W3C

Techniques for Web Content Accessibility Guidelines

W3C Working Draft 16-Mar-1999

This version:
http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH-19990316/wai-pageauth-tech
Latest version:
http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH/wai-pageauth-tech
Latest public version:
http://www.w3.org/TR/WD-WAI-PAGEAUTH/wai-pageauth-tech
Previous version:
http://www.w3.org/TR/1999/WD-WAI-PAGEAUTH-19990226/wai-pageauth-tech
Related Documents:
Web Content Accessibility Guidelines
List of Checkpoints for the Web Content Accessibility Guidelines
Editors:
Wendy Chisholm <chisholm@trace.wisc.edu>
Gregg Vanderheiden <gv@trace.wisc.edu>
Ian Jacobs <ij@w3.org>

Copyright © 1999 W3C (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

Abstract

This document is a list of techniques that implement the guidelines described in "Web Content Accessibility Guidelines". This document includes primarily techniques that Web content developers may use to implement the guidelines, but also refers to some CSS techniques as well.

While Web Content Accessibility Guidelines strives to be a stable document (as a W3C Recommendation), the current document will undoubtedly evolve as technologies change and content developers discover more effective techniques for designing accessible pages.

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

Status of this document

This is a W3C Working Draft for review by W3C members and other interested parties. It is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by, or the consensus of, either W3C or members of the WAI GL Working Group.

This document has been produced as part of the W3C Web Accessibility Initiative. The goal of the WAI Page Author Guidelines Working Group is discussed in the Working Group charter.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Please send detailed comments on this document to w3c-wai-gl@w3.org.

Available formats

This document is available in the following formats:

HTML:
http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH-19990316/wai-pageauth-tech.html
A plain text file:
http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH-19990316/wai-pageauth-tech.txt,
HTML as a gzip'ed tar file:
http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH-19990316/wai-pageauth.tgz,
HTML as a zip file (this is a '.zip' file not an '.exe'):
http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH-19990316/wai-pageauth.zip,
A PostScript file:
http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH-19990316/wai-pageauth-tech.ps,
A PDF file:
http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH-19990316/wai-pageauth-tech.pdf.

In case of a discrepancy between the various formats of the specification, http://www.w3.org/WAI/GL/WD-WAI-PAGEAUTH-19990316/wai-pageauth-tech.html is considered the definitive version.

Table of Contents

1 Strategies used in this document 4
2 Accessibility Themes for HTML 4
2.1 Structure vs. Presentation 4
2.2 Text equivalents and descriptions 7
2.3 Alternative pages 9
2.4 Keyboard access 10
2.5 Navigation 11
2.6 Comprehension 12
2.7 Content negotiation 13
2.8 Automatic page refresh 14
2.9 Other topics 15
2.10 Validation 15
2.11 Browser Support 15
3 HTML topics 16
3.1 Document structure and metadata 16
3.2 Language information 17
3.3 Text markup 18
3.4 Lists 19
3.5 Tables 21
3.6 Links 26
3.7 Images and image maps 27
3.8 Applets and other objects 33
3.9 Audio and video 35
3.10 Style and style sheets 37
3.11 Frames 42
3.12 Forms 46
3.13 Scripts 48
4 Acknowledgments 49
5 Reference specifications 51
6 Services 51
7 Checkpoint Map 52
8 Index of HTML elements and attributes 57
8.1 Elements 57
8.2 Attributes 60

Priorities

Each checkpoint has priority level assigned by the Working Group based on the checkpoint's impact on accessibility.

[Priority 1] or [Priority One]
: 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] or [Priority Two]
: 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] or [Priority Three]
: 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.

Some checkpoints specify a priority level that may change under certain (indicated) conditions.

The checkpoints in this document are numbered to match their numbering in Web Content Accessibility Guidelines.

---

1 Strategies used in this document

2 Accessibility Themes for HTML

The following sections discuss some accessibility that Web content developers should keep in mind as they design HTML documents.

2.1 Structure vs. Presentation

When designing a document or series of documents, content developers should strive to identify the desired structure for their documents without thinking about how the documents will be presented to the user. Distinguishing the structure of a document from how the content is presented offers a number of advantages, including improved accessibility, manageability, and scalability.

Identifying what is structure and what is presentation may be a challenging task at times that content developers must develop a mindset for recognizing. For instance, many content developers consider that a horizontal rule (the HR element) communicates a structural division. This may be true for sighted users, but to unsighted users or users without graphical browsers, a rule has next to no meaning (One might "guess" that an HR element implies a structural division, but without

other information, there is no guarantee.) Content developers should use the HTML 4.0 header elements (H1-H6) to identify new sections. These may be *complemented* by visual or other cues such as horizontal rules, but should not be replaced by them.

The inverse holds as well: content developers should not use structural elements to achieve presentation effects. For instance, even though the BLOCKQUOTE element may cause indented text in some browsers, that is not its meaning, only a presentation side-effect. BLOCKQUOTE elements used for indentation confuse users and search robots alike, who expect the element to be used to mark up block quotations.

The next two sections identify (by theme) precisely those elements and attributes considered structural (and some of their uses) and those that are considered to control presentation. The section on style and style sheets [p. 37] discusses how to use CSS to accomplish the same tasks as the HTML presentation elements and attributes.

Elements and attributes that are deprecated in HTML 4.0 ([HTML40] [p. 50]) appear in red and followed by an asterisk (*) in this document. Most presentation elements have been deprecated in HTML 4.0.

[Checkpoint 13.2] If W3C technologies are used, avoid deprecated language features. [Priority 2]

Structural elements and attributes

Document head and body	HTML [p. 58] , HEAD [p. 58] , BODY [p. 58]
Data about the document	TITLE [p. 60] , META [p. 59] , ADDRESS [p. 57]
Attributes	alt, title, longdesc
Chapters, sections, etc.	H1 [p. 58] -H6
Author-defined structures	DIV [p. 58] , SPAN [p. 60]
Attributes	class, id
Language, writing direction	BDO [p. 57]
Attributes	lang, hreflang, dir
Creating paragraphs	EM [p. 58] , STRONG [p. 60]

Acronyms and abbreviations	ACRONYM [p. 57] , ABBR [p. 57]
Subscripts and superscripts	SUB [p. 60] , SUP [p. 60]
Inserted and deleted text	INS [p. 59] , DEL [p. 58]
Attributes	cite, datetime
Quotations	BLOCKQUOTE [p. 58] , Q [p. 59]
Attributes	cite
Identifying chunks of text	DFN [p. 58] , CODE [p. 58] , SAMP [p. 59] , KBD [p. 59] , VAR [p. 60] , CITE [p. 58]
Lists	UL [p. 60] , OL [p. 59] , LI [p. 59] , DL [p. 58] , DT [p. 58] , DD [p. 58] , DIR* [p. 58] ,
Attributes	start*, value*
Tables	TABLE [p. 60] , CAPTION [p. 58] , THEAD [p. 60] , TFOOT [p. 60] , TBODY [p. 60] , COLGROUP [p. 58] , COL [p. 58] , TR [p. 60] , TH [p. 60] , TD [p. 60]
Attributes	abbr, axis, headers, scope, rowspan, span, summary
Links	LINK [p. 59] , IMG [p. 59] , OBJECT [p. 59] , PARAM [p. 59] , APPLET* [p. 57] , MAP [p. 59] , AREA [p. 57]
Attributes	shape, ismap, coords
Forms and keyboard control	FORM [p. 58] , INPUT [p. 59] , BUTTON [p. 58] , SELECT [p. 60] , OPTGROUP [p. 59] , OPTION [p. 59] , TEXTAREA [p. 60] , LABEL [p. 59] , FIELDSET [p. 58] , LEGEND [p. 59] , ISINDEX* [p. 59]
Attributes	tabindex, accesskey, label, for
Scripts	SCRIPT [p. 60] , NOSCRIPT [p. 59]
Attributes	onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onload, onload, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onreset, onselect, onsubmit, onunload

Presentation elements and attributes

Note 1. Not all attributes are listed in this section. Only those deemed pertinent to accessibility have been listed.

Note 2. While the STYLE element is presentational, it may be used to *enhance* the presentation of structural elements. The other presentation elements are often used *instead of* structural elements.

Style sheets	PRE [p. 59] , CENTER* [p. 58] , BR [p. 58] , STYLE [p. 60]
Attributes	align*, valign*, clear*, nowrap*, char, charoff , style
Spacing	
Attributes	hspace*, vspace*, cellpadding, cellspacing, compact*, type*
Text style and fonts	TT [p. 60] , I [p. 59] , B [p. 57] , BIG [p. 57] , SMALL [p. 60] , STRIKE* [p. 60] , S* [p. 59] , U* [p. 60] , FONT* [p. 58] , BASEFONT* [p. 57]
Attributes	face*, size*
Colors	
Attributes	background*, bgcolor*, color*, text*, link*, alink*, vlink*
Rules and borders	HR [p. 58]
Attributes	border, noshade*, rules, size (deprecated according to element).
Frames	FRAMESET [p. 58] , FRAME [p. 58] , IFRAME [p. 59] , NOFRAMES [p. 59]
Attributes	target, marginheight, marginwidth, frame, frameborder, noresize, rows, cols, scrolling

2.2 Text equivalents and descriptions

Text is considered accessible to almost all users since it may be handled by screen readers, non-visual browsers, braille readers, etc. It is good practice, as you design a document containing non-textual information (images, graphics, applets, sounds, etc.) to think about supplementing that information with textual equivalents wherever possible.

There are several types of textual supplements to consider:

Text equivalents
A text equivalent (or alternative text) describes the function or purpose of content. A text equivalent should not describe visual appearance or how

something sounds. For example, if an image of a magnifying glass is used for a search button, the alt-text would be "Search" rather than "Magnifying glass".

Text equivalents should be provided for logos, photos, submit buttons, applets, bullets in lists, ascii art, and all of the links within an image map as well as invisible images used to layout a page.

Quicktest! A good test to determine if a text equivalent is useful is to imagine reading the document aloud over the telephone. What would you say upon encountering this image to make the page comprehensible to the listener?

Related checkpoints:

1. Refer also to checkpoint 1.1.
2. Refer also to checkpoint 1.2.
3. For all image map links:
   - Refer also to checkpoint 1.4.
   - Refer also to checkpoint 1.7.
4. Refer also to checkpoint 1.8.
5. Refer also to checkpoint 1.6.
6. Refer also to checkpoint 1.3.
7. Refer also to checkpoint 5.5.
8. [Checkpoint 14.1] Title each frame so that users can keep track of frames by title. [Priority 1]

Long descriptions

In general, a long description should describe visual appearance or how something sounds. Long descriptions are generally stored in external documents (but may be inline if necessary).

1. Refer also to checkpoint 2.1.
2. [Checkpoint 14.2] Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone. [Priority 2]

HTML allows content developers to specify these substitutes in several ways. Here is a summary of attributes used for alt-text and long descriptions and the elements they apply to (by version of HTML):

To implement alt-text, use the "alt" attribute.

In HTML 4.0, applies to: IMG, AREA, INPUT, APPLET. The attribute is mandatory for the IMG and AREA elements.
In HTML 3.2, applies to: IMG, AREA, APPLET.
In HTML 2.0, applies to: IMG.

To implement long descriptions, use the "longdesc" attribute

In HTML 4.0, applies to: FRAME, IFRAME, IMG
In HTML 3.2, applies to: N/A.
In HTML 2.0, applies to: N/A.

Not every user agent supports these attributes. When required to design documents for a version of HTML known not to support one of these attributes, content developers should implement alt-text or descriptions in other ways. Methods include:

- Inline text equivalents or descriptions. For example, include a description of the image immediately after the image.
- Links to descriptions (called description links or "d-links"). Description links should be ordinary links (on the same page or another designed for such links) that designate a document containing a long description. The link text (content of the A element) should be "[D]".

Each of the HTML topics [p. 16] below describes prioritized scenarios for alt-text and descriptions.

Some image formats allow internal text in the data file along with the image information. If an image format supports such text (e.g., Portable Network Graphics, see [PNG] [p. 50]) content developers may also supply information there as well.

