HTML Techniques for Web Content Accessibility Guidelines 1.0

W3C Working Draft 26 August 2000

This version:
http://www.w3.org/WAI/GL/WD-WCAG10-HTML-TECHS-20000826
(plain text, postscript, pdf, gzip tar file of HTML, zip archive of HTML)
Latest version:
http://www.w3.org/WAI/GL/WCAG10-HTML-TECHS
Previous version:
http://www.w3.org/WAI/GL/WD-WCAG10-HTML-TECHS-20000808
Editors:
Wendy Chisholm, W3C,
Gregg Vanderheiden, Trace R & D Center, University of Wisconsin -- Madison
Ian Jacobs, W3C

Abstract

This document describes techniques for satisfying the checkpoints of "HTML Techniques for Web Content Accessibility Guidelines 1.0" *[WCAG10] [p. 56]* when authoring HTML *[HTML4] [p. 56]* . The sections are organized by topic (and mirror the organization of the HTML 4.01 specification, *[HTML4] [p. 56]*).

An index of HTML elements and attributes [p. 48] provides information about all elements of HTML 4.01 and all attributes that affect accessibility directly. For each element, the index includes links to techniques that refer to it.

"Techniques for Web Content Accessibility Guidelines 1.0" *[WCAG10-TECHS] [p. 56]* contains additional techniques and references for other formats and languages.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

While HTML Techniques for Web Content Accessibility Guidelines 1.0 strives to be a stable document (as a W3C Recommendation), the current document is expected to evolve as technologies change and content developers discover more effective techniques for designing accessible Web sites and pages.

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 participants in the Web Content Accessibility Guidelines (WCAG) Working Group.

This document is part of a series of accessibility documents published by the Web Accessibility Initiative. WAI Accessibility Guidelines are produced as part of the WAI Technical Activity. The goal of the Web Content 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 wai-wcag-editor@w3.org.

To do

- Clean up all of the editor questions (marked by @@).

Table of Contents

1 Document structure and metadata

Content developers should use structural markup and use it according to specification. Structural elements and attribute (refer to the index of HTML elements and attributes [p. 48] to identify them) 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.

1.1 Metadata

Checkpoints in this section:

- 13.2 Provide metadata to add semantic information to pages and sites. [Priority 2]

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:

1.1.1 TITLE: The document title.

Note that the (mandatory) TITLE element, which only appears once in a document, is different from the "title [p. 53] " attribute, which applies to almost every HTML 4.01 element. Content developers should use the "title" attribute in accordance with the HTML 4.01 specification. For example, "title" should be used with links to provide information about the target of the link.

1.1.2 The ADDRESS element

This element can be used to provide information about the creator of the page.

1.1.3 The META element

Checkpoints in this section:

- 7.4 Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2] ,
- 7.5 Until user agents provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects. [Priority 2] .

This element can specify metadata for a document including keywords, and information about the author. Please refer to the section on automatic page refresh for information on why META should **not** be used to redirect or auto-refresh pages.

The following are **deprecated** HTML examples. The first changes the user's page at page at regular intervals. Content developers should **not** use this technique to simulate "push" technology. Developers cannot predict how much 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.

Deprecated example.

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

End example.

The following HTML example (using the META element) forwards the user from one page to another after a timeout. 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.

Deprecated example.

```
<HEAD>
<TITLE>Don't use this!</TITLE>
<META http-equiv="refresh" content="5;
         http://www.acme.com/newpage">
</HEAD>
<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>
```

End example.

1.1.4 The !DOCTYPE statement

Checkpoints in this section:

- 3.2 Create documents that validate to published formal grammars. [Priority 2]

Validating to a published formal grammar and declaring that validation at the beginning of a document let's the user know that the structure of the document is sound. It also let's the user agent know where to look for semantics if it needs to. The W3C Validation Service validates documents against a whole list of published grammars.

It is preferable to validate to W3C grammars. Refer to the Accessibility Reviewed Technologies.

