Table of Contents

Web Content Accessibility Guidelines (WCAG) 1.0

Introduction

The Web Content Accessibility Guidelines (WCAG), version 1.0, is a set of guidelines designed to assist web developers in designing and creating universally accessible websites. These guidelines were developed by the Web Accessibility Initiative (WAI), a subgroup of the World Wide Web Consortium (W3C). The guidelines for accessible web design are separated into various checkpoints that are assigned priority levels. The priority levels are as follows:

Priority 1:

A Web content developer **must** satisfy this checkpoint. Otherwise, one or more groups will find it impossible to access information in the document. Satisfying this checkpoint is a basic requirement for some groups to be able to use Web documents.

Priority 2:

A Web content developer **should** satisfy this checkpoint. Otherwise, one or more groups will find it difficult to access information in the document. Satisfying this checkpoint will remove significant barriers to accessing Web documents.

Priority 3:

A Web content developer **may** address this checkpoint. Otherwise, one or more groups will find it somewhat difficult to access information in the document. Satisfying this checkpoint will improve access to Web documents.

The goal of these guidelines, as outlined by the WAI, is to encourage the development and implementation of accessible on-line content for all users, not just users requiring assistive technology. Accroding to the WAI, the guidelines are "…not to discourage content developers from using images, video, etc., but rather explain how to make multimedia content more accessible to a wide audience."

The following pages separate the WCAG 1.0 guidelines into the three priority levels established by the WAI along with the corresponding accessibility checkpoint.

WAI Priority 1 Checkpoints

In General (Priority 1)

- 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.
- 2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup.
- 4.1 Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions).
- 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 must still be possible to read the document.
- 6.2 Ensure that equivalents for dynamic content are updated when the dynamic content changes.
- 7.1 Until user agents allow users to control flickering, avoid causing the screen to flicker.
- 14.1 Use the clearest and simplest language appropriate for a site's content.

And if you use images and image maps (Priority 1)

- 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.

And if you use tables (Priority 1)

- 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.

And if you use frames (Priority 1)

- 12.1 Title each frame to facilitate frame identification and navigation.

And if you use applets and scripts (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.

And if you use multimedia (Priority 1)

- 1.3 Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation.
- 1.4 For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation.

And if all else fails (Priority 1)

- 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.

WAI Priority 2 Checkpoints

In General (Priority 2)

- 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 for images, Priority 3 for text].
- 3.1 When an appropriate markup language exists, use markup rather than images to convey information.
- 3.2 Create documents that validate to published formal grammars.
- 3.3 Use style sheets to control layout and presentation.
- 3.4 Use relative rather than absolute units in markup language attribute values and style sheet property values.
- 3.5 Use header elements to convey document structure and use them according to specification.
- 3.6 Mark up lists and list items properly.
- 3.7 Mark up quotations. Do not use quotation markup for formatting effects such as indentation.
- 6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page.
- 7.2 Until user agents allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off).
- 7.4 Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages.
- 7.5 Until user agents provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects.
- 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.

- 11.1 Use W3C technologies when they are available and appropriate for a task and use the latest versions when supported.
- 11.2 Avoid deprecated features of W3C technologies.
- 12.3 Divide large blocks of information into more manageable groups where natural and appropriate.
- 13.1 Clearly identify the target of each link.
- 13.2 Provide metadata to add semantic information to pages and sites.
- 13.3 Provide information about the general layout of a site (e.g., a site map or table of contents).
- 13.4 Use navigation mechanisms in a consistent manner.

And if you use tables (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).
- 5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting.

And if you use frames (Priority 2)

- 12.2 Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone.

And if you use forms (Priority 2)

- 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.
- 12.4 Associate labels explicitly with their controls.

And if you use applets and scripts (Priority 2)

- 6.4 For scripts and applets, ensure that event handlers are input device-independent.
- 7.3 Until user agents allow users to freeze moving content, avoid movement in pages.
- 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.]
- 9.2 Ensure that any element that has its own interface can be operated in a device-independent manner.
- 9.3 For scripts, specify logical event handlers rather than device-dependent event handlers.

WAI Priority 3 Checkpoints

In General (Priority 3)

- 4.2 Specify the expansion of each abbreviation or acronym in a document where it first occurs.
- 4.3 Identify the primary natural language of a document.
- 9.4 Create a logical tab order through links, form controls, and objects.
- 9.5 Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls.
- 10.5 Until user agents (including assistive technologies) render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links.
- 11.3 Provide information so that users may receive documents according to their preferences (e.g., language, content type, etc.)
- 13.5 Provide navigation bars to highlight and give access to the navigation mechanism.
- 13.6 Group related links, identify the group (for user agents), and, until user agents do so, provide a way to bypass the group.
- 13.7 If search functions are provided, enable different types of searches for different skill levels and preferences.
- 13.8 Place distinguishing information at the beginning of headings, paragraphs, lists, etc.
- 13.9 Provide information about document collections (i.e., documents comprising multiple pages.).
- 13.10 Provide a means to skip over multi-line ASCII art.
- 14.2 Supplement text with graphic or auditory presentations where they will facilitate comprehension of the page.
- 14.3 Create a style of presentation that is consistent across pages.

And if you use images and image maps (Priority 3)

- 1.5 Until user agents render text equivalents for client-side image map links, provide redundant text links for each active region of a client-side image map.

And if you use tables (Priority 3)

- 5.5 Provide summaries for tables.
- 5.6 Provide abbreviations for header labels.
- 10.3 Until user agents (including assistive technologies) render side-by-side text correctly, provide a linear text alternative (on the current page or some other) for *all* tables that lay out text in parallel, word-wrapped columns.

And if you use forms (Priority 3)

- 10.4 Until user agents handle empty controls correctly, include default, place-holding characters in edit boxes and text areas.

Standards for Electronic and Information Technology: An Overview

The Access Board is an independent Federal agency devoted to accessibility for people with disabilities. On December 21, 2000, the Board issued accessibility standards for electronic and information technology under section 508 of the Rehabilitation Act, as amended. The Board also develops and maintains accessibility guidelines for the built environment, transit vehicles, and telecommunications equipment under other laws and enforces design standards for federally funded facilities. Presented here is an overview of the new standards for electronic and information technology and section 508. Questions about the 508 standards should be sent to ***508@access-board.gov****.*

The Law: Section 508

In 1998, Congress amended the Rehabilitation Act and strengthened provisions covering access to information in the Federal sector. As amended, section 508 of the Rehabilitation Act requires access to the Federal government's electronic and information technology. The law covers all types of electronic and information technology in the Federal sector and is not limited to assistive technologies used by people with disabilities. It applies to all Federal agencies when they develop, procure, maintain, or use such technology. Federal agencies must ensure that this technology is accessible to employees and the public to the extent it does not pose an "undue burden." The law directs the Access Board to develop access standards for this technology that will become part of the Federal procurement regulations.

The scope of section 508 is limited to the Federal sector. It does *not* apply to the private sector, nor does it generally impose requirements on the recipients of Federal funds. (However, States receiving assistance under the Assistive Technology Act State Grant program are required to comply with section 508 and the Board's standards, according to the Department of Education, which administers the Act. The Department plans to issue guidance on how the standards apply to the States under the Assistive Technology Act. For further information, e-mail the **Department of Education** or call (202) 205-5666 (voice) or 202-205-5516 (TTY)).

Development of Standards

Shortly after the law was enacted, the Access Board created an advisory committee to develop recommendations on the standards to be developed. In May 1999, the Electronic and Information Technology Access Advisory Committee (EITAAC) completed its work and presented its recommendations to the Board. The committee consisted of 27 representatives from industry, various disability organizations, and other groups with an interest in the issues to be addressed. On March 31, 2000, the Board published proposed standards based closely on the committee's report. The proposed standards were available for public comment for 60 days through publication in the *Federal Register*. The Board sought information and comment on various issues through questions it posed in a

discussion provided in the proposed rule. Over 100 individuals and organizations submitted comments on the standards. Comments were submitted by Federal agencies, representatives of the information technology industry, disability groups, and persons with disabilities. The Board finalized the standards according to its review of the comments and republished them in the *Federal Register*. The final standards, which will become part of the Federal Acquisition Regulations, will help Federal agencies determine whether or not a technology product or system is accessible.