2.3 Alternative pages

[Checkpoint 13.4] If, after best efforts, you cannot avoid using a non-W3C technology or any W3C technology in an accessible way, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent information, and is updated as often as the inaccessible (original) page. [Priority 1]

Although it is possible to make most content accessible, it may happen that all or part of a page remains inaccessible. Here are several additional techniques for creating accessible alternatives:

- Allow users to navigate to a separate page that is accessible and maintained with the same frequency as the inaccessible original page. There are several techniques for linking to an accessible alternative page:
  1. Provide links at the top of both the main and alternative pages to allow a user to move back and forth between them.
  2. Use the LINK element to designate alternative documents. Browsers should load the alternative page automatically based on the user’s browser type and preferences.
- Instead of static alternative pages, set up server-side scripts that generate accessible versions of a page on demand.
- [Checkpoint 8.4] Provide an alternative presentation or page when the primary content is dynamic. [Priority 2] . See the examples for Frames [p. 44] and Scripts [p. 48] .
- Otherwise provide a phone number, fax number, e-mail, or postal address where information is available and accessible, preferably 24 hours a day

Example.

User agents that support LINK will load the alternative page for those users whose browsers may be identified as supporting "aural","braille", or "tty" rendering.

```
<HEAD>
<TITLE>Welcome to the Virtual Mall!</TITLE>
<LINK title="Text-only version"
      rel="alternate"
      href="text_only.html"
      media="aural, braille, tty">
</HEAD>
<BODY><P>...</BODY>
```

End example.

2.4 Keyboard access

Not every user has a graphic environment with a mouse or other pointing device. Some users rely on keyboard or voice input to navigate links, activate form controls, etc. Content developers should always ensure that users may interact with a page with devices other than a pointing device. A page designed for keyboard access (in addition to mouse access) will generally be accessible to users with other input devices. What's more, designing a page for keyboard access will usually improve its overall design as well.

Keyboard access to links and form controls may be specified in two ways:

Keyboard shortcuts
: Keyboard shortcuts allow users to combine keystrokes to navigate links or form controls on a page. **Note.** Keyboard shortcuts may be handled differently by different operating systems.

Tabbing order
: Tabbing order describes a (logical) order for navigating from link to link or form control to form control (usually by pressing the "tab" key, hence the name).

Here is a summary of attributes used for keyboard access and tabbing order and the elements they apply to (by version of HTML):

To implement shortcuts, use the "accesskey" attribute.
- In HTML, 4.0 applies to: A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA
- In HTML, 3.2 applies to: N/A.
- In HTML, 2.0 applies to: N/A.

To implement tabbing order, use the "tabindex" attribute
- In HTML, 4.0 applies to: A, AREA, BUTTON, INPUT, OBJECT, SELECT, TEXTAREA
- In HTML, 3.2 applies to: N/A
- In HTML, 2.0 applies to: N/A

In the following example, if the accesskey "C" is activated,"doc.html" is retrieved by the browser:

Example.

```
<A accesskey="C" href="doc.html" hreflang="en"
   title="XYZ company home page">
      XYZ company home page</A>
```

End example.

The next example assigns "U" as the access key. Typing "U" gives focus to the label, which in turn gives focus to the input control, so that the user can input text.

Example.

```
<FORM action="submit" method="post">
<P>
      <LABEL for="user" accesskey="U">name</LABEL>
      <INPUT type="text" id="user">
</FORM>
```

End example.

In the next example, we specify a tabbing order among elements (in order, "field2", "field1", "submit"):

Example.

```
<FORM action="submit" method="post">
<P>
<INPUT tabindex="2" type="text" name="field1">
<INPUT tabindex="1" type="text" name="field2">
<INPUT tabindex="3" type="submit" name="submit">
</FORM>
```

End example.

[Checkpoint 11.1] Ensure that all elements that have their own interface are keyboard operable. [Priority 2] Some elements import objects whose interfaces cannot be controlled through HTML (applets, images, etc.) In such cases, users should ensure that the imported objects themselves provide accessible interfaces.

2.5 Navigation

[Checkpoint 16.3] Create a style of presentation that is consistent across pages. [Priority 3]

A consistent style of presentation on each page allows users to easily find navigation buttons between pages, as well as find the primary content for each page. While this helps make it easier for everyone, it especially benefits people with learning and reading disabilities. Making it easy to predict where the needed information is found on each page will increase the likelihood that it will be found.

Examples of structures that may appear at the same place between pages:

1. navigation bars
2. the primary content of a page
3. advertising

[Checkpoint 15.6] Use navigation mechanisms in a consistent manner. [Priority 3]

A navigation structure is the set of possible paths available for a user to take through the content on your site. Providing navigation bars, site maps, and search features all increase the likelihood that a user will navigate easily to the information that they seek on your site. If your site is highly visual in nature, the structure might be harder to navigate through if the user can't form a mental map of where they are going or where they have been. Therefore, provide a description that will help users discover the mechanisms you have provided.

- [Checkpoint 15.3] Provide navigation bars to highlight and give access to the navigation mechanism. [Priority 3]
- [Checkpoint 15.5] Provide a description of the general layout of the site, the access features provided, and how to use them. [Priority 3]
- [Checkpoint 15.4] Provide a site map or table of contents that makes the structure of a Web site apparent and facilitates navigation. [Priority 3]
- [Checkpoint 15.7] Enable different types of searches for different skill levels and preferences. [Priority 3]
- [Checkpoint 15.2] Provide metadata to add semantic information to pages and sites. [Priority 2] . See [RDF] [p. 50] .

  Content developers should use link types (reference to HTML 4.0, 6.12) with LINK so that user agents may use build navigation structures. (Provide example with "next", "prev", "content", and "start").

[Checkpoint 15.8] Place distinguishing information at the beginning of headings, paragraphs, lists, etc. [Priority 3]

See also the section on links [p. 26] .

2.6 Comprehension

1. [Checkpoint 16.1] Use language that is clear and simple, yet appropriate for the site's content. [Priority 1] Provide a simplified text equivalent as an alternative to the primary text. This may be helpful to non-readers and people who have difficulty reading. Non-readers can hear the simplified text using speech output technology. Individuals with reading difficulties may be able to read the simplified text. Because of the challenges of maintaining separate texts, it is preferable to simplify only the main text and not provide a simplified text equivalent.
2. [Checkpoint 16.2] Provide icons or graphics (with text equivalents and descriptions) where they facilitate comprehension of the page. [Priority 3] Visual,

non-text equivalents may include, for example graphics, animations, or videos. These are especially helpful for non-readers who can perceive visual presentations. For example, sighted deaf non-readers may benefit from video equivalents in manual communication (sign language). Non-readers, whether they have disability or not, may also benefit from highly graphical equivalents.

Non-visual, non-text equivalents are very diverse. Among the most common are prerecorded audio of music, spoken language, or sound effects. Such equivalents would be especially important for non-readers who can perceive audio presentations. Presentations in the audio medium of synthesized speech and the tactile medium of braille are usually derived from text or text equivalents so usually require no additional work from the developer.

Follow these writing suggestions:

-
- Strive for clear and accurate headings and link descriptions. Scrutinize every heading, outline, and menu to see if the crucial words mean exactly what is intended, and if there are more common words that would convey the same meaning.
- State the topic of the sentence or paragraph at the beginning of the sentence or paragraph.
- Limit each paragraph to one main idea.
- Avoid idiomatic language, technical jargon, and other unfamiliar vocabulary and expressions.
- Avoid specialized meanings of familiar vocabulary, unless explanations are provided.
- Avoid the passive voice.
- Avoid complex sentence structures.
- Make link phrases terse and meaningful when read out of context.

Because people tend to scan rather than read Web pages, the quality of headings is particularly important. Good headings will at least get people to a section that has the information they need. From there they can go to a dictionary or even print out a section and ask for help.

2.7 Content negotiation

[Checkpoint 13.3] Provide information so that users may receive documents according to their preferences (e.g., language, content type, etc.) [Priority 3]

1. Instead of including links such as "Here is the French version of this document", use content negotiation so that the French version is served to clients preferring
2. If not possible to use content negotiation, in HTML use "type" and "hreflang".

2.8 Automatic page refresh

Content developers sometimes create pages that refresh or change without the user requesting the refresh. This automatic refresh can be very disorienting to some users. There are two types of refresh mechanisms commonly used.

[Checkpoint 9.1] Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages [Priority 1] This type of refresh changes the user's page at regular intervals. This might be accomplished by the following HTML markup:

Deprecated example.

```
<META http-equiv="refresh" content="60">
<BODY>
<P>...Information...
</BODY>
```

Content developers should **not** use this technique to simulate "push" technology. Developers cannot predict how must time a user will require to read a page; premature refresh can disorient users. Content developers should avoid periodic refresh and allow users to choose when they want the latest information.

[Checkpoint 9.4] 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. [Priority 2] This type of forward redirects the user from one page to another generally after a timeout. The following HTML markup is frequently used to achieve this effect:

Deprecated example.

```
<META http-equiv="refresh" content="5;
         http://www.acme.com/newpage">
<BODY>
<P>If your browser supports Refresh,
you'll be transported to our
<A HREF=http://www.acme.com/newpage<new site</A>
in 5 seconds, otherwise, select the link manually.
</BODY>
```

However, users should **not** redirect users with this markup since is non-standard, it disorients users, and it can disrupt a browser's history of visited pages. Instead, in order of preference, users should:

1. Use server-side redirects.
2. Replace the page that would be redirected with a static page containing a normal link to the new page.

Note. Both checkpoint 9.1 and checkpoint 9.4 address problems posed by legacy user agents. Newer user agents should disable refresh and substitute a link to new information at the top of the page.

2.9 Other topics

[Checkpoint 15.9] Facilitate off-line browsing by creating a single downloadable file for documents that exist as a series of separate pages. [Priority 3]

For example, Indicate which is the first page of the document and which page follows the current one. (e.g., by using the LINK element).

2.10 Validation

1. Use an automated accessibility validation tool such as Bobby (refer to [BOBBY] [p. 51]).
2. Use an HTML validation service such as the W3C HTML Validation Service (refer to [HTMLVAL] [p. 51]).
3. Use a style sheets validation service such as the W3C CSS Validation Service (refer to [CSSVAL] [p. 51]).
4. Test your pages with a text-only browser such as Lynx ([LYNX] [p. 52]) or a Lynx emulator such as Lynx Viewer ([LYNXVIEW] [p. 52]) or Lynx-me ([LYNXME] [p. 52]).
5. Use multiple graphic browsers, with:
   - sounds and graphics loaded,
   - graphics not loaded,
   - sounds not loaded,
   - no mouse,
   - frames, scripts, style sheets, and applets not loaded
6. Use several browsers, old and new.
7. Use a self-voicing browser, a screen reader, magnification software, a small display, etc. Self-voicing browsers include "pwwebspeak".
8. Use spell and grammar checkers. A person reading a page with a speech synthesizer may not be able to decipher the synthesizer's best guess for a word with a spelling error. Eliminating grammar problems increases comprehension.

If, after completing these tests and adjusting your design accordingly, you find that your page is still not accessible, you must create an alternative page [p. 9] that is accessible.

2.11 Browser Support

1. [Checkpoint 13.1] If W3C technologies are used, use the latest W3C specification where it is supported. [Priority 2] .
2. In general, user agents ignore HTML attributes they don't support and they render the content of unsupported elements.

3 HTML topics

The following sections list some techniques for using HTML and CSS to design accessible documents and some techniques for avoiding HTML accessibility traps. The sections are organized by topic (and mirror the organization of the HTML 4.0 specification, [HTML40] [p. 50]).

3.1 Document structure and metadata

[Checkpoint 5.4] Create documents that validate to published formal grammars. [Priority 2]

As discussed above, content developers should use structural markup wherever possible (and use it as intended by the authors of W3C specifications). Structural elements promote consistency in documents and supply information to other tools (e.g., indexing tools, search engines, programs that extract tables to databases, navigation tools that use header elements, and automatic translation software that translates text from one language into another.

