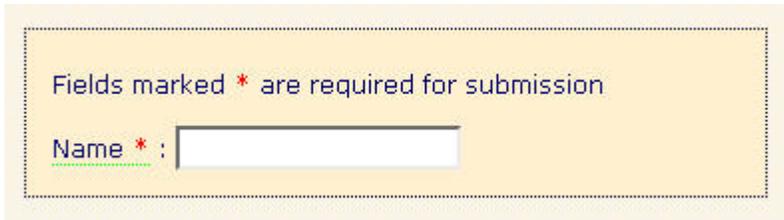
Submit and Reset Buttons are to be represented by equivalent text if graphics or images are used in place of a standard form control. Required alt-text for all images can be revealed when images are toggled off with the AIS Web Accessibility Toolbar. This equivalent text is essential in order for non-sighted users to identify the form control.



Do not rely on color alone:

All information conveyed with color is to also be provided without color. Do not rely on color to provide information about forms. The term "required" is to be stated in the text description. If a mark such as * is used,

then its meaning is to be announced prior to the first occurrence and is to *precede* each applicable form field or element. This is to be verified through visual inspection.



When in doubt, test with assistive technology.

Finally, the most foolproof method of form validation involves successful completion of the form by using a screen reader. First, read the form by tabbing from one input field to another without trying to fill out the form. Later test the form by entering the screen reader's "forms mode" and attempting to fill out the form and submit the information. Deliberately create an error to determine if your response to error prompts is usable with the screen reader.

Here are some important key strokes related to JAWS and forms:

Enter Forms Mode	ENTER or NUM PAD SLASH
Move to First Form Field	CTRL + INSERT + HOME
Move to Next Form Field	F
Exit Forms Mode	NUM PAD PLUS

Another alternative to full-fledged screen reader testing is the Fangs Screen Reader Emulator extension that is part of the Developer Toolbar for the Firefox browser. It renders a text version of a Web page or a Web form similar to screen reader output. Similar to JAWS, it can also provide a list of all links on a page, or all the Headers on a page. It works well for short forms and static Web content.

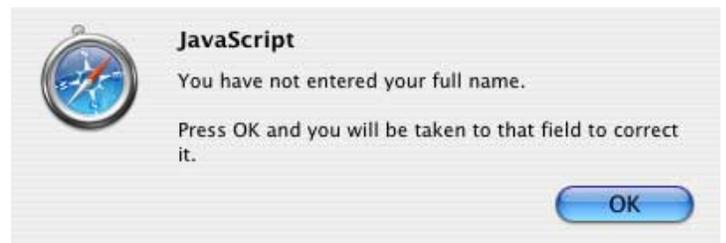


A Screen Shot From Fangs:

Page has three headings and thirty-one links Forms colon Sample Accessible Form dash Mozilla Firefox This page link Graphic skip navigation **Heading level one** Link The Ohio State University List of four items one Link Help two Link Campus map three Link Find people four Link Search OSU List end Table with two columns and one row Link Graphic Web Accessibility Center home page. List of two items bullet **Heading level one** Web Accessibility Center bullet Graphic search W.A.C. site Edit alt plus 3 List end List of five items bullet Link About WAC bullet Link Resources bullet Link Standards bullet Link Tutorials bullet Link Workshops List end List of five items bullet Link Accessible Design bullet Link Carmen bullet Link Flash bullet Link Forms bullet Link Javascript List end List of four items bullet Link Macromedia bullet Link PDF bullet Link WebAIM bullet Link WebCT List end Table end Table with one column and one row Link Home vertical bar Link tutorials vertical bar Link forms vertical bar Link examples vertical bar Forms colon Sample Accessible Form **Heading level one** Sample Form colon Ask a Question First Name colon Edit Last Name colon Edit E dash mail Address colon Edit Are you a current student? Radio button Not checked Current Graduate Student Radio button Not checked Current Undergraduate Student Radio button Checked None colon not a current student What is the topic of your question left paren choose best right paren colon Combo box Choose topic Tell us your question colon Edit Would you like to join any of the department listservs? left paren check any that interest you right paren Checkbox Not checked Department Listserv Checkbox Not checked Graduate Student Listserv Checkbox Not checked Undergraduate Listserv Checkbox Not checked eighteen th Century Literature Listserv Submit Your Question button Table end OSU Web Accessibility Center left paren WAC right paren one thousand seven hundred sixty Neil Ave one hundred fifty Pomerene Hall Columbus, Ohio forty-three thousand two hundred ten Phone colon left paren six hundred fourteen right paren two hundred ninety-two dash one thousand seven hundred sixty Fax colon left paren six hundred fourteen right paren two hundred ninety-two dash four thousand one hundred ninety E dash mail colon Link webaccess at osu.edu For questions or problems with this site, including incompatibility with assistive technology, Link email the WAC Webmaster . Table caption colon Our Partners Summary colon Links to our partner programs. Table with four columns and one row Link Graphic ADA Coordinator's Office logo ADA Coordinator's Office Link Graphic College of Education logo OSU College of Education Link Graphic Office of Disability Services logo Office for Disability Services Link Graphic T.E.L.R. logo T.E.L.R. Table end