Enforcement and Effective Date

Section 508 uses the Federal procurement process to ensure that technology acquired by the Federal government is accessible. The law also sets up an administrative process under which individuals with disabilities can file a complaint alleging that a Federal agency has not complied with the standards. This process uses the same complaint procedures established under section 504 of the Rehabilitation Act (which covers access to Federally funded programs and services). Individuals may also file a civil action against an agency to seek injunctive relief and attorney's fees (but not compensatory or punitive damages). The enforcement provisions of section 508 take effect six months from the date the Board published its final standards. The Board published its standards on December 21, 2000. Therefore, the enforcement provisions of section 508 are effective as of June 21, 2001. (Originally, they were to take effect August 7, 2000, but section 508 was further amended to allow time for compliance after publication of the standards).

By statute, the enforcement provisions of section 508 apply only to electronic and information technology *procured on or after the effective date.* As a result, section 508 does not authorize complaints or lawsuits to retrofit technology procured before this date to meet the Board's standards. However, even though section 508's enforcement mechanisms apply only to *procurement*, the law does require access to technology *developed, used or maintained* by a Federal agency. Further, other sections of the Rehabilitation Act require access to Federal programs (section 504) and accommodation of Federal employees with disabilities (sections 501 and 504); it is possible that Federal agencies will use the Board's section 508 standards as a yardstick to measure compliance with these other sections of the law.

"Undue Burden"

A Federal agency does not have to comply with the technology accessibility standards if it would impose an undue burden to do so. This is consistent with language used in the Americans with Disabilities Act (ADA) and other civil rights legislation, where the term `undue burden' has been defined as "significant difficulty or expense." However, the agency must explain why meeting the standards would pose an undue burden for a given procurement action, and must still provide people with disabilities access to the information or data that is affected.

The Standards

General (Subpart A)

The standards define the types of technology covered and set forth provisions that establish a minimum level of accessibility. The application section (1194.2) outlines the scope and coverage of the standards. The standards cover the full range of electronic and information technologies in the Federal sector, including those used for communication, duplication, computing, storage, presentation, control, transport and production. This includes computers, software, networks, peripherals and other types of electronic office equipment. The standards define *electronic and information technology*, in part, as "any equipment or interconnected system or subsystem of equipment, that is used in the creation, conversion, or duplication of data or information."

Subpart A also explains what is exempt (1194.3), defines terms (1194.4), and generally recognizes alternatives to what is required that provide equal or greater access (1194.5). Consistent with the law, the standards exempt systems used for military command, weaponry, intelligence, and cryptologic activities (but not routine business and administrative systems used for other defense-related purposes or by defense agencies or personnel). The standards also exempt "back office" equipment used only by service personnel for maintenance, repair, or similar purposes.

The standards cover technology procured by Federal agencies under contract with a private entity, but apply only to those products directly relevant to the contract and its deliverables. An exception clarifies that the standards do not apply to technology that is incidental to a Federal contract. Thus, those products that are not specified as part of a contract with a Federal agency would not have to comply with the standards. For example, a firm that produces a report for a Federal agency under a contract would not have to procure accessible computers and word processing software even if they were used exclusively for the contract; however, compliance would be required if such products were to become the property of the Federal agency as contract deliverables or if the Federal agency purchased the products to be used by the contractor as part of the project. If a Federal agency contracts with a firm to develop its web site, the standards would apply to the new web site for the agency but not to the firm's own web site.

Technical Standards (Subpart B)

The standards provide criteria specific to various types of technologies, including: software applications and operating systems; web-based information or applications; telecommunication products; video and multimedia products; self contained, closed products (e.g., information kiosks, calculators, and fax machines); desktop and portable computers.

This section provides technical specifications and performance-based requirements, which focus on the functional capabilities of covered technologies. This dual approach recognizes the dynamic and continually evolving nature of the technology involved as well as the need for clear and specific standards to facilitate compliance. Certain provisions are designed to ensure compatibility with adaptive equipment people with disabilities commonly use for information and communication access, such as screen readers, Braille displays, and TTYs.

Software Applications and Operating Systems (1194.21)

Most of the specifications for software pertain to usability for people with vision impairments. For example, one provision requires alternative keyboard navigation, which is essential for people with vision impairments who cannot rely on pointing devices, such as a mouse. Other provisions address animated displays, color and contrast settings, flash rate, and electronic forms, among others.

Web-based Intranet and Internet Information and Applications (1194.22)

The criteria for web-based technology and information are based on access guidelines developed by the Web Accessibility Initiative of the World Wide Web Consortium. Many of these provisions ensure access for people with vision impairments who rely on various assistive products to access computer-based information, such as screen readers, which translate what's on a computer screen into automated audible output, and refreshable Braille displays. Certain conventions, such as verbal tags or identification of graphics and format devices, like frames, are necessary so that these devices can "read" them for the user in a sensible way. The standards do not prohibit the use of web site graphics or animation. Instead, the standards aim to ensure that such information is also available in an accessible format. Generally, this means use of text labels or descriptors for graphics and certain format elements. (HTML code already provides an "Alt Text" tag for graphics which can serve as a verbal descriptor for graphics). This section also addresses the usability of multimedia presentations, image maps, style sheets, scripting languages, applets and plug-ins, and electronic forms.

The standards apply to Federal web sites but not to private sector web sites (unless a site is provided under contract to a Federal agency, in which case only that web site or portion covered by the contract would have to comply). Accessible sites offer significant advantages that go beyond access. For example, those with "text-only" options provide a faster downloading alternative and can facilitate transmission of web-based data to cell phones and personal digital assistants.

Telecommunications Products (1194.23)

The criteria of this section are designed primarily to ensure access to people who are deaf or hard of hearing. This includes compatibility with hearing aids, cochlear implants, assistive listening devices, and TTYs. TTYs are devices that enable people with hearing or speech impairments to communicate over the telephone; they typically include an acoustic coupler for the telephone handset, a simplified keyboard, and a visible message display. One requirement calls for a standard non-acoustic TTY connection point for telecommunication products that allow voice communication but that do provide TTY functionality. Other specifications address adjustable volume controls for output, product interface with hearing technologies, and the usability of keys and controls by people who may have impaired vision or limited dexterity or motor control.

Video or Multimedia Products (1194.24)

Multimedia products involve more than one media and include, but are not limited to, video programs, narrated slide production, and computer generated presentations. Provisions address caption decoder circuitry (for any system with a screen larger than 13 inches) and secondary audio channels for television tuners, including tuner cards for use in computers. The standards also require captioning and audio description for certain training and informational multimedia productions developed or procured by Federal agencies. The standards also provide that viewers be able to turn captioning or video description features on or off.

Self Contained, Closed Products (1194.25)

This section covers products that generally have imbedded software but are often designed in such a way that a user cannot easily attach or install assistive technology. Examples include information kiosks, information transaction machines, copiers, printers, calculators, fax machines, and similar types of products. The standards require that access features be built into the system so users do not have to attach an assistive device to it. Other specifications address mechanisms for private listening (handset or a standard headphone jack), touchscreens, auditory output and adjustable volume controls, and location of controls in accessible reach ranges.

Desktop and Portable Computers (1194.26)

This section focuses on keyboards and other mechanically operated controls, touch screens, use of biometric form of identification, and ports and connectors.

Functional Performance Criteria (Subpart C)

The performance requirements of this section are intended for overall product evaluation and for technologies or components for which there is no specific requirement under the technical standards in Subpart B. These criteria are designed to ensure that the individual accessible components work together to create an accessible product. They cover operation, including input and control functions, operation of mechanical mechanisms, and access to visual and audible information. These provisions are structured to allow people with sensory or physical disabilities to locate, identify, and operate input, control and mechanical functions and to access the information provided, including text, static or dynamic images, icons, labels, sounds or incidental operating cues.

Information, Documentation, and Support (Subpart D)

The standards also address access to all information, documentation, and support provided to end users (e.g., Federal employees) of covered technologies. This includes user guides, installation guides for end-user installable devices, and customer support and technical support communications. Such information must be available in alternate formats upon request at no additional charge. Alternate formats or methods of communication, can include Braille, cassette recordings, large print, electronic text, Internet postings, TTY access, and captioning and audio description for video materials.

Federal Accessibility Standards for Web-based Intranet and Internet Information and Applications

(Provided by www.usability.gov)

This page contains excerpts from Electronic and Information Technology Accessibility Standards issued by the ARCHITECTURAL AND TRANSPORTATION BARRIERS COMPLIANCE BOARD.

Shown below are:
Paragraphs from the Overview of the Standards
Subpart B — Technical Standards: Sec. 1194.22 Web-based intranet and internet information and applications.