Metadata

Some structural elements provide information about the document itself. This is called "metadata" about the document (Metadata is information about data). Well-crafted metadata can provide important orientation information to users. HTML elements that provide useful information about a document include:

- TITLE: The document title. Note that the TITLE element (one time only in a document) is different from the "title" attribute, which applies to almost every HTML 4.0 element. This document makes use of the "title" attribute for many occasions that require advisory titles. Content developers should use the "title" attribute in accordance with the HTML 4.0 specification. For example, "title" should be used with links to provide information about the target of the link.
- ADDRESS: Can be used to provide information about the creator of the page.
- LINK: Can be used to indicate alternative documents (different structure, different language, different target device, etc.).
- The META element can be used to describe metadata about a document. Please refer to the section on automatic page refresh [p. 14] for information on why META should not be used to redirect pages.

Section headers

Sections should be introduced with the HTML header elements (H1-H6). Other markup may complement these elements to improve presentation (e.g., the HR element to create a horizontal dividing line), but visual presentation is not sufficient to identify document sections.

[Checkpoint 5.1] Use header elements to convey logical structure and use them according to specification. [Priority 2] Since some users skim through a document by navigating its headings, it is important to use them appropriately to convey document structure. Users should order heading elements properly. For example, in HTML, H2 elements should follow H1 elements, H3 elements should follow H2 elements, etc. Content developers should not "skip" levels (e.g., H1 directly to H3). Do not use headings to create font effects; use style sheets [p. 37] .

Note that in HTML, heading elements (H1 - H6) do not contain entire logical sections in their content. They should be used to begin a section of a document. The following HTML markup shows how to create a true document "section" and control its appearance with style sheets:

Example.

```
<HEAD>
<TITLE>Cooking techniques</TITLE>
<STYLE type="text/css">
   /* Indent the entire section */
   DIV.section2 { margin-left: 5% }
</STYLE>
</HEAD>
<BODY>
<H1>Cooking techniques</H1>
... some text here ...
<DIV class="section2">
<H2>Cooking with oil</H2>
... text of the section ...
</DIV>

<DIV class="section2">
<H2>Cooking with butter</H2>
... text of the next section ...
</DIV>
```

End example.

3.2 Language information

[Checkpoint 6.1] Clearly identify changes in the natural language of a document's text. [Priority 1] If you use a number of different languages on a page, make sure that any changes in language are clearly identified by using the "lang" attribute:

Example.

```
<P>And with a certain <SPAN lang="fr">je ne sais quoi</SPAN>,
she entered both the room, and his life, forever. <Q>My name
is Natasha,</Q> she said. <Q lang="it">Piacere,</Q>
he replied in impeccable Italian, locking the door.
```

End example.

[Checkpoint 6.3] Identify the primary natural language of a document. [Priority 3]

3.3 Text markup

As mentioned above, structural elements add information to a page that may be used by browsers, search engines, and other software. Content developers are encouraged to use structural elements and attributes [p. 5] whenever possible. Below we discuss how to further improve accessibility by careful use of attributes with these elements.

Emphasis

The proper HTML elements should be used to mark up emphasis: EM and STRONG. The B and I elements should not be used; they are used to create a visual presentation effect. The EM and STRONG elements were designed to indicate structural emphasis that may be rendered in a variety of ways (font style changes, speech inflection changes, etc.)

Acronyms and abbreviations

[Checkpoint 6.2] Specify the expansion of abbreviations and acronyms. [Priority 2 for the first occurrence of the acronym or abbreviation in a given document, Priority 3 thereafter.] Mark up abbreviations and acronyms with ABBR and ACRONYM and use "title" to indicate the expansion:

Example.

```
<P>Welcome to the <ACRONYM title="World Wide Web">WWW</ACRONYM>!
```

End example.

Quotations

[Checkpoint 5.3] Mark up quotations. Do not use quotation markup for formatting effects such as indentation. [Priority 2] See the language example [p. 17] above for an illustration of the Q element.

Example.

```
<BLOCKQUOTE cite="http://www.shakespeare.com/loveslabourlost">
  <P>Remuneration! O! that's the Latin word for three farthings.
     --- William Shakespeare (Love's Labor Lost).
  </P>
</BLOCKQUOTE>
```

End example.

Text markup rather than images

[Checkpoint 5.6] When an appropriate markup language exists, use markup rather than images to convey information. [Priority 2]

3.4 Lists

[Checkpoint 5.2] Mark up lists and list items properly. [Priority 2] The HTML list elements DL, UL, and OL (available in HTML 3.2 and HTML 4.0) should only be used to create lists, not for formatting effects such as indentation.

When possible, use ordered (numbered) lists to help navigation. Non-visual users often "get lost" in lists, especially those with several layers of embedding and those that do not indicate the specific level of indentation for each item. In addition, it is often difficult for non-visual users to know where the list itself begins and ends and where each list item starts. Finally, if a list entry wraps to the next line, it may appear to be two separate items in the list.

Example.

Instead of nesting bulleted lists like this:

- writing tools
  - pens
    - highlighters
      - red
      - green
      - blue
    - ball-point
      - green
      - purple
      - mauve
  - pencils
    - soft lead
    - #2 lead
- erasers

which might be read by a screen reader (stripping bullets and indentation) as:

```
writing tools
pens
highlighters
red
green
blue
ball-point
green
purple
mauve
```

```
pencils
#2 lead
soft lead
erasers
```

Use ordered lists (or make sure to put distinguishing information at the beginning of each list item):

1. writing tools
    1. pens
        1. highlighters
            1. red
            2. green
            3. blue
        2. ball-point
            1. green
            2. purple
            3. mauve
    1. pencils
        1. soft lead
        2. #2 lead
2. erasers

End example.

Use style sheets to change list bullets

To change the "bullet" style of unordered list items, use style sheets. This way, if images are not loaded, the browser will draw a default bullet.

Example.

```
<HEAD>
<TITLE>Using style sheets to change bullets</TITLE>
<STYLE type="text/css">
   UL { list-style: url(star.gif) }
</STYLE>
</HEAD>
<BODY>
<UL>
   <LI>Audrey
   <LI>Laurie
   <LI>Alice
</UL>
```

End example.

Avoid using images as bullets in definition lists. However, if this method is used, be sure to provide alt-text [p. 7] for the images.

Deprecated example.

```
<DL>
   <DD><IMG src="star.gif" alt="Item">Audrey
   <DD><IMG src="star.gif" alt="Item">Laurie
   <DD><IMG src="star.gif" alt="Item">Alice
</DL>
```

Content developers should avoid list styles where bullets provide additional (visual) information. However, if this is done, be sure to provide alt-text describing meaning of the bullet:

Deprecated example.

```
<DL>
<DD><IMG src="red.gif" alt="New:">Roth IRA</DD>
<DD><IMG src="yellow.gif" alt="Old:">401(k)</DD>
</DL>
```

Here is a better way to change list bullet styles (using style sheets). To further ensure that users understand differences between list items indicated visually, content developers should provide a label before or after the list item phrase:

Example.

```
<HEAD>
<TITLE>Bullet styles example</TITLE>
<STYLE type="text/css">
   .newtxt { font-weight: bold;
             color: red;
             background-color: yellow }
   .newbullet { list-style : url(yellow.gif) }
</STYLE>
</HEAD>
<BODY>
<UL>
   <LI class="newbullet">Roth IRA <SPAN class="newtext">New</SPAN></LI>
   <LI> 401(k)</LI>
</UL>
</BODY>
```

End example.

3.5 Tables

Content developers may make tables more accessible in a number of ways:

- Provide a caption via the CAPTION element.
- [Checkpoint 7.5] Provide summaries for tables. [Priority 3] Summaries are especially useful for non-visual readers.
- [Checkpoint 7.1] For data tables, identify headers for rows and columns. [Priority 1] Future browsers and assistive technologies will be able to automatically translate tables into linear sequences if data is labeled appropriately.

- [Checkpoint 7.2] For data tables that have more than one row and/or more than one column of header cells, use markup to associate data cells and header cells. [Priority 1] Identify structural groups of rows (THEAD for repeated table headers, TFOOT for repeated table footers, and TBODY for other groups of rows) and groups of columns (COLGROUP and COL). Label table elements with the "scope", "headers", and "axis" attributes so that future browsers and assistive technologies will be able to select data from a table by filtering on categories. This markup will also help browsers translate tables into linear sequences automatically (also called table "serialization"). A linear sequence is usually generated by reading a row left to right and proceeding each cell with the label of its column.
- [Checkpoint 7.6] Provide abbreviations for header labels. [Priority 3] Provide terse substitutes for header labels with the "abbr" attribute on TH. These will be particularly useful for future speaking technologies that can read row and column labels for each cell. Abbreviations cut down on repetition and reading time.
- [Checkpoint 7.4] If a table is used for layout, do not use any structural markup for the purpose of visual formatting. [Priority 2]

Most of the above elements and attributes are only available in HTML 4.0.

This markup will allow accessible browsers and other user agents to restructure tables for non-visual media.

For information about table headers, see the table header algorithm and discussion in the HTML 4.0 Recommendation ([HTML40] [p. 50] , section 11.4.3).

The following example shows how to associate data cells with their corresponding headers by means of the "headers" attribute. The "headers" attribute specifies a list of header cells (row and column labels) associated with the current data cell. This requires each header cell to have an "id" attribute.

Example.

```
<TABLE border="1"
       summary="This table charts the number of
                cups of coffee consumed by each senator,
                the type of coffee (decaf or regular),
                and whether taken with sugar.">
  <CAPTION>Cups of coffee consumed by each senator</CAPTION>
  <TR>
      <TH id="t1">Name</TH>
      <TH id="t2">Cups</TH>
      <TH id="t3" abbr="Type">Type of  Coffee</TH>
      <TH id="t4">Sugar?</TH>
  <TR>
      <TD headers="t1">T. Sexton</TD>
      <TD headers="t2">10</TD>
      <TD headers="t3">Espresso</TD>
      <TD headers="t4">No</TD>
  <TR>
      <TD headers="t1">J. Dinnen</TD>
```

```
       <TD headers="t2">5</TD>
       <TD headers="t3">Decaf</TD>
      <TD headers="t4">Yes</TD>
</TABLE>
```

End example.

A speech synthesizer might render this tables as follows:

```
Caption: Cups of coffee consumed by each senator
Summary: This table charts the number of cups of coffee
         consumed by each senator, the type of coffee
        (decaf or regular), and whether taken with sugar.
Name: T. Sexton, Cups: 10, Type: Espresso, Sugar: No
Name: J. Dinnen, Cups: 5, Type: Decaf, Sugar: Yes
```

A visual user agent might render this table as follows:

Cups of coffee consumed by each senator

Name	Cups	Type of Coffee	Sugar?
T. Sexton	10	Espresso	No
J. Dinnen	5	Decaf	Yes

[D]

The next example associates the same header and data cells as before, but this time uses the "scope" attribute rather than"headers." "Scope" must have one of the following values: row, col, rowgroup or colgroup. Scope specifies the set of data cells to be associated with the current header cell. This method is particularly useful for simple tables. It should be noted that the spoken rendering of this table would be identical to that of the previous example. A choice between the "headers" and "scope" attributes is dependent on the complexity of the table. It does not affect the output so long as the relationships between header and data cells are made clear in the markup.

Example.

```
<TABLE border="1"
       summary="This table charts ...">
   <CAPTION>Cups of coffee consumed by each senator</CAPTION>
   <TR>
        <TH scope="col">Name</TH>
        <TH scope="col">Cups</TH>
        <TH scope="col" abbr="Type">Type of Coffee</TH>
        <TH scope="col">Sugar?</TH>
  <TR>
        <TD>T. Sexton</TD>  <TD>10</TD>
        <TD>Espresso</TD>   <TD>No</TD>
  <TR>
        <TD>J. Dinnen</TD>  <TD>5</TD>
        <TD>Decaf</TD>       <TD>Yes</TD>
</TABLE>
```

End example.

The following example shows how to create categories within a table using the "axis" attribute.