@@link to XML techniques when ready

1.1.5 The LINK element and navigation tools

Checkpoints in this section:

- 13.9 Provide information about document collections (i.e., documents comprising multiple pages.). [Priority 3]

Content developers should use the LINK [p. 50] element and link types (refer to *[HTML4] [p. 56]* , section 6.12) to describe document navigation mechanisms and organization. Some user agents may synthesize navigation tools or allow ordered printing of a set of documents based on such markup.

Example.

The following LINK [p. 50] elements might be included in the head of chapter 2 of a book:

```
<LINK rel="Next" href="chapter3">
<LINK rel="Prev" href="chapter1">
<LINK rel="Start" href="cover">
<LINK rel="Glossary" href="glossary">
```

End example.

1.1.6 The LINK element and alternative documents

Checkpoints in this section:

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

The LINK element may also be used to designate alternative documents. Browsers should load the alternative page automatically based on the user's browser type and preferences. For example, use the LINK element as follows:

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"
      media="aural, braille, tty">
</HEAD>
<BODY><P>...</BODY>
```

End example.

1.2 Structural grouping

Checkpoints in this section:

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

The following HTML 4.01 mechanisms group content and make it easier to understand.:

- Use FIELDSET [p. 50] to group form controls into semantic units [p. 41] and describe the group with the LEGEND [p. 50] element.
- Use OPTGROUP [p. 51] to organize long lists of menu options into smaller groups. [p. 42] .
- Use tables for tabular data [p. 16] and describe the table with CAPTION [p. 49] .
- Group table rows and columns [p. 16] with THEAD [p. 51] , TBODY [p. 51] , TFOOT [p. 51] , and COLGROUP [p. 49] .
- Nest lists [p. 13] with UL [p. 52] , OL [p. 51] , and DL [p. 49] .
- Use section headers (H1 [p. 50] - H6) to create structured documents and break up long stretches of text. Refer to the following section for more information.
- Break up lines of text into paragraphs (with the P [p. 51] element).
- Group related links. Refer to the section Grouping and bypassing links [p. 24]

All of these grouping mechanisms should be used when appropriate and natural, i.e., when the information lends itself to logical groups. Content developers should not create groups randomly, as this will confuse all users.

1.2.1 Section headers

Checkpoints in this section:

- 3.5 Use header elements to convey document structure and use them according to specification. [Priority 2]

Long documents are often divided into a variety of chapters, chapters have subtopics and subtopics are divided into various sections, sections into paragraphs, etc. These semantic chunks of information make up the structure of the document.

Sections should be introduced with the HTML header elements (H1 [p. 50] -H6). Other markup may complement these elements to improve presentation (e.g., the HR [p. 50] element to create a horizontal dividing line), but visual presentation is not sufficient to identify document sections.

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 to change font styles for example.

Note that in HTML, heading elements (H1 - H6) only start sections, they don't contain them as element content. The following HTML markup shows how style sheets may be used to control the appearance of a header and the content that follows:

Example.

```
<HEAD>
<TITLE>Cooking techniques</TITLE>
<STYLE type="text/css">
   /* Indent header and following content */
   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 section ...
</DIV>
```

End example.

See also the section on links [p. 22] .

2 Language information

2.1 Identifying changes in language

Checkpoints in this section:

- 4.1 Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions). [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 [p. 53] " 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.

Identifying changes in language are important for a number of reasons:

1. Users who are reading the document in braille will be able to substitute the appropriate control codes (markup) where language changes occur to ensure that the braille translation software will generate the correct characters (accented characters, for instance). These control codes also prevent braille contractions from being generated, which could further confuse the user. Braille contractions combine commonly used groups of characters that usually appear in multiple cells into a single cell. For example, "ing" which usually takes up three cells (one for each character) can be contracted into a single cell.
2. Similarly, speech synthesizers that "speak" multiple languages will be able to generate the text in the appropriate accent with proper pronunciation. If changes are not marked, the synthesizer will try its best to speak the words in the primary language it works in. Thus, the French word for car, "voiture" would be pronounced "voter" by a speech synthesizer that uses English as its primary language.
3. Users who are unable to translate between languages themselves, will be able to have unfamiliar languages translated by machine translators.

2.2 Identifying the primary language

Checkpoints in this section:

- 4.3 Identify the primary natural language of a document. [Priority 3]

It is also good practice to identify the primary language of a document, either with markup (as shown below) or through HTTP headers.

Example.