Additional Considerations:

Often, when JavaScript form validation is used, all the errors are collected and displayed back to user at the same time like the example on the left, below.



A screen reader user is to listen to all the errors read to them at once. A better method might be to alert the user of the first error encountered (e.g., the one on the right) and take the user back to that field that needs to be corrected, and so on until all are covered. This can be accomplished with JavaScript by setting the focus back to the offending field. Of course, this could lead to a lot of going back and forth; but depending on the complexity of the form it may be the best solution for screen reader users.

A Timed Response requirement can also impact a user's ability to successfully complete a form. This is address in the next section.

TIMED RESPONSES

Section 508:

[1194.22 \(p\)](#) When a timed response is required, the user is to be alerted and given sufficient time to indicate more time is required.

Relevant WCAG Checkpoints:

Not Addressed in WCAG

OVERVIEW

This is a Section 508 requirement that is not addressed directly in the WCAG Guidelines. Web pages can be designed with scripts so that the page disappears or "expires" if a response is not received within a specified amount of time. Sometimes, this technique is used for security reasons or to reduce the demands on the computer serving the Web pages. Someone's disability can have a direct impact on the rate with which a person can read, move around, or fill in a Web form. Some forms may even "time out" automatically and also delete whatever data has been entered. The result is that someone with a disability who is slow to enter data cannot complete the form or a transaction.

TECHNIQUE

Possible accommodations for a timed response:

- Precede all timed response mechanisms with an alert that a timed response is in use.
- Provide an accessible counter that notifies a user how much time remains.
- Unless there is an over-riding security concern, alert via a prompt and give the user sufficient time to request additional time if needed.

OTHER CONSIDERATIONS

Auto Forwards or Redirects create a similar time response problem.

Users may have difficulty with this method of redirect, because:

1. Screen readers may not have enough time to read the entire redirect message before the new location loads. Users may become confused or may be left unaware that the URL has changed and will continue to use the old address.
2. Redirects break the browser HISTORY and make it difficult to use the BACK button.

Because of the inherent usability problems with automatic redirects, the W3C recommends using a static page that requires the user to engage a link in order to go to the new page location. *This is currently a Commonwealth of Pennsylvania standard.*

VALIDATION TECHNIQUES

This is purely a visual inspection. If a timed response is required, ascertain if an appropriate accommodation is provided.

NAVIGATION AND HYPERLINKS

Section 508:

[1194.22 \(o\)](#) A method is to be provided that permits users to skip repetitive navigation links.

Relevant WCAG Checkpoints:

[13.6](#) Group related links, identify the group (for user agents), and, until user agents do so, provide a way to bypass the group. (Priority 3)

[13.1](#) Clearly identify the target of each link. (Priority 2)

[10.1](#) Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user. (Priority 2)

[13.4](#) Use navigation mechanisms in a consistent manner. (Priority 2)

[3.1](#) When an appropriate markup language exists, use markup rather than images to convey information. (Priority 2)

[13.2](#) Provide metadata to add semantic information to pages and sites. (Priority 2)

OVERVIEW

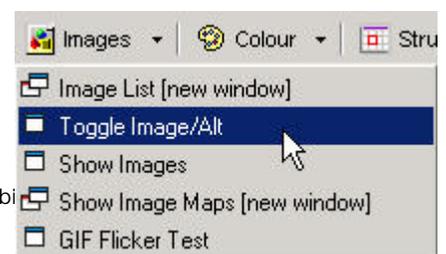
Clean, consistent navigation is important for any Web user and Web site. Users that cannot use a mouse are appreciative of any keyboard shortcut, structure or design scheme that facilitates navigation or avoids confusion. Less is usually more. The Federal Access Board (Section 508) and the W3C (WCAG) have developed several guidelines that address navigation. Some will be reviewed here.