Example.

```
<TABLE border="1">
  <CAPTION>Travel Expense Report</CAPTION>
  <TR>
       <TH></TH>
       <TH id="a2" axis="expenses">Meals
       <TH id="a3" axis="expenses">Hotels
       <TH id="a4" axis="expenses">Transport
       <TD>subtotals</TD>
  <TR>
       <TH id="a6" axis="location">San Jose
       <TH> <TH> <TH> <TD>
  <TR>
      <TD id="a7" axis="date">25-Aug-97
      <TD headers="a6 a7 a2">37.74
      <TD headers="a6 a7 a3">112.00
      <TD headers="a6 a7 a4">45.00
      <TD>
  <TR>
      <TD id="a8" axis="date">26-Aug-97
      <TD headers="a6 a8 a2">27.28
      <TD headers="a6 a8 a3">112.00
      <TD headers="a6 a8 a4">45.00
      <TD>
  <TR>
      <TD>subtotals
      <TD>65.02
      <TD>224.00
      <TD>90.00
      <TD>379.02
  <TR>
      <TH id="a10" axis="location">Seattle
      <TH> <TH> <TH> <TD>
  <TR>
      <TD id="a11" axis="date">27-Aug-97
      <TD headers="a10 a11 a2">96.25
      <TD headers="a10 a11 a3">109.00
      <TD headers="a10 a11 a4">36.00
      <TD>
  <TR>
      <TD id="a12" axis="date">28-Aug-97
      <TD headers="a10 a12 a2">35.00
      <TD headers="a10 a12 a3">109.00
      <TD headers="a10 a12 a4">36.00
      <TD>
  <TR>
      <TD>subtotals
      <TD>131.25
      <TD>218.00
      <TD>72.00
      <TD>421.25
```

```
    <TR>
        <TH>Totals
        <TD>196.27
        <TD>442.00
        <TD>162.00
        <TD>800.27
</TABLE>
```

End example.

This table lists travel expenses at two locations: San Jose and Seattle, by date, and category (meals, hotels, and transport). The following image shows how a visual user agent might render it.

Travel Expense Report

	Meals	**Hotels**	**Transport**	subtotals
San Jose				
25-Aug-97	37.74	112.00	45.00	
26-Aug-97	27.28	112.00	45.00	
subtotals	65.02	224.00	90.00	379.02
Seattle				
27-Aug-97	96.25	109.00	36.00	
28-Aug-97	35.00	109.00	36.00	
subtotals	131.25	218.00	72.00	421.25
Totals	196.27	442.00	162.00	**800.27**

[D]

Wrapped text in tables

One source of problems for screen readers that do not interpret the source HTML is wrapped text in table cells. They read across the page, reading sentences on the same row from different columns as one sentence.

For example, if a table is rendered like this on the screen:

```
There is a 30% chance of              Classes at the University of Wisconsin
rain showers this morning, but they   will resume on September 3rd.
should stop before the weekend.
```

This might be read by a screen reader as:

```
There is a 30% chance of Classes at the University of Wisconsin
rain showers this morning, but they will resume on September 3rd.
should stop before the weekend.
```

Screen readers that read the source HTML will recognize the structure of each cell, but for older screen readers, content developers should minimize the risk of word wrapping by limiting the amount of text in each cell. Also, the longest chunks of text should all be in the last column (rightmost for left-to-right tables). This way, if they wrap, they will still be read coherently.

Content developers should test tables for wrapping with a browser window dimension of "640x480".

Quicktest! To get a better understanding of how a screen reader would read a table, run a piece of paper down the page and read your table line by line.

Backwards compatibility issues for tables

Rows of a TFOOT element will appear before the BODY of the document in an HTML3.2 browser.

Avoid tables for layout

[Checkpoint 7.3] Avoid using tables for layout. [Priority 2]

3.6 Links

Users who are blind often jump from link to link when skimming a page or looking for information. When they do this, only the text of the link (the "link text") is read.

"Auditory users," people who are **blind**, **have difficulty seeing**, or who are **using devices with small or no displays** are unable to scan the page quickly with their eyes and often use a list of links to get an overview of a page or to quickly find a link. When links are not descriptive enough, do not make sense when read out of context, or are not unique, the auditory user must stop to read the text surrounding each link to identify it.

[Checkpoint 15.1] Make link phrases terse yet meaningful when read on their own or in succession. [Priority 2] Therefore, it is important that link text make sense when read without surrounding text. For example, content developers should not use "click here" as link text several times on the same page; this requires a user browsing the page with a screen reader to step through each link and read the surrounding text to determine the purpose of the link. Instead, link text should carry sufficient information, as in"download this document in ASCII text," "view the full version in HTML," or "for the text version select this link."

When an image is used as the content of a link, specify alt-text [p. 7] for the image that makes sense in context.

Quicktest! To choose alt-text in this case, think of what you would say in words rather than an image in this context.

Example.

```
<A href="routes.html">
   <IMG src="topo.html"
        alt="Current routes at Boulders Climbing Gym">
</A>
```

End example.

If more than one link on a page shares the same link text, all those links should point to the same resource. Such consistency will help page design as well as accessibility. Link text should be terse but complete. [p. 26]

"Navigation bars" (sets of links that appear on every page in a site) are usually the first thing someone encounters on a page. For speech users, this means taking the time to read through x number of links on every page before reaching the unique content of a page. Therefore, grouping links will allow a user with a user agent that can navigate by elements, to jump over the group. This is similar to how people with vision skip reading the links when they see the same set on each page. Since this type of mechanism is not available today, providing a link that skips over the links, or using a tabindex at a link just before the content begins are strategies that work today.

Example.

```
<TITLE>How to use our site</TITLE>

<BODY>
  <SPAN class="nav">
    <a href="home.html">Home</a>
    <a href="search.html">Search</a>
    <a href="new.html">New and highlighted</a>
    <a href="sitemap.html">Site map</a>
  </SPAN>
  <H1><a name="how" tabindex=1>How to use our site</a></H1>
<!-- content of page -->
</BODY>
```

End example.

[Checkpoint 15.10] Group related links, identify the group (for user agents), and, until user agents do so, provide a way to bypass the group. [Priority 3]

3.7 Images and image maps

Images may be inserted by two elements in HTML:

IMG
: Available in HTML 4.0, 3.2, 2.0

OBJECT
: Available in HTML 4.0.

Images include those that carry out simple animations (e.g., a"gif" image).

Equivalent text for images

[Checkpoint 1.1] Provide text equivalents for all images [Priority 1] When using IMG, specify alt-text with the "alt" attribute.

Example.

```
<IMG src="magnifyingglass.gif" alt="Search">
```

End example.

When using OBJECT, specify alt-text [p. 7]

in the body of the OBJECT element:

Example.

```
<OBJECT data="magnifyingglass.gif" type="image/gif">
   Search
</OBJECT>
```

End example.

Additionally, use the "title" attribute:

Example.

```
<OBJECT classid="Duke.class" title="Hello!"
        width="50" height="50">
     Duke waves hello!
</OBJECT>
```

End example.

Long descriptions for images

[Checkpoint 2.1] Provide a description of each graphic, script, or applet that conveys important information. [Priority 1] When using IMG, specify a long description [p. 8] of the image with the "longdesc" attribute:

Example.

```
<IMG src="97sales.gif" alt="Sales for 1997"
     title="Sales pie chart"
     longdesc="sales97.html">
```

In sales97.html:

A chart showing how sales in 1997 progressed. The chart
is a bar-chart showing percentage increases in sales
by month. Sales in January were up 10% from December 1996,
sales in February dropped 3%, ..

End example.

For browsers that don’t support "longdesc", provide a description link as well next to the graphic:

Example.

```
<IMG src="97sales.gif" alt="Sales for 1997" longdesc="sales.html">
<A href="sales.html" title="Description of 1997 sales figures">[D]</A>
```

End example.

When using OBJECT, provide a long description in the body of the element:

Example.

```
<OBJECT data="97sales.gif" type="image/gif">
       Sales in 1997 were down subsequent to our
       anticipated purchase ...
</OBJECT>
```

End example.

Or, provide a link to a long description within the body of the element:

Example.

```
<OBJECT data="97sales.gif" type="image/gif">
       Chart of our Sales in 1997.
       A <A href="desc.html">textual description</A> is available.
</OBJECT>
```

End example.

Ascii art

[Checkpoint 1.6] Replace ASCII art with an image or describe the ASCII art and provide a means (e.g., a link) to skip over it. [Priority 1 or Priority 2 depending on the importance of the information.]

Avoid ascii art (character illustrations) and use real images instead since it is easier to supply alt-text [p. 7] and long descriptions [p. 8] for images. The priority of this checkpoint depends on the importance of the information (e.g., an important chart).

However, if ascii art must be used provide a link to jump over the ASCII art, as follows.

Example.

```
<P>
<a href="#post-art">skip over ASCII art</a>
<!-- ASCII art goes here -->
<a name="post-art">caption for ASCII art</a>
```

End example.

ASCII art may also be marked up as follows:

Example.

```
<P>
<OBJECT data="cow.txt" type="text/x-ascii-art" title="drawing of a cow">
include cow ascii art from guidelines
</OBJECT>
```

End example.

Another option is to use a SPAN or an ABBR element with "title".

Example.

```
<P><SPAN class="smile" title="smiley in ascii art">:-)</SPAN>
```

End example.

If the description of (important) ASCII art is long, provide a description [p. 8] in addition to alt-text.

Another way to replace ascii art is to use human language substitutes. For example, <wink> might substitute for the emoticon <SPAN title="wink smiley">;-)</SPAN>, the word"therefore" could replace arrows consisting of dashes and greater than signs (e.g., -->), and the word "great" for the uncommon abbreviation "gr8".

Mathematical equations in images

Image maps

An image map is an image that has "active regions". When the user selects one of the regions, some action takes place -- a link may be followed, information sent to a server, etc. To make an image map accessible, content developers must ensure that each action associated with a visual region may be activated without a pointing device.

Image maps are created with the MAP element (available in HTML 4.0 and 3.2). HTML allows two types of image maps: client-side (the user's browser processes a URI) and server-side (the server processes click coordinates). For all image maps, content developers must supply alt-text [p. 7] (described below), and long descriptions [p. 8] where needed (see the section on long descriptions for images [p. 28]).

Content developers should either avoid server-side image maps (because they require a specific input device - a mouse) or provide the same functionality or information in an alternative accessible format. One way to achieve this is to provide a textual link for each active region so that each link is navigable with the keyboard [p. 10] . If you must use a server-side image map, please consult the section on server-side image maps [p. 32]

Client-side image maps

The active regions of a client-side image map are defined within the MAP element and may be created with two elements:

AREA
: Available in HTML 4.0, 3.2

A
Available in HTML 4.0.

[Checkpoint 1.5] Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape. [Priority 1]

Provide text equivalents [p. 7] for image maps since they convey visual information.

[Checkpoint 1.4] Provide a text equivalent for each active region of an image map. [Priority 1] If AREA is used, use the"alt" attribute:

Example.

```
<IMG src="welcome.gif" alt="Image map of areas in the library"
     usemap="#map1">
<MAP name="map1">
  <AREA shape="rect" coords="0,0,30,30"
        href="reference.html" alt="Reference">
  <AREA shape="rect" coords="34,34,100,100"
        href="media.html" alt="Audio visual lab">
</MAP>
```

End example.

The same idea, but use OBJECT instead of IMG to insert the image to provide more information about the image:

Example.

```
<OBJECT data="welcome.gif" type="image/gif" usemap="#map1">
    There are several areas in the library including
    the <A href="reference.html">Reference</A> section and the
    <A href="media.html">Audio Visual Lab</A>.
</OBJECT>
<MAP name="map1">
  <AREA shape="rect" coords="0,0,30,30"
        href="reference.html" alt="Reference">
  <AREA shape="rect" coords="34,34,100,100"
        href="media.html" alt="Audio visual lab">
</MAP>
```

End example.

[Checkpoint 1.7] Provide redundant text links for each active region of an image map. [Priority 1 - if server-side image maps are used, Priority 2 - if client-side image maps are used. Content developers will not need to provide redundant text links for client-side image maps once most user agents render text equivalents for the map links.] In addition to providing alt-text, provide redundant textual links. If the A element is used instead of AREA, the content developer may describe the active regions and provide redundant links at the same time:

Example.

```
<OBJECT data="navbar1.gif" type="image/gif" usemap="#map1">
<MAP name="map1">
  <P>Navigate the site.
  <A href="guide.html" shape="rect"
     coords="0,0,118,28">[Access Guide]</A>
  <A href="shortcut.html" shape="rect"
     coords="118,0,184,28">[Go]</A>
  <A href="search.html" shape="circle"
     coords="184.200,60">[Search]</A>
  <A href="top10.html" shape="poly"
     coords="276,0,373,28,50,50">[Top Ten]</A>
</MAP>
</OBJECT>
```

End example.