This page only contains an excerpt of the summary and standards that directly relate to Web sites. Other parts of the standard may apply to your situation. See the complete, officially posted standards at http://www.access-board.gov/news/508-final.htm.

Electronic and Information Technology Accessibility Standards
ARCHITECTURAL AND TRANSPORTATION BARRIERS COMPLIANCE BOARD
[Published in the Federal Register on December 21, 2000]

Summary:

Web-based Intranet and Internet Information and Applications (1194.22)
The criteria for web-based technology and information are based on access guidelines developed by the Web Accessibility Initiative of the World Wide Web Consortium. Many of these provisions ensure access for people with vision impairments who rely on various assistive products to access computer-based information, such as screen readers, which translate what's on a computer screen into automated audible output, and refreshable Braille displays. Certain conventions, such as verbal tags or identification of graphics and format devices, like frames, are necessary so that these devices can "read" them for the user in a sensible way. The standards do not prohibit the use of web site graphics or animation. Instead, the standards aim to ensure that such information is also available in an accessible format. Generally, this means use of text labels or descriptors for graphics and certain format elements. (HTML code already provides an "Alt Text" tag for graphics which can serve as a verbal descriptor for graphics). This section also addresses the usability of multimedia presentations, image maps, style sheets, scripting languages, applets and plug-ins, and electronic forms.

The standards apply to Federal web sites but not to private sector web sites (unless a site is provided under contract to a Federal agency, in which case only that web site or portion covered by the contract would have to comply). Accessible sites offer significant advantages that go beyond access. For example, those with "text-only" options provide a faster downloading alternative and can facilitate transmission of web-based data to cell phones and personal digital assistants.

§ 1194.22 Web-based intranet and internet information and applications.

(a) A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).

(b) Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.

(c) Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.

(d) Documents shall be organized so they are readable without requiring an associated style sheet.

(e) Redundant text links shall be provided for each active region of a server-side image map.

(f) Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

(g) Row and column headers shall be identified for data tables.

(h) Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

(i) Frames shall be titled with text that facilitates frame identification and navigation.

(j) Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.

(k) A text-only page, with equivalent information or functionality, shall 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 shall be updated whenever the primary page changes.

(l) When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.

(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 must provide a link to a plug-in or applet that complies with §1194.21**(a)** through **(l)**.

(n) When electronic forms are designed to be completed on-line, the form shall 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.

(o) A method shall be provided that permits users to skip repetitive navigation links.

(p) When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.

Note to §1194.22:

1. The Board interprets paragraphs **(a)** through **(k)** of this section as consistent with the following priority 1 Checkpoints of the Web Content Accessibility Guidelines 1.0 (WCAG 1.0) (May 5, 1999) published by the Web Accessibility Initiative of the World Wide Web Consortium:

Section 1194.22 Paragraph	WCAG 1.0 Checkpoint
(a)	1.1
(b)	1.4
(c)	2.1
(d)	6.1
(e)	1.2
(f)	9.1
(g)	5.1
(h)	5.2
(i)	12.1
(j)	7.1
(k)	11.4

2. Paragraphs **(l)**, **(m)**, **(n)**, **(o)**, and **(p)** of this section are different from WCAG 1.0. Web pages that conform to WCAG 1.0, level A (i.e., all priority 1 checkpoints) must also meet paragraphs **(l)**, **(m)**, **(n)**, **(o)**, and **(p)** of this section to comply with this section. WCAG 1.0 is available at http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505.

Using the Alt-Tag

Introduction

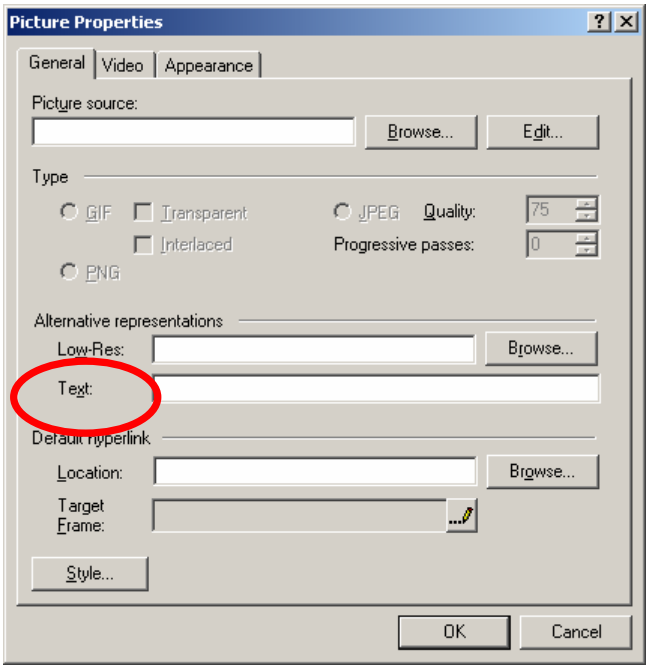
An important component of developing accessible web pages is to add the alt-attribute to images. This allows visually impaired individuals using screen reader software or a text-only browser to "hear" the description of the image presented on the web page. Adding the alt-attribute is required under the WAI Priority 1 Guidelines and the Section 508 Standards.

Using FrontPage 2000/2002:

1. Insert the image to the web page using the **Insert** command from the menu bar.

2. **Right click** on the image using the mouse, or select the image and use the **Alt+Enter** keystroke, to access the Picture Properties menu.

3. Under the **Alternative Representations** heading, type the alternative text for the image in the **Text:** box.



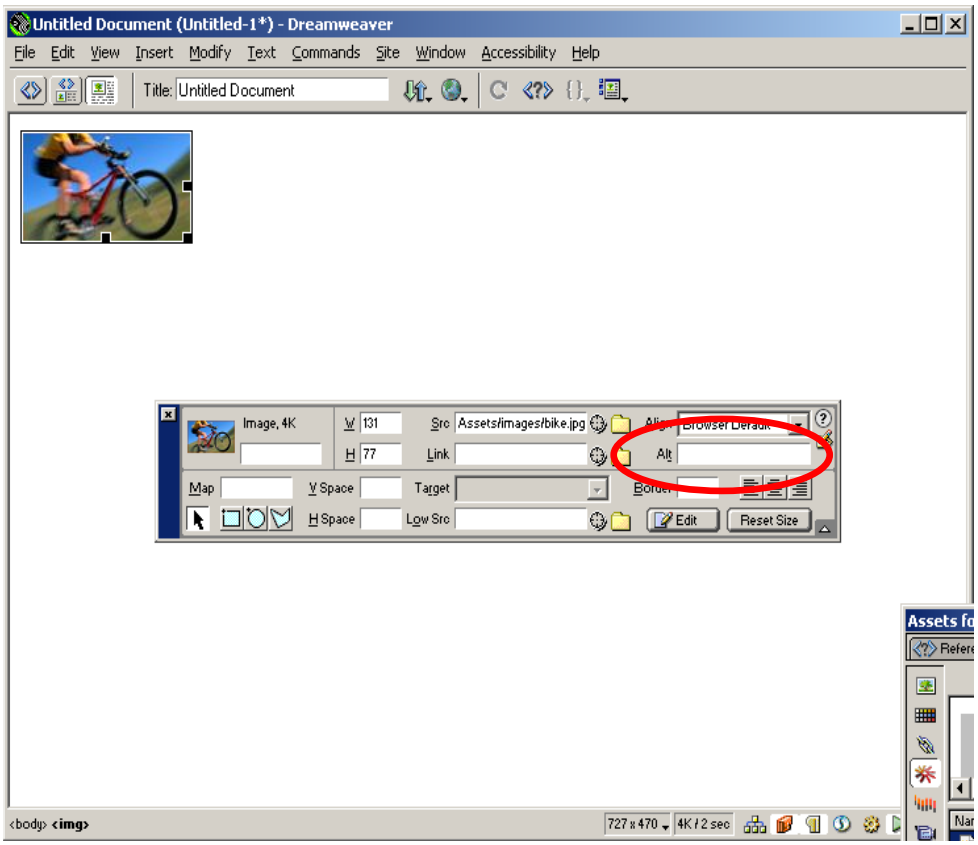
4. Select **OK**.

You have just added an alt-tag to an image.

Using Dreamweaver 4:

1. Insert the image to the web page using the **Insert** command from the menu bar, or the keyboard command **Ctrl+Alt+I**.

2. **Right click** on the image using the mouse, or select the image using the arrow and shift key. The Picture Properties box will then appear (see screen shot).