```
<HTML lang="fr">
   ....rest of an HTML document written in French...
</HTML>
```

End example.

3 Text markup

The following sections discuss ways to add structure to pieces of text.

3.1 Emphasis

Checkpoints in this section:

- 3.3 Use style sheets to control layout and presentation. [Priority 2]

The proper HTML elements should be used to mark up emphasis: EM [p. 50] and STRONG [p. 51] . The B [p. 49] and I [p. 50] 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.)

3.2 Acronyms and abbreviations

Checkpoints in this section:

- 4.2 Specify the expansion of each abbreviation or acronym in a document where it first occurs. [Priority 3]

Mark up abbreviations and acronyms with ABBR [p. 49] and ACRONYM [p. 49] and use "title [p. 53] " to indicate the expansion:

Example.

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

End example.

This also applies to shortened phrases used as headings for table row or columns. If a heading is already abbreviated provide the expansion in ABBR. If a heading is long, you may wish to provide an abbreviation, as described in Data Tables [p. 16] .

Example.

```
...
<TH>First name</TH>
<TH><ABBR title="Social Security Number">SS#</ABBR>
...
```

End example.

3.3 Quotations

Checkpoints in this section:

- 3.7 Mark up quotations. Do not use quotation markup for formatting effects such as indentation. [Priority 2]

The Q [p. 51] and BLOCKQUOTE [p. 49] elements mark up inline and block quotations, respectively.

Example.

This example marks up a longer quotation with BLOCKQUOTE [p. 49] :

```
<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.

3.4 Markup and style sheets rather than images: The example of math

Checkpoints in this section:

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

Using markup (and style sheets) where possible rather than images (e.g., a mathematical equation) promotes accessibility for the following reasons:

- Text may be magnified or interpreted as speech or braille.
- Search engines can use text information.

As an example, consider these techniques for putting mathematics on the Web:

- Ensure that users know what variables represent, for example, in the equation "F = m * a", indicate that F= Force, m = mass, a = acceleration.
- For straightforward equations, use characters, as in "x + y = z"
- For more complex equations, mark them up with MathML (*[MATHML] [p. 56]*) or TeX. **Note.** MathML can be used to create very accessible documents but currently is not as widely supported or used as TeX.
- Provide a text description of the equation and, where possible, use character entity references to create the mathematical symbols. A text equivalent must be provided if the equation is represented by one or more images.

TeX is commonly used to create technical papers which are then converted to HTML for publication on the Web. However, converters tend to generate images, use deprecated markup, and use tables for layout. Consequently, content providers should:

1. Make the original TeX (or LaTeX) document available on the Web. There is a system called "AsTeR" (*[ASTER] [p. 57]*) that can create an auditory rendition of TeX and LaTeX documents. Also, IBM has a plug-in for Netscape and Internet Explorer that reads TeX/LaTeX documents and some of MathML (refer to *[HYPERMEDIA] [p. 57]*). **Note.** These tools work primarily in the English environment and my not work so well with speech synthesizers whose primary language is not English.
2. Ensure that the HTML created by the conversion process is accessible. Provide a single description of the equation (rather than "alt" text on every generated image as there may be small images for bits and pieces of the equation).

3.5 Eight other structural elements (to identify citations, code fragments, deleted text, and others)

The HTML 4.01 specification defines the following structural elements for miscellaneous markup needs:

CITE [p. 49]
: Contains a citation or a reference to other sources.

DFN [p. 49]
: Indicates that this is the defining instance of the enclosed term.

CODE [p. 49]
: Designates a fragment of computer code.

SAMP [p. 51]
: Designates sample output from programs, scripts, etc.

KBD [p. 50]
: Indicates text to be entered by the user.

VAR [p. 52]
: Indicates an instance of a variable or program argument.

INS [p. 50]
: Indicates text inserted into a document.