Note that in the previous example the MAP element is the content of the OBJECT element so that the alternative links will only be displayed if the image map (navbar1.gif) is not.

Note also that links have been separated by brackets ([]). This is to prevent screen readers from reading several adjacent links as a single link.

[Checkpoint 12.5] Include non-link, printable characters (surrounded by spaces) between links that occur consecutively. [Priority 3] Content developers should make sure they include printable characters (such as brackets or a vertical bar (|)) surrounded by spaces between adjacent links.

Server-side image maps

When a server-side image map must be used, content developers should provide an alternative list of image map choices. There are three techniques:

- If an alternative list of links follows the image map, content developers should indicate the existence and location of the alternative list. If IMG is used to insert the image, provide this information in the"alt" attribute. If OBJECT is used, provide it in the "title" attribute.

  Example.

  ```
  <IMG src="welcome.gif" alt="Links to this image map follow immediately"
       usemap="#map1" title="Welcome insignia" >
  <MAP name="map1">
    <AREA shape="rect" coords="0,0,30,30"
          href="reference.html" alt="Reference">
    <AREA shape="rect" coords="34,34,100,100"
          href="media.html" alt="Audio visual lab">
  </MAP>

  <P><A href="reference.html">[Reference]</A>
     <A href="media.html">[Audio Visual Lab]</A>
  ```

End example.

- A more straightforward solution, although newer and less backwards compatible, is to include the alternative links within the body of an OBJECT element (see the previous example illustrating links in the OBJECT element [p. 32]).
- One final possibility is to create an alternative page [p. 9] that is accessible.

[Checkpoint 1.8] Provide individual button controls in a form rather than simulating a set of buttons with an image map. [Priority 2]

3.8 Applets and other objects

Applets may be inserted by two elements in HTML:

APPLET
: Available in HTML 4.0 (deprecated), 3.2.

OBJECT
: Available in HTML 4.0.

Other objects, such as those requiring a plug-in, should also use the OBJECT element. However, for backwards compatibility with Netscape browsers, use the proprietary EMBED element within the OBJECT element as follows:

Example.

```
<OBJECT classid="clsid:A12BCD3F-GH4I-56JK-xyz"
codebase="http://site.com/content.cab" width=100 height=80>
<PARAM name="Movie" value="moviename.swf">
    <EMBED src="moviename.swf" width=100 height=80
    pluginspage="http://www.macromedia.com/shockwave/download/">
    </EMBED>

    <NOEMBED>
        <IMG src="moviename.gif" width=100 height=80>
    </NOEMBED>

</OBJECT>
```

End example.

For more information refer to [MACROMEDIA] [p. 52] .

Audio and Video produced by dynamic objects

There are several checkpoints that content developers should use to ensure that audio or visual information presented by a dynamic object is accessible:

1. Refer also to checkpoint 1.2.
2. Provide a long description if necessary. See the section on long descriptions for images [p. 28] .
3. [Checkpoint 2.1] Provide a description of each graphic, script, or applet that

conveys important information. [Priority 1]
4. Refer also to checkpoint 10.1.
5. [Checkpoint 8.5] Provide non-text equivalents for all applets and other programmatic objects. [Priority 2]

Equivalent text and descriptions for applets and programmatic objects

[Checkpoint 1.2] Provide text equivalents for all applets and other programmatic objects. [Priority 1] If OBJECT is used, provide alt-text [p. 7] as the content of the element:

Example.

```
<OBJECT classid="java:Press.class" width="500" height="500"
        title="Java applet: how temperature affects pressure">
      As temperature increases, the molecules in the balloon...
</OBJECT>
```

End example.

A more complex example takes advantage of the fact the OBJECT elements may be embedded to provide for alternative representations of information:

Example.

```
<OBJECT title="How temperature affects pressure"
        classid="java:Press.class" width="500" height="500">
   <OBJECT data="Pressure.mpeg" type="video/mpeg">
      <OBJECT data="Pressure.gif" type="image/gif">
         As temperature increases, the molecules in the balloon...
      </OBJECT>
   </OBJECT>
</OBJECT>
```

End example.

Refer also to checkpoint 1.2. If APPLET is used, provide alt-text [p. 7] with the "alt" attribute and content in the APPLET element. This enables them to transform gracefully for those user agents that only support one of the two mechanisms ("alt" or content).

Deprecated example.

```
<APPLET code="Press.class" width="500" height="500"
        alt="Java applet: how temperature affects pressure">
      As temperature increases, the molecules in the balloon...
</APPLET>
```

Directly accessible applets

[Checkpoint 10.1] 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.] If an applet requires user interaction (e.g., the ability to manipulate a physics experiment) that cannot be duplicated in an alternative format, make the applet directly accessible.

For more information about accessible applets, please refer to [JAVAACCESS] [p. 51] and [IBMJAVA] [p. 51] .

3.9 Audio and video

Audio and video should be accompanied by *text transcripts*, textual descriptions or equivalents of auditory or visual events. When these transcripts are presented synchronously with a video presentation they are called "captions" and are used by people who cannot hear the audio track of the video material. Full audio transcripts include spoken dialogue as well as any other significant sounds including on-screen and off-screen sounds, music, laughter, applause, etc. The following two examples show captions, a text transcript, and auditory descriptions.

Example.

Captions for a scene from "E.T." The phone rings three times, then is answered.

[phone rings]

[ring]

[ring]

Hello?"

End example.

Example.

Here's an example of a transcript of a clip from "The Lion King" (available at [DVS] [p. 51]).

Simba: Yeah!

Describer: Simba races outside, followed by his parents. Sarabi smiles and nudges Simba gently toward his father. The two sit side-by-side, watching the golden sunrise.

Mufasa: Look Simba, everything the light touches is our kingdom.

Simba: Wow.

End example.

Audio information

[Checkpoint 3.1] For stand-alone audio files, provide a text transcript of all words (spoken or sung) and all important sounds. [Priority 1]

[Checkpoint 3.2] For audio associated with video, synchronize the text transcript with the video. [Priority 1]

Some media formats, such as QuickTime 3.0, allow captions and video descriptions to be added to the multimedia clip.

Until the format you are using supports alternative tracks, two versions of the movie could be made available, one with captions and descriptive video, and one without. Future technologies, on the other hand, will allow separate audio/visual files to be combined with text files via a synchronization file to create captioned audio and movies. It will also allow the user to choose from multiple sets of captions to match their reading skills. For more information see the SMIL 1.0 ([SMIL] [p. 51]) specification.

[Checkpoint 3.3] Where sounds are played automatically, provide visual notification and transcripts. [Priority 1 or Priority 2 depending on the importance of the sound.]

This can be provided in the form of a text phrase on the page that links to a text transcript or description of the sound file. The link to the transcript should appear in a highly visible location such as at the top of the page. However, if a script is automatically loading a sound, it should also be able to automatically load a visual indication that the sound is currently being played and provide a description or transcript of the sound.

Note. Some controversy surrounds this technique because the browser should load the visual form of the information instead of the auditory form if the user preferences are set to do so. However, strategies must also work with today's browsers.

For more information, please refer to [NCAM] [p. 52] .

Visual information and motion

Video descriptions are used primarily by people who are blind to follow the action and other non-auditory information in video material. The description provides narration of the key visual elements without interfering with the audio or dialogue of a movie. Key visual elements include actions, settings, body language, graphics, and displayed text.

[Checkpoint 2.2] For movies, provide auditory descriptions that are synchronized with the original audio. [Priority 1] For movies, provide auditory descriptions that are synchronized with the original audio. See the section on audio information [p. 35] for more information about multimedia formats.

[Checkpoint 3.4] Provide a text version of the auditory description and collate it with the text transcript (captions) of the primary audio track. [Priority 2] This text transcript, in conjunction with the full audio transcript described above, allows access by people with both visual and hearing disabilities. This also provides everyone with the ability to index and search for information contained in audio/visual materials.

[Checkpoint 9.3] Avoid movement in pages. When a page includes moving content, provide a mechanism to allow users to freeze motion or updates in applets and scripts or use style sheets and scripting to create movement. [Priority 2]

However, if necessary to include an applets that involves motion or updates, content developers should provide a mechanism for freezing this motion (for an example, refer to [TRACE] [p. 52]). Content developers should use animated gifs to create motion that may be suspended by the browser for people that have trouble with it.

[Checkpoint 1.3] For short animations such as "animated gifs," provide a text equivalent and a description if needed. [Priority 1] When necessary, a long description should be provided for visual information that is necessary to understanding the page. For example for an ad that displays text like a marquee, the text should be provided in the text equivalent, unless there is a lot of text. For a looping image of cloud cover over the United States, if the image is in the context of a weather status report, where the information is presented in text, a less verbose description of the image is necessary. However, if the image appears on in a pedagogical setting, elaborating on cloud formations in relation to land mass, ought to be described.

See also the section on text style [p. 38] for controlling blinking.

3.10 Style and style sheets

[Checkpoint 5.5] Use style sheets to control layout and presentation. [Priority 2] CSS1 ([[CSS1] [p. 50]) and CSS2 ([[CSS2] [p. 50]) allow content developers to duplicate almost every HTML 4.0 presentation feature and offer more power with less cost. However, until most users have browsers that support style sheets, not every presentation idiom may be expressed satisfactorily with style sheets. In the following sections, we show how style sheets may be used to create accessible pages. We also provide examples of how to use HTML 4.0 features (e.g., tables, bitmap text) more accessibly when they must be used.

See also the section on text markup [p. 19] .

General style sheet techniques

- [Checkpoint 8.3] For pages that use style sheets or presentation markup, ensure that the content of each page is organized logically. [Priority 1]
- [Checkpoint 5.7] Use relative rather than absolute units in markup language attribute values and style sheet property values. [Priority 2]

Text formatting

Content developers should use style sheets for text formatting rather than converting text to images. For example, stylized text on a colored background can be created with style sheets instead of as an image. This provides flexibility for people to view the text in a form that is most readable to them including magnified, in a particular color combination such as white on black, or in a particular font.

However, if you must use a bitmap to create a text effect (special font, transformation, shadows, etc.) it must be accessible.

When bitmapped text is used in a way that makes a page inaccessible, content developers must supply alternative pages [p. 9] .

To make a bitmap representing text accessible, it must have alt-text [p. 7] that is the same text represented by the image.

Example.

In this example, the inserted image shows the large red characters "Example", reflected by the alt-text.

```
<P>This is an
   <IMG src="BigRedExample.gif" alt="Example"> of what we mean.
</P>
```

End example.

This is true of Drop Caps (large first letter of a paragraph) as well. However, we recommend using style sheets to create the effect, as the following example illustrates.

Example.

```
<HEAD>
<TITLE>Drop caps</TITLE>
<STYLE type="text/css">
      .dropcap { font-size : 120%; font-family : Helvetica }
</STYLE>
</HEAD>
<BODY>
<P><SPAN class="dropcap">O</SPAN>nce upon a time...
</BODY>
```

Note. As of the writing of this document, the CSS pseudo-element ':first-letter', which allows content developers to refer to the first letter of a chunk of text, is not widely supported.

Text style

Content developers should use style sheets instead of deprecated presentation elements and attributes [p. 7] that control visual presentation.

- [Checkpoint 9.2] Avoid any blinking or updating of the screen that causes flicker. [Priority 1] Blinking may cause seizures in some users and is annoying to many other users. They should also provide a mechanism for freezing motion. If style sheets are used to create an effect (e.g., 'text-decoration: blink'), users may cancel the effect through style sheets as well.
  Note. Do not use the BLINK and MARQUEE elements. These elements are not part of any W3C specification for HTML (i.e., they are non-standard elements).

- Indentation: Use the CSS 'text-indent' property to indent text. Do not use the BLOCKQUOTE or any other structural element to indent text.
- Spacing: Content developers should achieve spacing effects with the CSS 'word-spacing' and 'white-space' properties rather than by putting actual spaces between letters and using the PRE element.