Skip Navigation:

This technique makes finding the main content of a Web page easier for users of assistive technology. A method to facilitate the easy tracking of page content that provides users of assistive technology the option to skip repetitive navigation links. Web developers routinely place a host of routine navigational links at a standard location – often across the top, bottom, or side of a page. If a non-disabled user returns to a Web page and knows that he or she wants to view the contents of that particular page instead of selecting a navigation link to go to another page, he or she may simply look past the links and begin reading wherever the desired text is located. For those who use screen readers or other types of assistive technologies, however, it can be a tedious and time-consuming chore to wait for the assistive technology to work through and announce each of the standard navigational links before getting to the intended location. In order to alleviate this problem, the section 508 rule requires that when repetitive navigational links are used, there is to be a mechanism for users to skip repetitive navigational links. This link precedes the main navigation structure and typically links to an anchor that immediately precedes the main content area. For a detailed overview of this technique, [visit WebAIM](http://www.Webaim.org/techniques/skipnav/) at: <http://www.Webaim.org/techniques/skipnav/>

Validation Technique for Skip Navigation

Unless Skip Navigation is represented by a visible text link (in which case its presence and location is easy to verify), it will always be represented by Alt-text. The AIS Accessibility Toolbar can reveal the associated Alt-text for the Skip Navigation. Use the Toggle Image/Alt item on the AIS Images Menu to reveal the Alternative Text for the



page. The "Skip to Main Content" link is to be located near the top of the page or immediately preceding the repetitive navigation. The Alt-text may vary. It could be "Skip Navigation," "Skip Page Navigation," or something similar. The link is to also be tested to determine that it actually goes to the main content.

Clearly Identify the Target of Each Link:

When creating links to relevant pages or Web sites, use text that **makes sense when read out of context**. For example, avoid "click here."

When scanning a page, sometimes a user will just visually scan the links, to find out what link will take him/her to the desired destination.

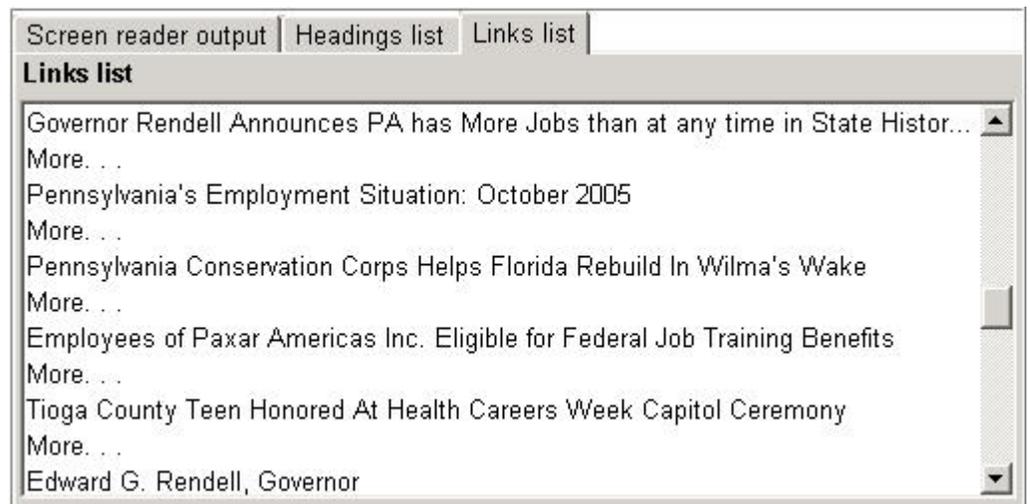
In the same way, the visually disabled use the screen readers to jump between hyperlinks. If "click here" is the link's description, then the screen reader will only express "click here." If all the links have the same kind of non-intuitive description, then a list of "click here", "click here", "click here" will be of no help to those using a screen reader.

When creating hyperlinks make them meaningful. In addition to clear link text, if necessary, content developers may further distinguish the target of a link with an informative link title (e.g., in HTML, the "title" attribute). This provides another mechanism to include more information about a link target and is especially helpful when space for link text is limited.