DEL [p. 49]
: Indicates text deleted from a document.

4 Lists

Checkpoints in this section:

- 3.6 Mark up lists and list items properly. [Priority 2]

The HTML list elements DL [p. 49] , UL [p. 52] , and OL [p. 51] should only be used to create lists, not for formatting effects such as indentation. Refer to information on css and tables for layout in the CSS Techniques chapter.

Ordered lists help non-visual users navigate. Non-visual users may "get lost" in lists, especially in nested lists and those that do not indicate the specific nest level for each list item. Until user agents provide a means to identify list context clearly (e.g., by supporting the ':before' pseudo-element in CSS2), content developers should include contextual clues in their lists.

For numbered lists, compound numbers are more informative than simple numbers. Thus, a list numbered "1, 1.1, 1.2, 1.2.1, 1.3, 2, 2.1," provides more context than the same list without compound numbers, which might be formatted as follows:

```
1.
  1.
  2.
    1.
  3.
2.
  1.
```

and would be spoken as "1, 1, 2, 1, 2, 3, 2, 1", conveying no information about list depth.

[CSS1] [p. 56] and *[CSS2] [p. 56]* allow users to control number styles (for all list, not just ordered) through user style sheets.

Example.

The following CSS2 style sheet shows how to specify compound numbers for nested lists created with either UL or OL elements. Items are numbered as "1", "1.1", "1.1.1", etc.

```
<STYLE type="text/css">
   UL, OL { counter-reset: item }
   LI { display: block }
   LI:before { content: counters(item, "."); counter-increment: item }
</STYLE>
```

End example.

Until either CSS2 is widely supported or user agents allow users to control rendering of lists through other means, authors should consider providing contextual clues in unnumbered nested lists. Non-visual users may have difficulties knowing where a list begins and ends and where each list item starts. For example, if a list entry wraps to the next line on the screen, it may appear to be two separate items in the list. This may pose a problem for legacy screen readers.

4.1 Use style sheets to change list bullets

To change the "bullet" style of unordered list items created with the LI [p. 50] element, use style sheets. In CSS, it is possible to specify a fallback bullet style (e.g., 'disc') if a bullet image cannot be loaded.

Example.

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

End example.

To further ensure that users understand differences between list items indicated visually, content developers should provide a text label before or after the list item phrase:

Example.

In this example, new information is communicated through text ("New"), font style (bold), and color (yellow bullet, red text on yellow background).

```
<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) disc }
</STYLE>
</HEAD>
<BODY>
<UL>
   <LI class="newbullet">Roth IRA <SPAN class="newtext">New</SPAN></LI>
   <LI> 401(k)</LI>
</UL>
</BODY>
```

End example.

4.1.1 Images used as bullets

Checkpoints in this section:

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

Avoid using images as bullets in definition lists created with DL [p. 49] , DT [p. 50] , and DD [p. 49] . However, if this method is used, be sure to provide a text equivalent for the images.

Deprecated example.

```
<HEAD>
<TITLE>Deprecated example using image in DL lists</TITLE>
</HEAD>
<BODY>
<DL>
   <DD><IMG src="star.gif" alt="* ">Audrey
   <DD><IMG src="star.gif" alt="* ">Laurie
   <DD><IMG src="star.gif" alt="* ">Alice
</DL>
```

End example.