Fonts

Instead of using deprecated presentation elements and attributes [p. 7] , use the many CSS properties to control font characteristics: 'font-family', 'font-size', 'font-size-adjust', 'font-stretch', 'font-style', 'font-variant', and 'font-weight'.

Colors

Use these CSS properties to specify colors:

- 'color', for foreground text color.
- 'background-color', for background colors.
- 'border-color', for border colors.
- For link colors, see the :link, :visited, and :active pseudo-classes.

[Checkpoint 4.1] Ensure that all information conveyed with color is also available without color, for example from context or markup. [Priority 1] For example, in this document, examples appear in a different color than the rest of the text. However, that is not enough to identify them as examples, so we precede each one with the word "Example." or "Deprecated example."

Quicktest! To test whether your page passes the text, examine it with a monochrome monitor or colors turned off.

[Checkpoint 4.2] 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]

Quicktest! To test whether color contrast is sufficient to be read by people with color deficiencies or by those with low resolution monitors, print pages on a black and white printer (with backgrounds and colors appearing in grayscale).

For more information about colors and contrasts, refer to [LIGHTHOUSE] [p. 52] .

Layout, positioning, layering, and alignment

Layout, positioning, layering, and alignment should be done through style sheets (notably by using CSS floats and absolute positioning).

When laying out tabular information, content developers should use tables that are designed for accessibility (see the section on tables [p. 21] . Do not use PRE to create a tabular layout of text.

[Checkpoint 12.3] Provide a linear text alternative (on the current page or some other) for *all* tables that lay out text in parallel, word-wrapped columns. [Priority 2] However, until user agents and screen readers are able to handle text presented side-by-side, all tables that lay out text in parallel, word-wrapped columns require a linear text alternative (on the current page or some other). See also the section on tables [p. 21] for additional information on table accessibility.

Provide text equivalents for all images, including invisible or transparent images.

If content developers cannot use style sheets and must use invisible or transparent images to lay out images on the page, they should supply "null" alt-text (alt="") for them.

Deprecated example.

In this example, an image is used to create a carefully defined space between words or graphics. "White space" alt-text is used to prevent the words from running together when the image is not loaded:

```
my poem requires a big space<IMG src="10pttab.gif" alt="   ">here
```

In this next example, an image is used to force a graphic to appear in a certain position:

```
<IMG src="spacer.gif" alt="spacer">
<IMG src="colorfulwheel.gif" alt="The wheel of fortune">
```

Rules and borders

Rules and borders may convey the notion of "separation" to visually enabled users but that meaning cannot be inferred out of a visual context.

While content developers may use HR to create a horizontal rule, they should do so in a way that also conveys the structure in a non-visual way (e.g., by using DIV in conjunction with the "class" attribute).

Example.

```
<DIV class="navigation-bar">
   <HR title="navigation-bar">
   <A rel="Next" href="next.html">[Next page]</A>
   <A rel="Previous" href="previous.html">[Prevous page]</A>
   <A rel="First" href="first.html">[First page]</A>
</DIV>
```

End example.

When using graphics (e.g., horizontal rules) as section separators, content developers may provide an advisory title of what the graphic represents to the visually enabled user via the "title" attribute. Hence, in the previous example, we specified title="navigation-bar".

Example.

In this example, a red line is used to separate Chapter 7 from Chapter 8:

```
<IMG src="redline.gif" alt="redline graphic"
     title="End of Chapter 7 - Visual Displays">
<H1>Chapter 8 - Auditory and Tactile Displays</H1>
```

We recommend using style sheets to accomplish such styling of the line:

```
<HEAD>
<TITLE>Redline with style sheets</TITLE>
<STYLE type="text/css">
     HR.redline { color : red }
</STYLE>
</HEAD>
<BODY>
<HR class="redline" title="End of Chapter 7 - Visual Displays">
<H1>Chapter 8 - Auditory and Tactile Displays</H1>
</BODY>
```

End example.

Guidelines for good CSS style sheets

Use a minimal number of style sheet for your site

If you have more than one, use the same "class" name for the same concept in all of the style sheets.

Use linked style sheets rather than embedded styles, and avoid inline style sheets.

Content developers should not write "!important" rules. Users should where necessary.

Always use the "em" unit to set font sizes.

Example.

Use:

```
H1 { font-size: 2em }
```

rather than:

```
H1 { font-size: 12pt }
```

End example.

Use relative length units and percentages:

Example.

```
BODY { margin-left: 15%; margin-right: 10%}
```

End example.

Only use absolute length units when the physical characteristics of the output medium are known.

Example.

```
.businesscard { font-size: 8pt }
```

End example.

Always specify a fallback generic font:

Example.

```
BODY { font-family: "Gill Sans", sans-serif }
```

End example.

Use numbers, not names, for colors:

Example.

```
H1 {color: #808000}
H1 {color: rgb(50%,50%,0%)}
```

End example.

3.11 Frames

For visually enabled users, frames may organize a page into different zones. For non-visual users, relationships between the content in frames (e.g., one frame has a table of contents, another the contents themselves) must be conveyed through other means.

Frames as implemented today are problematic for several reasons:

- Without scripting, they tend to break the "previous page" functionality offered by browsers.
- It is impossible to refer to the "current state" of a frameset with a URI; once a frameset changes contents, the original URI no longer applies.
- Opening a frame in a new browser window can disorient or simply annoy users.

In the following sections, we discuss how to make frames more accessible. We also provide an alternative to frames [p. 45] that uses HTML 4.0 and CSS and addresses many of the limitations of today's frame implementations.

Title frames for easy orientation

[Checkpoint 14.1] Title each frame so that users can keep track of frames by title. [Priority 1]

Example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>A simple frameset document</TITLE>
</HEAD>
<FRAMESET cols="10%, 90%"
```

```
          title="Our library of electronic documents">
    <FRAME src="nav.html" title="Navigation bar">
    <FRAME src="doc.html" title="Documents">
    <NOFRAMES>
       <A href="lib.html" title="Library link">
             Select to go to the electronic library</A>
    </NOFRAMES>
</FRAMESET>
```

End example.

Long descriptions of frames

[Checkpoint 14.2] Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone. [Priority 2]

Example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
  <HEAD>
    <TITLE>Today's news</TITLE>
  </HEAD>

  <FRAMESET COLS="10%,*,10%">

  <FRAMESET ROWS="20%,*">
    <FRAME SRC="promo.html" NAME="promo" title="promotions">
    <FRAME SRC="sitenavbar.html" NAME="navbar"
       title="Sitewide navigation bar" longdesc="frameset-desc.html#navbar">
  </FRAMESET>

  <FRAME SRC="story.html" NAME="story" title="Selected story - main content"
     longdesc="frameset-desc.html#story">

  <FRAMESET ROWS="*,20%">
    <FRAME SRC="headlines.html" NAME="index" title="Index of other
      national headlines" longdesc="frameset-desc.html#headlines">
    <FRAME SRC="ad.html" NAME="adspace" title="Advertising">
  </FRAMESET>

  <NOFRAMES>
    <p><a href="noframes.html">No frames version</a></p>
    <p><a href="frameset-desc.html">Descriptions of frames.</a></p>
  </NOFRAMES>

  </FRAMESET>
</HTML>
```

frameset-desc.html might say something like: #Navbar - this frame provides links to the major sections of the site: World News, National News, Local News, Technological News, and Entertainment News. #Story - this frame displays the currently selected story. #Index - this frame provides links to the day's headline stories within this section.

End example.

Note that if the a frame's contents change, the long descriptions will no longer apply. Also, links to a frame's longdesc ("d-links") ought to be provided with other alternative contents in the NOFRAMES element of a FRAMESET.

Invisible d-links

Ensure documents are readable without frames

Example.

In this example, if the user reads "top.html":

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>This is top.html</TITLE>
</HEAD>
<FRAMESET cols="50%, 50%" title="Our big document">
    <FRAME src="main.html" title="Where the content is displayed">
    <FRAME src="table_of_contents.html" title="Table of Contents">
    <NOFRAMES>
        <A href="table_of_contents.html">Table of Contents.</A>
     <!-- other navigational links that are available in main.html
          are available here also. -->
    </NOFRAMES>
</FRAMESET>
</HTML>
```

and the user agent is not displaying frames, the user will have access (via a link) to a non-frames version of the same information.

End example.

Always make the source of a frame an HTML document

Content developers must provide descriptions of frames so that their contents and the relationships between frames make sense. Note that as the contents of a frame change, so must change any description. This is not possible if an IMG is inserted directly into a frame, as in this deprecated example:

Deprecated example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>A bad frameset document</TITLE>
</HEAD>
<FRAMESET cols="100%" title="Static frameset">
   <FRAME name="badframe"
          src="apples.gif" title="Apples">
</FRAMESET>
</HTML>
```

Note that if, for example, a link causes a new image to be inserted into the frame:

```
<P>Visit a beautiful grove of
<A target="badframe" href="oranges.gif" title="Oranges">oranges</A>
```

the initial title of the frame ("Apples") will no longer match the current content of the frame ("Oranges").

[Checkpoint 8.1] Ensure that descriptions and text alternatives for dynamic content are updated when the dynamic content changes. [Priority 1] To solve this problem, content developers should always make the source ("src") of a frame an HTML file. Images may be inserted into the HTML file and their text alternatives will evolve correctly.

Example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>A correct frameset document</TITLE>
</HEAD>
<FRAMESET cols="100%" title="Evolving frameset">
<FRAME name="goodframe" src="apples.html" title="Apples">
</FRAMESET>
</HTML>

   <!-- In apples.html -->
   <P><IMG src="apples.gif" alt="Apples">
```

End example.

Avoid opening a new window as the target of a frame

[Checkpoint 12.1] Do not cause pop-ups or other windows to appear and do not change the current window without informing the user. [Priority 2] For example, content developers should avoid specifying a new window as the target of a frame with target="_blank".

Alternatives to frames

One of the most common uses of frames is to split the user's browser window into two parts: a navigation window and a content window. As an alternative to frames, we encourage you to try the following:

1. Create one document for the navigation mechanism (call it "nav.html"). A separate document means that the navigation mechanism may be shared by more than one document.
2. In each document requiring the navigation mechanism, include it at the bottom of the document with the following (or similar) OBJECT markup:

   Example.

```
<OBJECT data="nav.html">
Go to the <A href="nav.html"<table of contents</A>
</OBJECT>
```

Putting the navigation mechanism at the end of the document means that when style sheets are turned off, users have access to the document's important information first.

3. Use style sheets to position the navigation mechanism where you want on the screen. For example, the following CSS rule floats the navigation bar to the left of the page and makes it take up 25% of the available horizontal space:

```
OBJECT { float: left; width: 25% }
```

The following CSS rule attaches the navigation mechanism to the bottom-left corner of the page of the page and keeps it there even if the user scrolls down the page:

```
OBJECT { position: fixed; left: 0; bottom: 0 }
```

Note. Navigation mechanisms or other content may be inserted in a document by means of server-side includes.

3.12 Forms

Make controls keyboard accessible

[Checkpoint 11.3] Create a logical tab order through links, form controls, and objects. [Priority 3]

[Checkpoint 11.4] Provide keyboard shortcuts to links, including those in client-side image maps, form controls, and groups of form controls. [Priority 3] See the section on keyboard access [p. 10] for more information.

Group form controls

[Checkpoint 14.3] Group form controls. [Priority 2 - for radio buttons and checkboxes, Priority 3 - for other controls.] Content developers should group form controls into semantic units with the FIELDSET element and label those units with the LEGEND element (both available in HTML 4.0):

Example.

```
<FORM action="http://somesite.com/adduser" method="post">
   <FIELDSET>
   <LEGEND>Personal information</LEGEND>
   <LABEL for="firstname">First name: </LABEL>
   <INPUT type="text" id="firstname" tabindex="1">
   <LABEL for="lastname">Last name: </LABEL>
   <INPUT type="text" id="lastname" tabindex="2">
   ...more personal information...
   </FIELDSET>
   <FIELDSET>
```

```
   <LEGEND>Medical History</LEGEND>
   ...medical history information...
   </FIELDSET>
</FORM>
```

End example.

Label form controls explicitly

[Checkpoint 14.4] Associate labels explicitly with their controls. [Priority 2] An example of LABEL used with "for" in HTML 4.0 is given in the previous section.

[Checkpoint 12.2] For all form controls with implicitly associated labels, ensure that the label is properly positioned. [Priority 2]

Group menu options

[Checkpoint 14.5] Divide long lists of choices into manageable groups. [Priority 2] For long lists of selections (which are hard to remember), content developers should group items into a hierarchy using the OPTGROUP element (available in HTML 4.0).

Example.