Validation Technique for Reviewing Link Clarity

It is easy to review links out of context with a screen reader like JAWS that enables the user to list all links on a page (INSERT + F7).

However, the [FANGS Screen Reader Emulator](#) for the Firefox browser can also list all links on a page as well as display the associated <Title> attributes for each link.



Avoid Spawned Windows and Pop-Ups or Warn the User:

Suffice it to say that pop-ups create accessibility problems. However, there are instances where they do serve a real purpose in conjunction with online transactions. For a detail review of the problems associated with Pop-Ups as well as possible solutions, review this article at Accessify.COM

(<http://accessify.com/features/tutorials/the-perfect-popup/>). If Pop-Ups are unavoidable, these design standards are to be noted for accessibility:

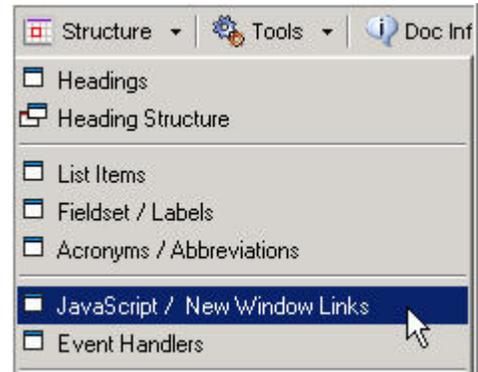
- **NOSCRIPT tags:** Pop-Ups work with JavaScript. Always include text descriptions and links for important information that's contained in JavaScript statements. That way, visitors using tools that aren't JavaScript enabled can get comparable information.

- **Warn users:** Always alert users if your link opens a new browser window. For image links, use alt-text. With text links, you can use the link's Title attribute to describe the link. Both methods increase accessibility and help you include content and keywords for search engine spiders.
- **Alternate navigation:** Avoid using pop-up boxes as the only navigation option - as with a Table of Contents box. This works great for some users, but can be really confusing for someone using a screen reader.

Validation Techniques for Pop-Ups

The AIS Accessibility Toolbar provides a simple tool for identifying links that open new windows or Pop-Ups.

- Displays an icon  next to links that contain JavaScript in the href attribute. The links are also given a dashed border and background color.
- Displays an icon  next to links that contain an onclick event handler and common words used in new window scripts in the href and/or the onclick attribute. The links are also given a dashed border and background color.
- Displays an icon  next to links that have their target attribute specified (_self & _top are excluded). The links are also given a background color.



Pop-Ups driven with JavaScript are to include <NOSCRIPT>. Pop-Ups associated with onclick event handlers are to be evaluated to determine if they are device independent.

Use Navigation Mechanisms in a Consistent Manner:

This guideline is clearly open for debate. A consistent style and structure using common mechanisms across a site is beneficial. Clearly, a reviewer is to exercise judgment in this instance. However, this requirement is included because a complete breakdown in structure can result in significant usability problems for everyone. Review several pages on a single Web site. If you feel like you have visited several Web sites instead of just one, there may not be a consistent design or navigation structure.

Avoid Exclusive Dependence on Graphic Text for Primary Navigation:

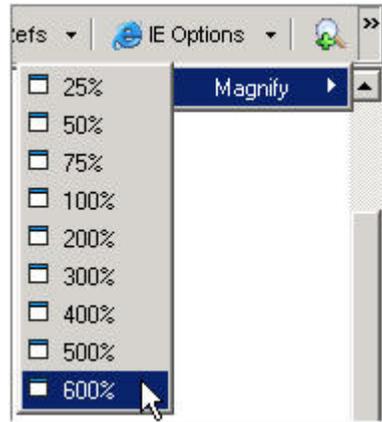
Graphical representation of text is frequently used for primary navigation elements. This is often done to create a unique look, rollovers, or to avoid underlined links, or to apply special effects to text. Often developers use graphic text to make sure users cannot mess up their layouts by resizing the text on the page. All of these reasons have to do more with how the text looks than functions.