Content developers should avoid list styles where bullets provide additional (visual) information. However, if this is done, be sure to provide a text equivalent 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>
```

End example.

5 Tables

This section discusses the accessibility of tables and elements that one can put in a TABLE [p. 51] element. Two types of tables are discussed: tables used to organize data, and tables used to create a visual layout of the page.

5.1 Tables of data

Content developers may make HTML 4.01 data tables more accessible in a number of ways:

- Providing summary information
- Identifying row and column information

5.1.1 Providing summary information

Checkpoints in this section:

- 5.5 Provide summaries for tables. [Priority 3]
- 5.6 Provide abbreviations for header labels. [Priority 3]

- Provide a caption via the CAPTION [p. 49] element. A table caption describes the nature of the table in one to three sentences. Two examples:
  1. "Cups of coffee consumed by each senator."
  2. "Who spends the most on pollution cleanup?"

  A caption may not always be necessary.
- If a CAPTION is not provided, use the "title" attribute on the TABLE element to

describe the nature of the table in a few words.

- Provide a summary via the "summary [p. 53] " attribute. Summaries are especially useful for non-visual readers. A summary of the relationships among cells is especially important for tables with nested headings, cells that span multiple columns or rows, or other relationships that may not be obvious from analyzing the structure of the table but that may be apparent in a visual rendering of the table. A summary may also describe how the table fits into the context of the current document. If no caption is provided, it is even more critical to provide a summary. Two examples:
    1. "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."
    2. "Total required by pollution control standards as of January 1, 1971. Commercial category includes stores, insurance companies and banks. The table is divided into two columns. The left-hand column is 'Total investment required in billions of dollars'. The right--hand column is 'Spending' and is divided into three sub-columns. The first sub-column is titled '1970 actual in millions of dollars', the second is '1971 planned in millions of dollars', and the third is 'Percent change, 1970 versus 1971.' The rows are industries." [NBA, 1996].
- Provide terse substitutes for header labels with the "abbr [p. 53] " 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.

@@separate out the examples?

5.1.2 Identifying rows and column information

Checkpoints in this section:

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

- Identify structural groups of rows (THEAD [p. 51] for repeated table headers, TFOOT [p. 51] for repeated table footers, and TBODY [p. 51] for other groups of rows) and groups of columns (COLGROUP [p. 49] and COL [p. 49]).
- Label table elements with the "scope [p. 53] ", "headers [p. 53] ", and "axis [p. 53] " attributes so that future browsers and assistive technologies will be able to select data from a table by filtering on categories.
- Do not use PRE [p. 51] to create a tabular layout of text -- use the TABLE element so that assistive technologies may recognize that it is a table.
- For information about table headers, refer to the table header algorithm and discussion in the HTML 4.01 Recommendation (*[HTML4] [p. 56]* , section 11.4.3).

Example.

This example shows how to associate data cells (created with TD [p. 51]) with their corresponding headers by means of the "headers [p. 53] " 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.

```
<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="header1">Name</TH>
      <TH id="header2">Cups</TH>
      <TH id="header3" abbr="Type">Type of  Coffee</TH>
      <TH id="header4">Sugar?</TH>
  <TR>
      <TD headers="header1">T. Sexton</TD>
      <TD headers="header2">10</TD>
      <TD headers="header3">Espresso</TD>
      <TD headers="header4">No</TD>
  <TR>
      <TD headers="header1">J. Dinnen</TD>
      <TD headers="header2">5</TD>
      <TD headers="header3">Decaf</TD>
     <TD headers="header4">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

[Description of coffee table]

The next example associates the same header (TH [p. 51]) and data (TD [p. 51]) cells as before, but this time uses the "scope [p. 53] " attribute rather than "headers [p. 53] ". "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 [p. 53] " attribute.