```
<FORM action="http://somesite.com/prog/someprog" method="post">
 <P>
 <SELECT name="ComOS">
     <OPTGROUP label="PortMaster 3">
       <OPTION label="3.7.1" value="pm3_3.7.1">PortMaster 3 with ComOS 3.7.1
       <OPTION label="3.7" value="pm3_3.7">PortMaster 3 with ComOS 3.7
       <OPTION label="3.5" value="pm3_3.5">PortMaster 3 with ComOS 3.5
     </OPTGROUP>
     <OPTGROUP label="PortMaster 2">
       <OPTION label="3.7" value="pm2_3.7">PortMaster 2 with ComOS 3.7
       <OPTION label="3.5" value="pm2_3.5">PortMaster 2 with ComOS 3.5
     </OPTGROUP>
     <OPTGROUP label="IRX">
       <OPTION label="3.7R" value="IRX_3.7R">IRX with ComOS 3.7R
       <OPTION label="3.5R" value="IRX_3.5R">IRX with ComOS 3.5R
     </OPTGROUP>
 </SELECT>
</FORM>
```

End example.

Techniques for specific controls

[Checkpoint 12.4] Include default, place-holding characters in edit boxes and text areas. [Priority 3]

Example.

```
<FORM action="http://somesite.com/prog/text-read" method="post">
     <P>
     <TEXTAREA name=yourname rows="20" cols="80">
     Please enter your name here.
     </TEXTAREA>
     <INPUT type="submit" value="Send"><INPUT type="reset">
     </P>
</FORM>
```

End example.

Provide alt-text [p. 7] for images used as "submit" buttons:

Example.

```
<FORM action="http://somesite.com/prog/text-read" method="post">
<P>
<INPUT type="image" name=submit src="button.gif" alt="Submit">
</FORM>
```

End example.

Refer also to checkpoint 1.8.

Also see the section on keyboard access [p. 10] since this applies to form controls.

Backwards compatibility issues for forms

The BUTTON element does not appear and <INPUT type="button"> will appear as a text input field in HTML3.2 browsers.

3.13 Scripts

Content developers must ensure that pages are accessible with scripts turned off or in browsers that don't support scripts.

Alternative presentation of scripts

[Checkpoint 8.2] For scripts that present important information or functionality, provide an equivalent. (Refer to the definition of equivalent.) [Priority 1] One way to accomplish this is with the NOSCRIPT element (available in HTML 4.0). The content of this element is rendered with scripts are not enabled.

Example.