3. Under the **Alt** heading, type the alternative text for the image in the box.

You have just added an alt-tag to an image.

Editing the HTML Code:

You can edit the HTML code using several different methods. Most web page authoring programs will allow the developer to switch between the WYSIWYG interface and the underlying HTML code interface. You can also edit the HTML code by opening the web page in Notepad, WordPerfect, or Word 2000. The HTML code can then be edited as a regular text document. Here is more information for adding an alt-attribute by editing the HTML code:

1. Identify the image that you wish to add the alt-attribute. An easy method is to look for the words **img src=**. This will be the line in which you place the alt= tag

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<img src="file:///C|/Program%20Files/Macromedia/Dreamweaver%204/Tutorial/
Compass_Site /Assets/images/bike.jpg" width="131" height="77">
</body>
</html>
```

2. Insert the HTML code **alt=”descriptive text goes here”** before the > sign.

3. This will add the alt-attribute to the desired image.

You have just added an alt-tag to an image by editing the HTML code.

Using the d-Link:

The d-link is an interim solution that web designers are using to communicate the meaning of an image in greater detail. This solution will be replaced by the use of the LONGDESC attribute when browsers are able to recognize HTML 4.0. Until then, it is appropriate to use this solution to provide the user with a greater description of the image presented on the web page.

The convention for the "d-link" option is to place a lower-case "d" near the lower right region of the desired image. The lower-case "d" is then hyperlinked to a text only document that describes the image in more detail. An example is provided below.

1. Insert an image into the web page you are creating.

2. Add a lower-case "d" to the bottom right corner of the image.

3. Create a separate web page that contains a more detailed description of the image you are displaying on your web page.

4. Create a hyperlink from the lower-case "d" to the separate web page that contains the detailed description of the image on your web page.

You have created a "d-link" that will provide more information to a visually impaired or blind individual.

Editing a JAVA Applet:

Java applets are widely used by web designers who desire an interactive or more dynamic presentation for the internet visitor. JAVA applets are essentially small programs that can be launched from a web page and will run on the visitor's computer. The advantage of a JAVA applet is that it will not write information to the user's hard disk and the program can be run on several different operating system platforms.

A major disadvantage of JAVA applets is that these programs are not accessible to screen-readers. For example, a screen-reader program will not read a JAVA applet that displays updated stock values on a web page. Classroom management systems, such as Blackboard or WebCT, contain virtual chat modules that are JAVA applets. Once again, screen-readers will not recognize or communicate to the user information presented in a JAVA applet window.

To communicate to a user that a JAVA applet is on the page, a web author can use the alt-attribute.

Example:

1. Locate the applet code on in the HTML. This will look like <APPLET>.

2. Insert the appropriate description of the JAVA applet using the following syntax: **alt= "descriptive text goes here"** .

3. Another method to use is to simply insert text between the <APPLET> and </APPLET> tags. The inserted text could be a brief description of the applet and what is the function of the applet. This is beneficial for users who have applets turned off in their internet browser or their internet browser does not support JAVA applets.

Cascading Style Sheets, Level 1

Overview

Cascading Style Sheets (CSS) are a method of web design that formats web page content according to a presentation style specified by the web page author. There are several advantages to using CSS to format the presentation elements of a web page. CSS essentially separates document content from the manner in which it is presented, thus allowing for more fluid transitions between various browser platforms. CSS also provides for more precise control for spacing, alignment, and positioning of content without relying on the need for layout tables or frames. Font style, color, and font size can all be manipulated using CSS as well.

Unfortunately, not all browsers render style sheets in a similar manner. Additionally, earlier browsers may try to interpret CSS code as part of the HTML, thus rendering the document unreadable. This chapter will discuss the benefits and limitations of using CSS for web design.

Accessibility Benefits

CSS offers the potential for users to customize their web browser such that the content displayed in their preferred Internet browser will present the information in the manner the user chooses. For a low-vision individual, this provides the opportunity to enlarge web content displayed in the internet browser at a resolution appropriate to the individual. Using CSS provides the user with the option of manipulating the way in which the web content is presented in their specific Internet browser.

Types of CSS

There are essentially four different types of style sheets that can impact the manner in which content is displayed.

Inline
Inline style sheets apply the formatting elements locally – that is, at the level of the text within the web page document. An inline style sheet may contain the following:

```
<h1 style="font-family:sans-serif; font-size:20pt;
color:#996633">
```

Internal
Internal style sheets reside within the <**head**> section of an HTML document and affect the presentation of content throughout the document. An internal style sheet may contain the following:

```
<head>
<style>
<!—
body {font-family:Helvetica, sans-serif; font-size:extra-
large; color:orange}
-->
</ style>
</ head>
```

External

External style sheets exist outside of the document structure and are referenced when the document (or web page) is loaded into the browser. External style sheets have the advantage of being easy to change in order to affect a large number of web pages simultaneously. An external style sheet may contain the following:

```
h2 {font-family: Helvetica, sans-serif;
   font-size: extra-large;
   color: wheat
}

p {font-family: serif;
   font-size: 14;
   color: black;
}
```

This specifies all level 2 headings will be Helvetica font (or another sans-serif font), extra-large size, and a wheat color. All paragraph content will be a serif font (the default on the computer), 14 point, and a black color.

Imported

Imported style sheets are similar to **External** style sheets in that the attributed styles are *imported* into the document within the **<Style>** tag. An imported style sheet could include the following:

```
STYLE TYPE="text/css" MEDIA="screen, projection">
<!--
  @import url(http://www.htmlhelp.com/style.css);
  @import url(/stylesheets/punk.css);
  DT { background: yellow; color: black }
-->
   </STYLE>
```

Style Sheet Formatting

The syntax for a linked style sheet is contained in a text document with an extension of **.css**. The **.css** files remain on the web server and are linked to their respective web page documents. The syntax for style sheets is the following:

```
h1  {property1: value1;
     property2: value1, value2;
     property3: value1;
}

h2  {property1: value1;
     property2: value1;
}

P  {property1: value1;
     property2: value1;
     property3: value1;
}
```

Elements
The elements are those for which you are setting a style. For instance, in the previous example, the Heading 1, Heading 2, and Paragraph functions would receive specific formatting as set by the style sheet.

Element Name	**Element Code**
Headings:	H1, H2, H3, H4, H5, H6
Body Text:	P, BLOCKQUOTE
Hyperlinks:	A
Lists:	UL, LI, OL

Properties
Properties represent the category of the desired presentation appearance. Properties include descriptions for the font-family, font-style, color, etc. These properties are then followed by the appropriate value for this category. A list of the different properties and the associated values is in the next section.

Values
Values are what specify how the property is to be displayed within the style sheet. Each property has several values from which to choose in order to present the content according to the web designers intent. A list of the different values with their associated properties is available in the next section.

Font Properties:

Syntax:	Allowed Values:
font-family	serif (e.g., Times) sans-serif (e.g., Arial) cursive (e.g., *Zapf-Chancery*) fantasy monospace (e.g., `Courier`) or specific font name

Example code:
```
P { font-family: "New Century Schoolbook", Times, serif }
```

Syntax:	Allowed Values:
font-style	normal *italic* *oblique*

Example code:
```
H1 { font-style: oblique }
P  { font-style: normal }
```

Syntax:	Allowed Values:
font-variant	normal small-caps

Example code:
```
SPAN { font-variant: small-caps }
```

Syntax:	Allowed Values:
font-weight	normal **bold** **bolder** lighter 100-500 (lighter) **600-900 (darker)**

Example code:
```
H1 { font-weight: 800 }
```

Syntax:	Allowed Values:
font-size	absolute-size (xx-small, x-small, small, medium, large, x-large, xx-large) relative-size (larger, smaller) length (pt) percentage (%)

Example code:

```
H1 { font-size: large }
P  { font-size: 12pt }
LI { font-size: 90% }
```

Color and Background Properties

Syntax:	Allowed Values:
color	The color values can be defined using a RGB color chart or a verbal representation of a particular color: (aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow, wheat)

Example code:

```
H1 { color: blue }
H2 { color: #000080 }
```

Syntax:	Allowed Values:
background-color	color-value transparent

Example code:

```
BODY { background-color: white }
H1   { background-color: #000080 }
```

Syntax:	Allowed Values:
background-image	<url> none

Example code:

```
BODY { background-image: url(/images/foo.gif) }
P    { background-image: url(http://www.htmlhelp.com/bg.png) }
```

Syntax:	Allowed Values:
background-repeat	repeat repeat-x repeat-y no-repeat

Example code:

```
BODY { background: white url(candybar.gif);
       background-repeat: repeat-x }
```

The background-repeat property determines how a specified background image is repeated. The repeat-x value will repeat the image horizontally while the repeat-y value will repeat the image vertically

Text Properties

Syntax:	Allowed Values:
word-spacing	normal length format (e.g., relative or absolute)

Example code:

```
P EM   { word-spacing: 0.4em }
P.NOTE { WORD-SPACING: -0.2EM }
```

Syntax:	Allowed Values:
letter-spacing	normal length format (e.g., relative or absolute)

Example code:

```
H1     { letter-spacing: 0.1em }
P.note { letter-spacing: -0.1em }
```

Syntax:	Allowed Values:
text-decoration	none underline overline ~~line-through~~ blink

Example code:

```
   A:link, A:visited, A:active { text-decoration: none }
```

Syntax:	Allowed Values:
vertical-align	baseline sub super top text-top middle bottom text-bottom percentage (%)

Percentage will move the element the specified percentage from the baseline (negative values can be used for subscripts).

Example code:

```
IMG.middle { vertical-align: middle }
.exponent  { vertical-align: super }
```

Syntax:	Allowed Values:
text-transform	none Capitalize (capitalize first letter in each word) UPPERCASE (capitalize all the words) lowercase (uses small letters for each word)

Example code:

```
H1 { text-transform: uppercase }
H2 { text-transform: capitalize }
```

Syntax:	Allowed Values:
text-align	left right center justify

Example code:

```
H1          { text-align: center }
P.newspaper { text-align: justify }
```

Syntax:	Allowed Values:
text-indent	length (relative or absolute) percentage (%)

Example code:

```
   P { text-indent: 5em }
```

Syntax:	Allowed Values:
line-height	normal number (value) length (relative or absolute) percentage (%)

The line-height property will accept a value to control the spacing between baselines of text. When the value is a number, the line height is calculated by multiplying the element's font size by the number. Percentage values are relative to the element's font size. Negative values are not permitted.

Example code:

```
P { line-height: 200% }
```

List Properties

Syntax:	Allowed Values:
list-style-type	none disc ○ circle ● square ■ decimal 1 2 3 etc. lower-roman i ii iii etc. upper-roman I II III etc. lower-alpha a b c etc. upper-alpha A B C etc.

The list-style-type property specifies the type of list-item marker, and is used if list-style-image is none or if image loading is turned off.

Examples:

```
LI.square { list-style-type: square }
UL.plain  { list-style-type: none }
OL        { list-style-type: upper-alpha } /* A B C */
```

Syntax:	Allowed Values:
list-style-image	none <url of image>

The list-style-image property specifies the image that will be used as list-item marker when image loading is turned on, replacing the marker specified in the **list-style-type** property.

Examples:

```
UL.check { list-style-image: url(/LI- markers/checkmark.gif) }
UL LI.x  { list-style-image: url(x.png) }
```

Syntax:	Allowed Values:
list-style-position	inside outside

The list-style-position property takes the value inside or outside, with outside being the default. This property determines where the marker is placed in regard to the list item. If the value inside is used, the lines will wrap under the marker instead of being indented.
Examples:
Outside rendering:
* **List item 1**
  second line of list item

Inside rendering:
* **List item 1**
second line of list item

Syntax:	Allowed Values:
list-style	<list-style-type> <list-style-position> <url>

The list-style property is a shorthand for the list-style-type, list-style-position, and list-style-image properties.

Examples:
```
LI.square { list-style: square inside }
UL.plain  { list-style: none }
UL.check  { list-style: url(/LI-markers/checkmark.gif) circle }
OL        { list-style: upper-alpha }
   OL OL     { list-style: lower-roman inside }
```

Length and Percentage Units

(from www.w3c.org)

Length units
The format of a length value is an optional sign character ('+' or '-', with '+' being the default) immediately followed by a number (with or without a decimal point) immediately followed by a unit identifier (a two-letter abbreviation). After a '0' number, the unit identifier is optional.

Some properties allow negative length units, but this may complicate the formatting model and there may be implementation-specific limits. If a negative length value cannot

be supported, it should be clipped to the nearest value that can be supported. There are two types of length units: relative and absolute. Relative units specify a length relative to another length property. Style sheets that use relative units will more easily scale from one medium to another (e.g. from a computer display to a laser printer).

These relative units are supported:

```
H1 { margin: 0.5em }      /* ems, the height of the element's font */
H1 { margin: 1ex }        /* x-height, ~ the height of the letter 'x' */
P  { font-size: 12px }    /* pixels, relative to canvas */
```

The relative units 'em' and 'ex' are relative to the font size of the element itself. The only exception to this rule in CSS1 is the 'font-size' property where 'em' and 'ex' values refer to the font size of the parent element.

Pixel units, as used in the last rule, are relative to the resolution of the canvas, i.e. most often a computer display. If the pixel density of the output device is very different from that of a typical computer display, the UA should rescale pixel values. The suggested *reference pixel* is the visual angle of one pixel on a device with a pixel density of 90dpi and a distance from the reader of an arm's length. For a nominal arm's length of 28 inches, the visual angle is about 0.0227 degrees.

Child elements inherit the computed value, not the relative value:

```
BODY {
  font-size: 12pt;
  text-indent: 3em;  /* i.e. 36pt */
}

H1 { font-size: 15pt }
```

In the example above, the 'text-indent' value of 'H1' elements will be 36pt, not 45pt.

Absolute length units are only useful when the physical properties of the output medium are known. These absolute units are supported:

```
H1 { margin: 0.5in }      /* inches, 1in = 2.54cm */
H2 { line-height: 3cm }   /* centimeters */
H3 { word-spacing: 4mm }  /* millimeters */
H4 { font-size: 12pt }    /* points, 1pt = 1/72 in */
H4 { font-size: 1pc }     /* picas, 1pc = 12pt */
```

Percentage units
The format of a percentage value is an optional sign character ('+' or '-', with '+' being the default) immediately followed by a number (with or without a decimal point) immediately followed by "%".

Percentage values are always relative to another value, for example a length unit. Each property that allows percentage units also defines what value the percentage value refer to. Most often this is the font size of the element itself:

P { line-height: 120% } /* 120% of the element's 'font-size' */

Color

(from www.w3c.org)

A color is either a keyword or a numerical RGB specification. The suggested list of keyword color names is:

aqua	black
blue	fuchsia
gray	green
lime	maroon
navy	olive
purple	red
silver	teal
white	yellow

These 16 colors are taken from the Windows VGA palette, and their RGB values are not defined in this specification.

```
BODY {color: black; background: white }
H1 { color: maroon }
H2 { color: olive }
```

The RGB color model is being used in numerical color specifications. These examples all specify the same color:

```
EM { color: #f00 }              /* #rgb */
EM { color: #ff0000 }           /* #rrggbb */
EM { color: rgb(255,0,0) }      /* integer range 0 - 255 */
EM { color: rgb(100%, 0%, 0%) } /* float range 0.0% - 100.0% */
```

The format of an RGB value in hexadecimal notation is a '#' immediately followed by either three or six hexadecimal characters. The three-digit RGB notation (#rgb) is converted into six-digit form (#rrggbb) by replicating digits, not by adding zeros. For example, #fb0 expands to #ffbb00. This makes sure that white (#ffffff) can be specified

with the short notation (#fff) and removes any dependencies on the color depth of the display.

The format of an RGB value in the functional notation is 'rgb(' followed by a comma-separated list of three numerical values (either three integer values in the range of 0-255, or three percentage values in the range of 0.0% to 100.0%) followed by ')'. Whitespace characters are allowed around the numerical values.

Values outside the numerical ranges should be clipped.

```
EM { color: rgb(255,0,0) }       /* integer range 0 - 255 */
EM { color: rgb(300,0,0) }       /* clipped to 255 */
EM { color: rgb(110%, 0%, 0%) }  /* clipped to 100% */
```

Classes of Style

An advantage of using CSS for web pages is the ability to globally affect the presentation of specific throughout a website. However, there is always the situation in which certain elements require different formatting. This can be accomplished by using the class element in order to specify the final presentation of select elements.

The following code would apply a large, sans-serif, italicized font to all first-level headings.

```
H1 {font-family: Arial, sans-serif;
      font-style: italic;
      font-size: large;
}
```

However, if you wanted to have some of the first-level heading to appear in a blue, serif font, it would be necessary to create a class the specifies the desired appearance.