Example.

```
  <TABLE border="1">
    <CAPTION>Travel Expense Report</CAPTION>
    <TR>
         <TH></TH>
         <TH id="header2" axis="expenses">Meals
         <TH id="header3" axis="expenses">Hotels
         <TH id="header4" axis="expenses">Transport
         <TD>subtotals</TD>
    <TR>
         <TH id="header6" axis="location">San Jose
         <TH> <TH> <TH> <TD>
    <TR>
        <TD id="header7" axis="date">25-Aug-97
        <TD headers="header6 header7 header2">37.74
        <TD headers="header6 header7 header3">112.00
        <TD headers="header6 header7 header4">45.00
        <TD>
    <TR>
        <TD id="header8" axis="date">26-Aug-97
        <TD headers="header6 header8 header2">27.28
        <TD headers="header6 header8 header3">112.00
        <TD headers="header6 header8 header4">45.00
        <TD>
    <TR>
        <TD>subtotals
        <TD>65.02
        <TD>224.00
        <TD>90.00
```

```
        <TD>379.02
    <TR>
        <TH id="header10" axis="location">Seattle
        <TH> <TH> <TH> <TD>
    <TR>
        <TD id="header11" axis="date">27-Aug-97
        <TD headers="header10 header11 header2">96.25
        <TD headers="header10 header11 header3">109.00
        <TD headers="header10 header11 header4">36.00
        <TD>
    <TR>
        <TD id="header12" axis="date">28-Aug-97
        <TD headers="header10 header12 header2">35.00
        <TD headers="header10 header12 header3">109.00
        <TD headers="header10 header12 header4">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. [Description of travel table]

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**

5.2 Tables for layout

Checkpoints in this section:

- 5.3 Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version). [Priority 2]
- 5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting. [Priority 2]

Authors should use style sheets for layout and positioning. However, when it is necessary to use a table for layout, the table must linearize in a readable order. When a table is linearized, the contents of the cells become a series of paragraphs (e.g., down the page) one after another. Cells should make sense when read in row order and should include structural elements (that create paragraphs, headers, lists, etc.) so the page makes sense after linearization.

Also, when using tables to create a layout, do not use structural markup to create visual formatting. For example, the TH (table header) element, is usually displayed visually as centered, and bold. If a cell is not actually a header for a row or column of data, use style sheets or formatting attributes of the element.

5.3 Linearizing tables

Checkpoints in this section:

- 10.3 Until user agents (including assistive technologies) render side-by-side text correctly, provide a linear text alternative (on the current page or some other) for *all* tables that lay out text in parallel, word-wrapped columns. [Priority 3]

Tables used to lay out pages where cell text wraps pose problems for older screen readers that do not interpret the source HTML or browsers that do not allow navigation of individual table cells. These screen readers will 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.
```

It is usually very simple to linearize a table used to layout a page - simply strip the table markup from the table. There are several tools that do this, and it is becoming more common for screen readers and some browsers to linearize tables.

However, linearizing data tables [p. 16] requires a different strategy. Since data cells rely on the information provided by surrounding and header cells, the relationship information that is available visually needs to be translated into the linear table.

For example, specify the column layout order. The natural language writing direction may affect column layout and thus the order of a linearized table. The "dir [p. 53] " attribute specifies column layout order (e.g., dir="rtl" specifies right-to-left layout).

@@also the case for layout tables?

@@example

again, there are tools to help produce linearized versions of data tables. @@link

@@this copied from earlier in the document. smooth it out. This markup will also help browsers linearize tables (also called table "serialization"). A row-based linear version may be created by reading the row header, then preceding each cell with the cell's column header. Or, the linearization might be column-based. Future browsers and assistive technologies will be able to automatically translate tables into linear sequences or navigate a table cell by cell if data is labeled appropriately. The WAI Evaluation and Repair working group is tracking the progress of tools as well as developing their own that will allow users to linearize or navigate tables cell by cell. Refer to *[WAI-ER] [p. 57]* .

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.

5.4 Backward compatibility issues for tables

@@checkpoint that this can fit under?

In HTML 3.2 browsers, the rows of a TFOOT [p. 51] element will appear before the table body.

6 Links

6.1 Link text

Checkpoints in this section:

- 13.1 Clearly identify the target of each link. [Priority 2]

Good link text should not be overly general; don't use "click here." Not only is this phrase device-dependent (it implies a pointing device) it says nothing about what is to be found if the link if followed. Instead of "click here", link text should indicate the nature of the link target, as in "more information about sea lions" or "text-only version of this page". Note that for the latter case (and other format- or language-specific documents), content developers are encouraged to use content negotiation instead, so that users who prefer text versions will have them served automatically.

In addition to clear link text, content developers may specify a value of the "title" attribute that clearly and accurately describes the target of the link.

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.

If two or more links refer to different targets but share the same link text, distinguish the links by specifying a different value for the "title [p. 53] " attribute of each A [p. 49] element.

"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. To get an overview of a page or to quickly find a link, these users will often tab from one link to the next or review a list of available links on a page.

Thus, for a series of related links, include introductory information in the first link, then distinguishing information in the links that follow. This will provide context information for users reading them in sequence.