```
<SCRIPT type="text/tcl">
 ...some Tcl script to show a billboard of sports scores...
</SCRIPT>
<NOSCRIPT>
   <P>Results from yesterday's games:</P>
   <DL>
      <DT>Bulls 91,  Sonics 80.
      <DD><A href="bullsonic.html">Bulls vs. Sonics game highlights</A>
      ...more scores...
   </DL>
</NOSCRIPT>
```

End example.

Device-independent event handlers

[Checkpoint 11.2] For scripts, specify logical event handlers rather than device-dependent event handlers. [Priority 2]

An event handler is a script that is invoked when a certain event occurs (e.g, the mouse moves, a key is pressed, the document is loaded, etc.). In HTML 4.0, event handlers are attached to elements via script attributes [p. 6] (the attributes beginning with "on", as in "onkeyup").

Some event handlers, when invoked, produce purely decorative effects such as highlighting an image or changing the color of an element's text. Other event handlers produce much more substantial effects, such as carrying out a calculation, providing important information to the user, or submitting a form. For event handlers that do more than just change the presentation of an element, content developers should do the following:

1. Use application-level event triggers rather than user interaction-level triggers. In HTML 4.0, application-level event attributes are "onfocus", "onblur" (the opposite of "onfocus"), and "onselect". Note that these attributes are also device-independent.
2. Otherwise, if you must use device-dependent attributes, provide redundant input mechanisms (i.e., specify two handlers for the same element):
   - Use "onmousedown" with "onkeydown".
   - Use "onmouseup" with "onkeyup"
   - Use "onclick" with "onkeypress"

   Note that there is no keyboard equivalent to double-clicking ("ondblclick") in HTML 4.0.
3. Do not write event handlers that rely on mouse coordinates since this prevents device-independent input.

---

4 Acknowledgments

WAI Markup Guidelines Working Group Co-Chairs:
Chuck Letourneau, Starling Access Services
Gregg Vanderheiden, Trace Research and Development
W3C Team contacts:
Judy Brewer and Daniel Dardailler
We wish to thank the following people who have contributed their time and valuable comments to shaping these guidelines:
Harvey Bingham, Kevin Carey, Chetz Colwell, Neal Ewers, Geoff Freed, Al Gilman, Larry Goldberg, Jon Gunderson, Eric G. Hansen, Phill Jenkins, Leonard Kasday, George Kerscher, Marja-Riitta Koivunen, Josh Krieger, Scott Luebking, William Loughborough, Murray Maloney, Charles McCathieNevile, MegaZone

(Livingston Enterprises), Masafumi Nakane, Mark Novak, Charles Oppermann, Mike Paciello, David Pawson, Michael Pieper, Greg Rosmaita, Liam Quinn, Dave Raggett, T.V. Raman, Robert Savellis, Jutta Treviranus, Steve Tyler, Jaap van Lelieveld, and Jason White

The original draft of this document is based on "The Unified Web Site Accessibility Guidelines" ([UWSAG] [p. 51]) compiled by the Trace R & D Center at the University of Wisconsin. That document includes a list of additional contributors.

5 Reference specifications

[CSS1]
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds. The CSS1 Recommendation is available at:
http://www.w3.org/TR/1999/REC-CSS1-19990111.

[CSS2]
"CSS, level 2 Recommendation", B. Bos, H. Wium Lie, C. Lilley, and I. Jacobs, eds. The CSS2 Recommendation is available at:
http://www.w3.org/TR/1998/REC-CSS2-19980512.

[DOM1]
"Document Object Model (DOM) Level 1 Specification", V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood, eds. The DOM Level 1 Recommendation is available at:
http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001.

[HTML40]
"HTML 4.0 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds. The HTML 4.0 Recommendation is available at:
http://www.w3.org/TR/1998/REC-html40-19980424.

[HTML32]
"HTML 3.2 Recommendation", D. Raggett, ed. The HTML 3.2 Recommendation is available at:
http://www.w3.org/TR/REC-html32.

[MATHML]
"Mathematical Markup Language", P. Ion and R. Miner, eds. The MathML 1.0 Recommendation is available at:
http://www.w3.org/TR/1998/REC-MathML-19980407.

[PNG]
"PNG (Portable Network Graphics) Specification", T. Boutell, ed., T. Lane, contributing ed. The PNG Recommendation is available at:
http://www.w3.org/TR/REC-png.

[RDF]
"Resource Description Framework (RDF) Model and Syntax Specification", O. Lassila, R. Swick, eds. The RDF Recommendation is available at:
http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[SMIL]
"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", P. Hoschka, editor. The SMIL 1.0 Recommendation is available at: http://www.w3.org/TR/1998/REC-smil-19980615

[WAI-AUTOOLS]
"Authoring Tool Accessibility Guidelines", J. Treviranus, J. Richards, I. Jacobs, C. McCathieNevile, eds. The latest Working Draft of these guidelines for designing accessible authoring tools is available at: http://www.w3.org/TR/WD-WAI-AUTOOLS/

[WAI-USERAGENT]
"User Agent Accessibility Guidelines", J. Gunderson and I. Jacobs, eds. The latest Working Draft of these guidelines for designing accessible user agents is available at: http://www.w3.org/TR/WD-WAI-USERAGENT/

[UWSAG]
"The Unified Web Site Accessibility Guidelines", G. Vanderheiden, W. Chisholm, eds. The Unified Web Site Guidelines were compiled by the Trace R & D Center at the University of Wisconsin under funding from the National Institute on Disability and Rehabilitation Research (NIDRR), U.S. Dept. of Education. This document is available at: http://www.tracecenter.org/docs/html_guidelines/version8.htm

[XML]
"Extensible Markup Language (XML) 1.0.", T. Bray, J. Paoli, C.M. Sperberg-McQueen, eds. The XML 1.0 Recommendation is available at: http://www.w3.org/TR/1998/REC-xml-19980210

6 Services

Note. *W3C cannot maintain stability for any of the following references outside of its control. These references are included for convenience.*

[DVS]
DVS Descriptive Video Services.

[BOBBY]
Bobby is an automatic accessibility validation tool developed by Cast.

[CSSVAL]
The W3C CSS Validation Service.

[HTMLVAL]
The W3C HTML Validation Service.

[IBMJAVA]
IBM Guidelines for Writing Accessible Applications Using 100% Pure Java are available from IBM Special Needs Systems.

[JAVAACCESS]
Information about Java Accessibility and Usability is available from the Trace R&D Center.

[LIGHTHOUSE]
The Lighthouse provides information about accessible colors and contrasts.
[LYNX]
Lynx is a text-only browser.
[LYNXME]
Lynx-me is a Lynx emulator.
[LYNXVIEW]
Lynx Viewer is a Lynx emulator.
[MACROMEDIA]
Flash OBJECT and EMBED Tag Syntax from Macromedia.
[NCAM]
The National Center for Accessible Media includes information about captioning and audio description on the Web.
[TRACE]
The Trace Research & Development Center. Consult this site for a variet of information about accessibility, including a scrolling Java applet that may be frozen by the user.

---

7 Checkpoint Map

This index lists each checkpoint and the sections in this document where it is discussed.

Guideline 1:

Checkpoint 1.1 [p. 27]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Equivalent text for images [p. 27] "
Checkpoint 1.2 [p. 34]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Audio and Video produced by dynamic objects [p. 33] " and "Equivalent text and descriptions for applets and programmatic objects [p. 34] "
Checkpoint 1.3 [p. 37]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Visual information and motion [p. 36] "
Checkpoint 1.4 [p. 31]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Client-side image maps [p. 30] "
Checkpoint 1.5 [p. 31]
Refer to "Client-side image maps [p. 30] "
Checkpoint 1.6 [p. 29]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Ascii art [p. 29] "

Checkpoint 1.7 [p. 31]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Client-side image maps [p. 30] "
Checkpoint 1.8 [p. 33]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Server-side image maps [p. 32] " and "Techniques for specific controls [p. 47] "

Guideline 2:

Checkpoint 2.1 [p. 33]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Long descriptions for images [p. 28] " and "Audio and Video produced by dynamic objects [p. 33] "
Checkpoint 2.2 [p. 36]
Refer to "Visual information and motion [p. 36] "

Guideline 3:

Checkpoint 3.1 [p. 35]
Refer to "Audio information [p. 35] "
Checkpoint 3.2 [p. 35]
Refer to "Audio information [p. 35] "
Checkpoint 3.3 [p. 36]
Refer to "Audio information [p. 35] "
Checkpoint 3.4 [p. 36]
Refer to "Visual information and motion [p. 36] "

Guideline 4:

Checkpoint 4.1 [p. 39]
Refer to "Colors [p. 39] "
Checkpoint 4.2 [p. 39]
Refer to "Colors [p. 39] "

Guideline 5:

Checkpoint 5.1 [p. 17]
Refer to "Section headers [p. 16] "
Checkpoint 5.2 [p. 19]
Refer to "3.4 Lists [p. 19] "
Checkpoint 5.3 [p. 18]
Refer to "Quotations [p. 18] "
Checkpoint 5.4 [p. 16]
Refer to "3.1 Document structure and metadata [p. 16] "

Checkpoint 5.5 [p. 37]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "3.10 Style and style sheets [p. 37] "
Checkpoint 5.6 [p. 19]
Refer to "Text markup rather than images [p. 19] "
Checkpoint 5.7 [p. 37]
Refer to "General style sheet techniques [p. 37] "

Guideline 6:

Checkpoint 6.1 [p. 17]
Refer to "3.2 Language information [p. 17] "
Checkpoint 6.2 [p. 18]
Refer to "Acronyms and abbreviations [p. 18] "
Checkpoint 6.3 [p. 18]
Refer to "3.2 Language information [p. 17] "

Guideline 7:

Checkpoint 7.1 [p. 21]
Refer to "3.5 Tables [p. 21] "
Checkpoint 7.2 [p. 22]
Refer to "3.5 Tables [p. 21] "
Checkpoint 7.3 [p. 26]
Refer to "Avoid tables for layout [p. 26] "
Checkpoint 7.4 [p. 22]
Refer to "3.5 Tables [p. 21] "
Checkpoint 7.5 [p. 21]
Refer to "3.5 Tables [p. 21] "
Checkpoint 7.6 [p. 22]
Refer to "3.5 Tables [p. 21] "

Guideline 8:

Checkpoint 8.1 [p. 45]
Refer to "Always make the source of a frame an HTML document [p. 44] "
Checkpoint 8.2 [p. 48]
Refer to "Alternative presentation of scripts [p. 48] "
Checkpoint 8.3 [p. 37]
Refer to "General style sheet techniques [p. 37] "
Checkpoint 8.4 [p. 9]
Refer to "2.3 Alternative pages [p. 9] " and "Ensure documents are readable without frames [p. 44] "
Checkpoint 8.5 [p. 34]
Refer to "Audio and Video produced by dynamic objects [p. 33] "

Guideline 9:

Checkpoint 9.1 [p. 14]
Refer to "2.8 Automatic page refresh [p. 14] "
Checkpoint 9.2 [p. 38]
Refer to "Text style [p. 38] "
Checkpoint 9.3 [p. 36]
Refer to "Visual information and motion [p. 36] "
Checkpoint 9.4 [p. 14]
Refer to "2.8 Automatic page refresh [p. 14] "

Guideline 10:

Checkpoint 10.1 [p. 34]
Refer to "Audio and Video produced by dynamic objects [p. 33] " and "Directly accessible applets [p. 34] "

Guideline 11:

Checkpoint 11.1 [p. 11]
Refer to "2.4 Keyboard access [p. 10] "
Checkpoint 11.2 [p. 49]
Refer to "Device-independent event handlers [p. 49] "
Checkpoint 11.3 [p. 46]
Refer to "Make controls keyboard accessible [p. 46] "
Checkpoint 11.4 [p. 46]
Refer to "Make controls keyboard accessible [p. 46] "

Guideline 12:

Checkpoint 12.1 [p. 45]
Refer to "Avoid opening a new window as the target of a frame [p. 45] "
Checkpoint 12.2 [p. 47]
Refer to "Label form controls explicitly [p. 47] "
Checkpoint 12.3 [p. 40]
Refer to "Layout, positioning, layering, and alignment [p. 39] "
Checkpoint 12.4 [p. 47]
Refer to "Techniques for specific controls [p. 47] "
Checkpoint 12.5 [p. 32]
Refer to "Client-side image maps [p. 30] "

Guideline 13:

Checkpoint 13.1 [p. 15]
Refer to "2.11 Browser Support [p. 15] "
Checkpoint 13.2 [p. 5]
Refer to "2.1 Structure vs. Presentation [p. 4] "
Checkpoint 13.3 [p. 13]
Refer to "2.7 Content negotiation [p. 13] "
Checkpoint 13.4 [p. 9]
Refer to "2.3 Alternative pages [p. 9] "

Guideline 14:

Checkpoint 14.1 [p. 42]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Title frames for easy orientation [p. 42] "
Checkpoint 14.2 [p. 43]
Refer to "2.2 Text equivalents and descriptions [p. 7] " and "Long descriptions of frames [p. 43] "
Checkpoint 14.3 [p. 46]
Refer to "Group form controls [p. 46] "
Checkpoint 14.4 [p. 47]
Refer to "Label form controls explicitly [p. 47] "
Checkpoint 14.5 [p. 47]
Refer to "Group menu options [p. 47] "

Guideline 15:

Checkpoint 15.1 [p. 26]
Refer to "3.6 Links [p. 26] "
Checkpoint 15.2 [p. 12]
Refer to "2.5 Navigation [p. 11] "
Checkpoint 15.3 [p. 12]
Refer to "2.5 Navigation [p. 11] "
Checkpoint 15.4 [p. 12]
Refer to "2.5 Navigation [p. 11] "
Checkpoint 15.5 [p. 12]
Refer to "2.5 Navigation [p. 11] "
Checkpoint 15.6 [p. 12]
Refer to "2.5 Navigation [p. 11] "
Checkpoint 15.7 [p. 12]
Refer to "2.5 Navigation [p. 11] "
Checkpoint 15.8 [p. 12]
Refer to "2.5 Navigation [p. 11] "

Checkpoint 15.9 [p. 15]
Refer to "2.9 Other topics [p. 15] "
Checkpoint 15.10 [p. 27]
Refer to "3.6 Links [p. 26] "

Guideline 16:

Checkpoint 16.1 [p. 12]
Refer to "2.6 Comprehension [p. 12] "
Checkpoint 16.2 [p. 12]
Refer to "2.6 Comprehension [p. 12] "
Checkpoint 16.3 [p. 11]
Refer to "2.5 Navigation [p. 11] "

8 Index of HTML elements and attributes

8.1 Elements

This index lists all elements in HTML 4.0 and whether they are defined in earlier versions of HTML. Elements that are deprecated in HTML 4.0 ([HTML40] [p. 50]) are followed by an asterisk (*). Elements that are obsolete in HTML 4.0 or don't exist in a W3C specification of HTML (2.0, 3.2, 4.0) do not appear in this table. An entry of "N/A" means that an element is not discussed in this document.

Name (links to 4.0)	**Also defined in**	**Related HTML topics**
A	2.0, 3.2	Keyboard access [p. 10] , Links [p. 26]
ABBR		Acronyms and abbreviations [p. 18]
ACRONYM		Acronyms and abbreviations [p. 18]
ADDRESS	2.0, 3.2	Metadata [p. 16]
APPLET*	3.2	Applets and other objects [p. 33]
AREA	3.2	Keyboard access [p. 10] , Images and image maps [p. 27] ,
B	2.0, 3.2	Text style [p. 38] , Emphasis [p. 18]
BASE	2.0, 3.2	Links [p. 26]
BASEFONT*	3.2	Fonts [p. 39]
BDO		N/A
BIG	3.2	Text style [p. 38]

BLOCKQUOTE	2.0, 3.2	Quotations [p. 18] , Text style [p. 38]
BODY	2.0, 3.2	N/A
BR	2.0, 3.2	Layout, positioning, and alignment [p. 39]
BUTTON		Keyboard access [p. 10] , Server-side image maps [p. 32] , Forms [p. 46]
CAPTION	3.2	Tables [p. 21]
CENTER*	3.2	Layout, positioning, and alignment [p. 39]
CITE	2.0, 3.2	N/A
CODE	2.0, 3.2	N/A
COL		Tables [p. 21]
COLGROUP		Tables [p. 21]
DD	2.0, 3.2	Lists [p. 19]
DEL		N/A
DFN	3.2	N/A
DIR*	2.0, 3.2	N/A
DIV	3.2	N/A
DL	2.0, 3.2	Lists [p. 19]
DT	2.0, 3.2	Lists [p. 19]
EM	2.0, 3.2	Emphasis [p. 18] , Text style [p. 38]
FIELDSET		Grouping form controls [p. 46]
FONT*	3.2	Fonts [p. 39]
FORM	2.0, 3.2	Forms [p. 46]
FRAME		Frames [p. 42]
FRAMESET		Frames [p. 42]
H1-H6	2.0, 3.2	Section headers [p. 17]
HEAD	2.0, 3.2	N/A
HR	2.0, 3.2	Rules and border [p. 40] , Section headers [p. 17]
HTML	2.0, 3.2	N/A

I	2.0, 3.2	Text style [p. 38] , Emphasis [p. 18]
IFRAME		Frames [p. 42]
IMG	2.0, 3.2	List bullets [p. 20] ,
INPUT	2.0, 3.2	Images and image maps [p. 27] , Keyboard access [p. 10] , Layout, positioning, and alignment [p. 39] , Rules and borders [p. 40]
INS		N/A
ISINDEX*	2.0, 3.2	N/A
KBD	2.0, 3.2	N/A
LABEL		Keyboard access [p. 10] , Form labels [p. 47]
LEGEND		Keyboard access [p. 10] ,
LI	2.0, 3.2	Lists [p. 19]
LINK	2.0, 3.2	Links [p. 26] , Metadata [p. 16]
MAP	3.2	Images and image maps [p. 27]
MENU*	2.0, 3.2	N/A
META	2.0, 3.2	Metadata [p. 16]
NOFRAMES		Frames [p. 42]
NOSCRIPT		Scripts [p. 48]
OBJECT		Keyboard access [p. 10] , Images and image maps [p. 27] , Applets and other objects [p. 33]
OL	2.0, 3.2	Lists [p. 19]
OPTGROUP		Forms [p. 46]
OPTION	2.0, 3.2	Forms [p. 46]
P	2.0, 3.2	N/A
PARAM	3.2	N/A
PRE	2.0, 3.2	Layout, positioning, and alignment [p. 39] , Text style [p. 38]
Q		Quotations [p. 18]
S		Text style [p. 38]
SAMP	2.0, 3.2	N/A

SCRIPT	3.2 (place-holder)	Scripts [p. 48]
SELECT	2.0, 3.2	Keyboard access [p. 10] , Forms [p. 46]
SMALL	3.2	Text style [p. 38]
SPAN		N/A
STRIKE*	3.2	Text style [p. 38]
STRONG	2.0, 3.2	Emphasis [p. 18] , Text style [p. 38]
STYLE	3.2 (place-holder)	Keyboard access [p. 37] ,
SUB	3.2	N/A
SUP	3.2	N/A
TABLE	3.2	Tables [p. 21]
TBODY		Tables [p. 21]
TD	3.2	Tables [p. 21]
TEXTAREA	2.0, 3.2	Keyboard access [p. 10] , Forms [p. 46]
TFOOT		Tables [p. 21]
TD	3.2	Tables [p. 21]
TD	3.2	Tables [p. 21]
THEAD		Tables [p. 21]
TITLE	2.0, 3.2	Metadata [p. 16]
TR	3.2	Tables [p. 21]
TT	2.0, 3.2	Text style [p. 38]
U	3.2	Text style [p. 38]
UL	2.0, 3.2	Lists [p. 19]
VAR	2.0, 3.2	N/A

8.2 Attributes

This index lists some attributes in HTML 4.0 that affect accessibility and what elements they apply to. Elements that are deprecated in HTML 4.0 ([HTML40] [p. 50]) are followed by an asterisk (*). Elements that are obsolete in HTML 4.0 or don't exist in a W3C specification of HTML (2.0, 3.2, 4.0) do not appear in this table.

Attributes that apply to most elements of HTML 4.0 are indicated as such; please consult the HTML 4.0 specification for the exact list of elements with this attribute.

Attributes that affect presentation [p. 7] , most of which are deprecated in HTML 4.0, are not are not listed in this table. Content developers should use style sheets instead of these attributes.

Attributes that associate scripts with events [p. 6] are not are not listed in this table. Please see the section on device-independent event handlers [p. 49] for more detail.

Name (links to HTML 4.0)	**Applies to elements**	**Related HTML topics**
abbr	TD, TH	
accesskey	A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA	
alt	APPLET*, AREA, IMG, INPUT	
axis	TD, TH	
class	Most elements	
for	LABEL	
headers	TD, TH	
hreflang	A, LINK	
id	Most elements	
lang	Most elements	
longdesc	IMG, FRAME, IFRAME	
name	FRAME (and others)	
scope	TD, TH	
style	Most elements	
summary	TABLE	
tabindex	A, AREA, BUTTON, INPUT, OBJECT, SELECT, TEXTAREA	
title	Most elements	
usemap	IMG, INPUT, OBJECT	