```
.specialh1  {font-family: Times, serif;
               color: blue;
}
```

In the HTML code, it would be necessary to call in this special class in order to display this specific presentation style.

```
<h1>This is an example of the regular Heading 1 style</h1>

<h1 class="specialh1">This is an example of the special Heading 1
style</h1>
```

Using the `<div>` tag, a designer can also apply specific styles to whole blocks of content on a web page. Content within the `<div>` tag would change the appearance to that specified by the class element.

Example:

```
<div class=”specialh1”>
   <p>Paragraph content would go here…..</p>
   <table>
        Table data would be formatted as well….
   </table>
</div>
```

Inheritance

Another advantage to using CSS in web page design is the ability of different styles to *inherit* appearance values from previous levels. For instance, in the previous example discussing the Class property, the Heading 1 style was a large, sans-serif, italicized font.

```
H1 {font-family: Arial, sans-serif;
     font-style: italic;
     font-size: large;
}
```

We also created a special class that changed the font-family to Times and the color to blue.

```
.specialh1  {font-family: Times, serif;
              color: blue;
}
```

However, because we did not specify differences to the font-style and the font-size, the .specialh1 class will retain those property values when displaying the web page. This is due to inheritance.

Contextual selectors

(from www.w3c.org)

Inheritance saves CSS designers typing. Instead of setting all style properties, one can create defaults and then list the exceptions. To give 'EM' elements within 'H1' a different color, one may specify:

```
H1 { color: blue }
EM { color: red }
```

When this style sheet is in effect, all emphasized sections within or outside 'H1' will turn red. Probably, one wanted only 'EM' elements within 'H1' to turn red and this can be specified with:

```
H1 EM { color: red }
```

The selector is now a search pattern on the stack of open elements, and this type of selector is referred to as a *contextual selector*. Contextual selectors consist of several simple selectors separated by whitespace (all selectors described up to now have been simple selectors). Only elements that match the last simple selector (in this case the 'EM' element) are addressed, and only if the search pattern matches. Contextual selectors in CSS1 look for ancestor relationships, but other relationships (e.g. parent-child) may be introduced in later revisions. In the example above, the search pattern matches if 'EM' is a descendant of 'H1', i.e. if 'EM' is inside an 'H1' element.

```
UL LI    { font-size: small }
UL UL LI { font-size: x-small }
```

Here, the first selector matches 'LI' elements with at least one 'UL' ancestor. The second selector matches a subset of the first, i.e. 'LI' elements with at least two 'UL' ancestors. The conflict is resolved by the second selector being more specific because of the longer search pattern. Contextual selectors can look for element types, CLASS attributes, ID attributes or combinations of these:

```
DIV P           { font: small sans-serif }
.reddish H1     { color: red }
#x78y CODE      { background: blue }
DIV.sidenote H1 { font-size: large }
```

The first selector matches all 'P' elements that have a 'DIV' among the ancestors. The second selector matches all 'H1' elements that have an ancestor of class 'reddish'. The third selector matches all 'CODE' elements that are descendants of the element with 'ID=x78y'. The fourth selector matches all 'H1' elements that have a 'DIV' ancestor with class 'sidenote'.

Several contextual selectors can also be grouped together:

```
H1 B, H2 B, H1 EM, H2 EM { color: red }
```

Which is equivalent to:

```
H1 B { color: red }
H2 B { color: red }
H1 EM { color: red }
H2 EM { color: red }
```

The ! Important Property

Using the **! important** property, web designers can force specific styles to appear on a web page. The **! important** declaration will overide all other CSS rules for the specified property and value. Thus, the web page authors can create style sheet rules that will overide those as set by users.

Example:

```
H1 { color: black ! important; background: white ! important }
P  { font-size: 12pt ! important; font-style: italic }
```

This will format the heading 1 styles to have a black text color with a white background. Additionally, all paragraphs will have a 12pt font-size, but may or may not have a font-style of italics.

It is not recommended that web designers use the **! important** property to format web page content. One advantage to using CSS is that there is some flexibility in the manner in which web content is rendered. A low-vision user could create their own style sheet in order to make the on-line content more usable for their needs. The following style sheet is just one example that may be useful to a low-vision user.

```
H1 { font-family: Verdana ! important;
        font-style: normal ! important;
        font-size: xx-large ! important;
        color: black ! important;
}

P   { font-family: Times ! important;
        font-style: normal !important;
        font-size: large ! important;
        color: yellow ! important;
        background: black ! important;
}
```

Using Data Tables

Introduction

The World Wide Web offers the ability of designers to communicate vast amounts of information in a relatively simple and easy method. However, the rendering of such content can vary from web browser to web browser in such a manner that the presentation and design considerations that originally dictated the web page are now lost due to variations in the user's web browser. To solve this issue, many web authors utilize tables to control for layout and presentation purposes. In addition to using tables for presentation purposes, tables are also used to present data information on a web page. Tables used for presentation purposes are generally referred to a Layout Tables while tables used to organize information are referred to as Data Tables. This section will present information on designing accessible Data Tables.

Formatting Data Tables

Data tables are exactly what the name implies - a table that contains data about some information sequenced in a specific format. For example, if you have a list of days, a list of appointments on specific days, and a list of specific times for those appointments on specific days, it would make sense to develop a table to display that information. Here is a practical example below:

	Sun.	**Mon.**	**Tues.**	**Wed.**	**Thurs.**	**Fri.**	**Sat.**
9:00 AM	Sleep		Work	Work		Work	Sleep
12:00 PM	Nap	Lunch			Lunch		Nap
2:00 PM	Hamster Feeding			Cat Pruning			Breakfast
5:00 PM				Hospital			Nap
6:00 PM		Football					

The benefit of such a data table is that it provides information in a visually organized format such that one can quickly identify the day of the appointment and time in a simple and easy manner. There are several methods to make data tables more accessible using specific HTML attributes.

Using the Scope Method:

(from Section 508 Standards, www.access-board.gov)
Using the "scope" attribute is one of the most effective ways of making HTML compliant with these requirements. It is also the simplest method to implement. The scope attribute also works with some (but not all) assistive technology in tables that use "colspan" or "rowspan" attributes in table header or data cells.

The first row of each table should include column headings. Typically, these column headings are inserted in <TH> tags, although <TD> tags can also be used. These tags at the top of each column should include the following attribute:

scope="col"

By doing this simple step, the text in that cell becomes associated with every cell in that column. Unlike using other approaches (notably "id" and "headers") there is no need to include special attributes in each cell of the table. Similarly, the first column of every table should include information identifying information about each row in the table. Each of the cells in that first column are created by either <TH> or <TD> tags. Include the following attribute in these cells:

scope="row"

By simply adding this attribute, the text in that cell becomes associated with every cell in that row. While this technique dramatically improves the usability of a web page, using the scope attribute does not appear to interfere in any way with browsers that do not support the attribute.

Example:

```
<table>
<tr>
<th> </th>
<th scope="col" >Spring</th> <th scope="col" >Summer</th> <th scope="col"
>Autumn</th> <th scope="col" >Winter</th> </tr>
<tr> <td scope="row" >Betty</td> <td>9-5</td> <td>10-6</td> <td>8-4</td><td>7-
3</td>
</tr>
<tr> <td scope="row" >Wilma</td> <td>10-6</td> <td>10-6</td> <td>9-5</td> <td>9-
5</td>
</tr>
<tr> <td scope="row" >Fred</td> <td>10-6</td> <td>10-6</td> <td>10-6</td> <td>10-
6</td>
</tr>
</table>
```

This table would be displayed as follows:

	Spring	Summer	Autumn	Winter
Betty	9-5	10-6	8-4	7-3
Wilma	10-6	10-6	9-5	9-5
Fred	10-6	10-6	10-6	10-6

Using the "ID" and "Headers" attribute:

(from Section 508 Standards, www.access-board.gov)
The "id" and "headers" attributes requires that every data cell in a table include special attributes for association. Although its usefulness for accessibility may have been diminished as browsers provide support for the "scope" attribute, the "id" and "headers" attributes are still very useful and provide a practical means of providing access in smaller tables.

The following table is much more complicated than the previous example and demonstrates the use of the "id" and "headers" attributes. Both methods provide a means of complying with the requirements for data tables in web pages. The table in this example includes the work schedules for two employees. Each employee has a morning and afternoon work schedule that varies depending on whether the employee is working in the winter or summer months. The "summer" and "winter" columns each span two columns labeled "morning" and "afternoon." Therefore, in each cell identifying the work schedule, the user needs to be told the employee's name (Fred or Wilma), the season (Summer or Winter), and the shift (morning or afternoon).

Example:

```
<table>
<tr>
<th>&amp;nbsp;</th>
<th colspan="2" id="winter" >Winter</th>
<th colspan="2" id="summer" >Summer</th>
</tr>
<tr>
<th>&amp;nbsp;</th>
<th id="am1" >Morning</th>
<th id="pm1" >Afternoon</th>
<th id="am2" >Morning</th>
<th id="pm2" >Afternoon</th>
</tr>
<tr>
<td id="wilma" >Wilma</td>
<td headers="wilma am1 winter" >9-11</td>
<td headers="wilma pm1 winter" >12-6</td>
```

```
<td headers="wilma am2 summer" >7-11</td>
<td headers="wilma pm2 summer" >12-3</td>
</tr>
<tr>
<td id="fred" >Fred</td>
<td headers="fred am1 winter" >10-11</td>
<td headers="fred pm1 winter" >12-6</td>
<td headers="fred am2 summer" >9-11</td>
<td headers="fred pm2 summer" >12-5</td>
</tr>
</table>
```

This table would be displayed as follows:

	Winter		**Summer**	
	Morning	**Afternoon**	**Morning**	**Afternoon**
Wilma	9-11	12-6	7-11	12-3
Fred	10-11	12-6	9-11	12-5

Summary Attribute:

Not all assistive technology products will interpret data tables using these options in the same manner, but data tables encoded in an accessible format will provide a greater opportunity for those needing assistive technology to gain access to on-line information. Using the **summary** attribute can also assist screen-reader users in understanding the layout and content presented in a table. It is important to note that not all assistive technology developers support this attribute at the current time.

Example:

1. Design the table using either the **scope** or **id** and **header** attributes.

2. After the <table-tag, and before the > sign, insert the attribute **summary= "descriptive table summary goes here"**.

3. A good description of the table will include a brief identification of the content displayed in the row, the column, and the cell.

4. If possible, check the table and summary using screen-reader technology.

Evaluation and Repair Tools List

The following tools evaluate a page or a website for potential accessibility concerns. Standards for "accessible" range from the Web Accessibility Initiative (WAI) Web Content Accessibility Guidelines (v.1.0), and the Section 508 Standards of the Rehabilitation Act (effective June 21, 2001). These tools provide information covering how to fix flagged errors as well as provide methods to evaluate and "repair" a website. Bobby and A-Prompt are discussed in the next section in detail.

A-Prompt

http://aprompt.snow.utoronto.ca/

A-Prompt is a free program developed between the Adaptive Technology Resource Centre (ATRC) at the University of Toronto and the TRACE Center at the University of Wisconsin. A-Prompt uses the WAI Guidelines and Section 508 Standards to evaluate HTML code, suggests compliant HTML code syntax, and allows the developer to edit and replace code.

Bobby

http://www/cast.org/bobby/

Bobby is a web evaluation tool to help Web page authors identify and repair significant barriers to access by individuals with disabilities. Enter the URL of the page that Bobby is to examine and click Submit. This dialog will only test one page at a time. You may download a version that will test an entire site ($99 for single user license).

InFocus

http://www.ssbtechnologies.com/

InSight has been folded into the new version of the software from SSB Technologies and packaged as **InFocus**. Together these tools perform automated diagnostics to determine web site accessibility and retrofit web sites automatically making them technically accessible. These programs are designed for persons familiar with authoring web pages and provide tools to retrofit large websites.

VisCheck

http://www.vischeck.com

VisCheck is a free, on-line program that will evaluate your web page for color contrast and simulate what your site will look like to an individual with colorblindness. The download version requires the Photoshop software.

Using Bobby to Evaluate a Website

Introduction

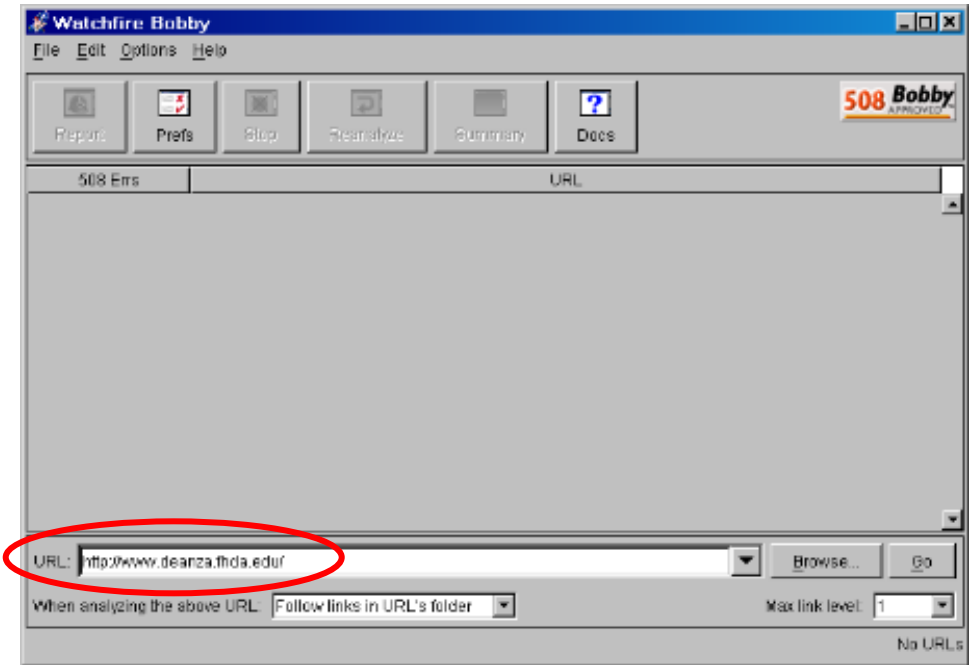
Bobby is a web evaluation tool developed at the Center for Applied Special Technology (CAST) and was acquired by WatchFire in July, 2002. Bobby is a tool that will evaluate a web page, or website, and identify potential accessibility concerns that may impair the ability of an individual to access the website. There are two version of Bobby; a user may download the Bobby tool (Cost: $99.00) or access the Bobby website to perform an accessibility review (Cost: Free). For extensive reviews or to evaluate pages from behind a firewall, it is recommended to use the download version of Bobby.