However, text is for reading, and there are many instances when people cannot read graphic text. People who need large text for reading cannot enlarge graphic text effectively without a significant reduction in quality or clarity. People who use text-to-speech software to read Web pages cannot read graphic text, unless the developer supplies alternate text. People that are colorblind who need to customize their view of the Web, for example, by applying a custom text color, cannot change the color of graphic text (Alt-text doesn't readily solve this problem), and the solution may require a user to change the way they normally access the Web. Graphic text generally does not work well in flexible layouts, which allow people to access the Web on different devices. Good-looking graphic text can interfere with its primary purpose: reading. This means that the choice to use graphic text is one of form over function. If graphic text is needed for aesthetic reasons, redundant text links can be provided elsewhere on the page.

Validation Techniques for Graphic Text

Use the browser controls to scale text upwards. If the text doesn't resize, it's because it is graphic text or is controlled by "absolute" sizing. Use the **Toggle Image/Alt** function in the AIS Accessibility Toolbar to turn off images. The Alternative Text for the Graphic Text is to be visible if it was included in the markup.

After you've identified Graphic Text that is used for Primary Navigation, determine if redundant navigation is provided elsewhere on the page.



Use the **Magnify** (Zoom) tool in the AIS Accessibility Toolbar to simulate screen magnification software. It only zooms to 600 percent, but can still be a helpful simulation tool. Check to determine if Graphic Text degrades beyond recognition at this level. Consider how it may look to a low vision user. Use the color simulation tools previously described under the "Use of Color" section to test color contrasts. Magnify and Grey Scale the page.

If the Graphic Text Navigation does not "degrade gracefully" under these conditions, redundant text navigation is to be provided:

Provide Descriptive Metadata (Use a Descriptive Page Title):

One of the most simple and useful things a developer can do to enhance accessibility is to use descriptive page titles for their Web pages. The Metadata <Title> is displayed in the Title Bar of the browser Window. This is the first thing a screen reader voices when a Web page is loaded or reloaded. If used correctly, it provides important clues about the content that resides on a particular page and can save a screen reader user considerable time.

Validation Technique

Evaluate the Title Bar content of the browser window when a new page loads. Determine if there is any descriptive relationship between the page title and the page content.

PROPRIETARY FORMATS AND DOWNLOADABLE DOCUMENTS

Section 508:

[1194.22 \(m\)](#) When a Web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page is to provide a link to a plug-in or applet that complies with §1194.21(a) through (l).

OVERVIEW / VALIDATION PROCESS

It is a common practice today to offer documents for download over the Internet. If documents are offered to the general public in this manner, special care is to be taken with proprietary formats that require specialized software for viewing. A link to an appropriate plug-in or viewer is to always be provided.

Accessibility requirements also demand that downloadable documents be accessible and usable by assistive technology such as screen readers. Care is to be taken to ensure that downloadable documents have been constructed with accessibility in mind. The Portable Document Format (PDF), in particular, has raised controversy among users of voice output devices. It is possible to design an accessible PDF document if the content can be “tagged” appropriately with descriptive text that adequately portrays the content of the document.

The Adobe Acrobat Professional Software (version 6.0 and above) provides a PDF accessibility validation tool within the application. This automatic tool can generate a thorough analysis of the documents accessibility. A report file is also generated.

Even the latest version of the Free Acrobat Reader offers a “Quick Accessibility Checker” option. It can identify if “tagged objects” were added to the document. This is the critical structural component of an accessible PDF document.

At least one of these tools is to be utilized to validate a PDF document’s accessibility. If the PDF is not accessible, an accessible version is to be created or an alternative format is to be provided. Not every PDF can be made accessible. In some instances, the nature of the content precludes accessible design. This is particularly true of content that is highly graphical in nature.

A similar approach is to be applied to other proprietary file formats. Microsoft PowerPoint presentations are inherently inaccessible unless the PowerPoint developer relies exclusively on the PowerPoint software’s outline features to construct the presentation and no images or media are incorporated into the presentation. Otherwise, the presentation is to be validated manually to determine if appropriate equivalent content is incorporated for images and other visual elements. This requires a human review of each PowerPoint slide. Use of a screen reader may be necessary for a foolproof check. Objects on a PowerPoint slide can also be “tagged,” and equivalent text is to be provided to each tagged object. Any required reading order of the tagged objects is also to be verified. If accessibility cannot be established for proprietary file formats; an accessible alternative such as HTML or TEXT is to be provided.

Use of an appropriate and approved disclaimer that incorporates information as to where the consumer can obtain required formats, accessible alternatives or assistance is also encouraged. In some instances, an abstract or summary of the requested document may meet the consumer’s needs.