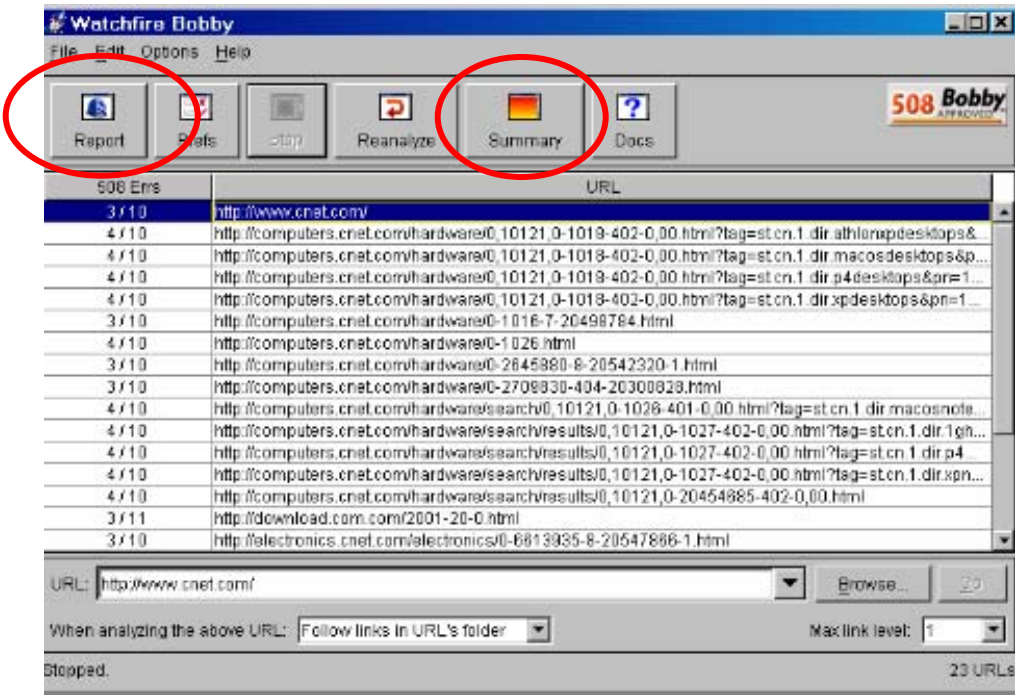
Bobby performs an evaluation of a site based on the Web Content Accessibility Guidelines 1.0 (WCAG) developed by the Web Accessibility Initiative (WAI) and the Section 508 Standards. The advantage of Bobby is the library of information provided through the Bobby interface.

Evaluating a Site

1. Download Bobby from the CAST website and install the program onto the computer. You can start Bobby through the Start menu and going to the Cast folder.

2. In the **URL** field, enter the address of the website that you wish to evaluate, or use the **Browse** function to locate the file on your computer.





3. You can customize how "deep" into a website Bobby will analyze by adjusting the **When analyzing the above URL:** drop down menu. Additionally, you can designate the Priority level that Bobby uses to evaluate the site by selecting the **Prefs** button and choosing the various Priority levels.

Using A-Prompt to Evaluate a Website

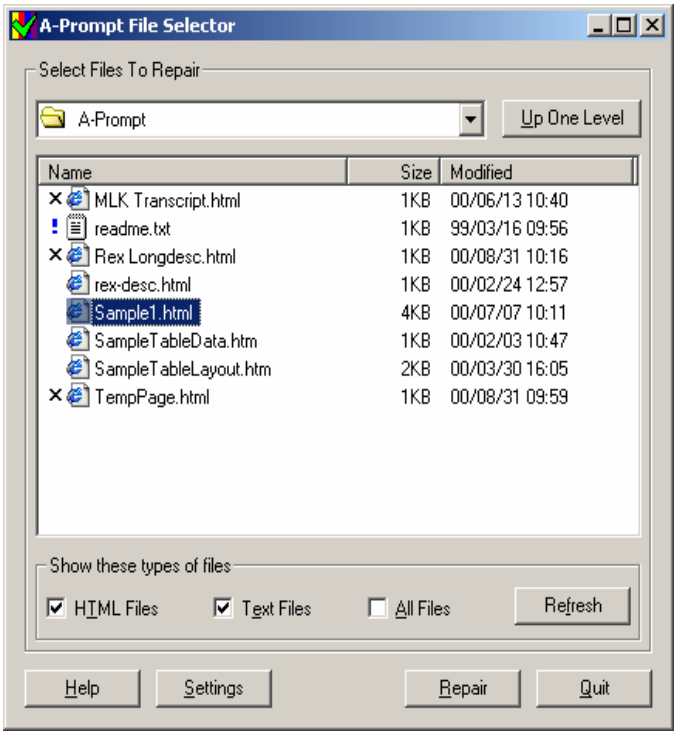
Introduction

A-Prompt (Accessibility-Prompt) is free evaluation tool developed by the Adaptive Technology Resource Centre at the University of Toronto and the TRACE Center at the University of Wisconsin. A-Prompt is designed to assist web authors in designing and developing accessible web page content by evaluating the HTML code against a list of potential accessibility issues. The most recent version of A-Prompt (ver. 1.0.6) contains an evaluation checklist that will compare a website's HTML code structure against the Section 508 Standards of the Rehabilitation Act and the Web Content Accessibility Guidelines developed by the Web Accessibility Initiative.

The A-Prompt website (http://aprompt.snow.utoronto.ca/index.html) contains further information regarding the development of A-Prompt as well as a brief overview of its functions. **You will need WinZip or similar program in order to install A-Prompt.**

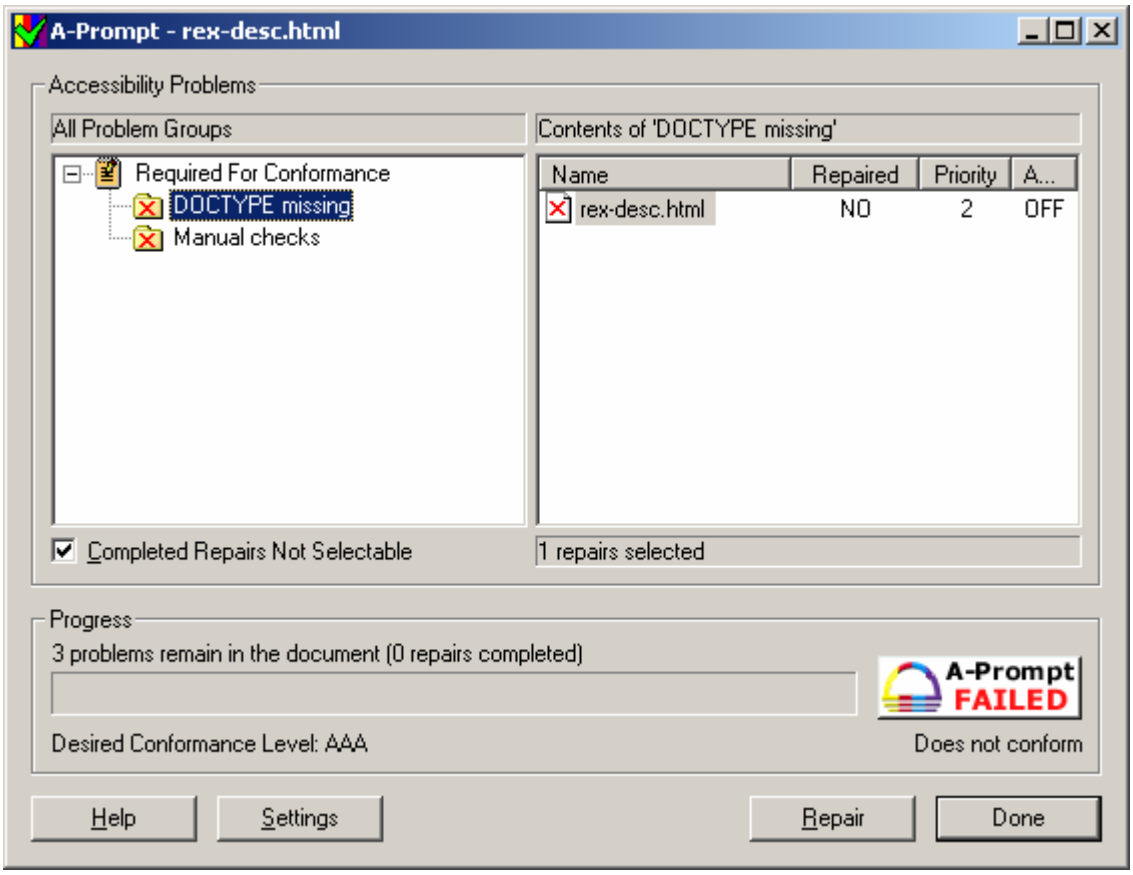
Evaluating a Site

1. A-Prompt begins with the **File Selector** interface. At this level, you can choose the .html files that you want to evaluate or browse to the file folder location your website is saved. To evaluate a web page file, select the appropriate file and click the Repair button.



2. You can choose the conformance level at which the web pages will be evaluated. Click on the **Settings** button, and you can select either the WAI Priority Guidelines or the Section 508 Standards. The **Settings** button is available on all screen views.

3. After selecting the file you wish to evaluate, you will be prompted to select the accessibility issue in the left frame. The right frame displays the specific errors that require attention.



4. In the right frame, select the accessibility problem and click the **Repair** button. This action will alert you to the changes necessary to make the web page accessible and prompt you for the necessary alterations. Once you have completed the accessibility improvements to the files, select the **Done** button. **Remember to save your corrected files after selecting the Done button.**

5. The **Help** button can provide more information as to specific questions or guidance in using the A-Prompt program.

Web Resources

Accessible Web Design Guidelines

Microsoft: http://www.microsoft.com/enable/dev/web/default.htm

WebAIM: http://www.webaim.org/howto/frontpage.php

EASI: http://www.isc.rit.edu/~easi/

A-Prompt

http://aprompt.snow.utoronto.ca/

Bobby

http://bobby.watchfire.com/bobby/html/en/index.jsp

InFocus

http://www.ssbtechnologies.com/

Section 508 Standards

http://www.section508.org

http://www.access-board.gov

Usability.gov

http://www.usability.gov

VisCheck

http://www.vischeck.com

Web Accessibility Initiative (WAI) Guidelines

http://www.w3c.org/WAI/

WebAIM

http://www.webaim.org

WebABLE

http://www.webable.com/

Macromedia

http://www.macromedia.com/macromedia/accessibility/tools/contents.html

http://www.macromedia.com/macromedia/accessibility/