Example.

```
<A href="my-doc.html">My document is available in HTML</A>,
<A href="my-doc.pdf" title="My document in PDF">PDF</A>,
<A href="my-doc.txt" title="My document in text">plain text</A>
```

End example.

6.1.1 Text for images used as links

Checkpoints in this section:

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

When an image is used as the content of a link, specify a text equivalent for the image.

Example.

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

End example.

Or, if you provide link text, use a space as the "alt" attribute value of the IMG element. Note that this text will appear on the page next to the image.

Example.

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

End example.

6.2 Grouping and bypassing links

Checkpoints in this section:

- 13.6 Group related links, identify the group (for user agents), and, until user agents do so, provide a way to bypass the group. [Priority 3]
- 10.5 Until user agents (including assistive technologies) render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links. [Priority 3]

When links are grouped into logical sets (for example, in a navigation bar that appears on every page in a site) they should be marked up as a unit. Navigation bars are usually the first thing someone encounters on a page. For users with speech synthesizers, this means having to hear a number of links on every page before reaching the interesting content of a page. There are several ways to allow users to bypass groups of links (as users with vision do when they see the same set on each page):

- Include a link that allows users to skip over the set of navigation links.
- Provide a style sheet that allows users to hide the set of navigation links.
- Use the HTML 4.01 MAP element to group links, then identify the group with the "title [p. 53] " attribute.

In the future, user agents may allow users to skip over elements such as navigation bars.

Example.

In this example, the MAP [p. 50] element groups a set of links, the "title" attribute identifies it as a navigation bar , and a link at the beginning of the group links to the anchor after the group. Also, note that the links are separated by non-link, printable characters (surrounded by spaces).

```
<BODY>
  <MAP title="Navigation Bar">
    <P>
    [<A href="#how">Bypass navigation bar</A>]
    [<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>]
      </P>
    </MAP>
    <H1><A name="how">How to use our site</A></H1>
  <!-- content of page -->
  </BODY>
```

End example.

6.3 Keyboard access

Checkpoints in this section:

- 9.4 Create a logical tab order through links, form controls, and objects. [Priority 3]
- 9.5 Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls. [Priority 3]

Keyboard access to active elements of a page is important for many users who cannot use a pointing device. User agents may include features that allow users to bind keyboard strokes to certain actions. HTML 4.01 allows content developers to specify keyboard shortcuts in documents via the "accesskey [p. 53] " attribute.

Example.

In this example, if the user activates the "C" key, the link will be followed.

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

End example.

Note: Until user agents provide an overview of which key bindings are available, provide information on the key bindings.

6.4 Anchors and targets

Checkpoints in this section:

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

@@include info about using target attribute on A to generate new windows. test that an unknown target causes a new window to open.

7 Images and image maps

The following sections discuss accessibility of images (including simple animations such as GIF animations) and image maps.

For information about math represented as images, refer to the section on using text markup and style sheets rather than images [p. 12] .

7.1 Short text equivalents for images ("alt-text")

Checkpoints in this section:

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

When using IMG [p. 50] , specify a short text equivalent with the "alt [p. 53] " attribute. **Note.** The value of this attribute is referred to as "alt-text".

Example.

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

End example.

When using OBJECT [p. 50] , specify a text equivalent in the body of the OBJECT element:

Example.

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

End example.

7.2 Long descriptions of images

Checkpoints in this section:

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

When a short text equivalent does not suffice to adequately convey the function or role of an image, provide additional information in a file designated by the "longdesc [p. 53] " attribute:

Example.

```
<IMG src="97sales.gif" alt="Sales for 1997"
     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 user agents 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, specify longer text equivalent within the element’s content:

Example.

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

End example.

Note that OBJECT content, unlike "alt" text, can include markup. Thus, content developers can provide a link to additional information from within the OBJECT 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.

7.2.1 Invisible d-links

Note. Invisible d-links are deprecated in favor of the "longdesc" attribute.

An invisible d-link is a small (1-pixel) or transparent image whose "alt [p. 53] " attribute value is "D-link" or "D" and is part of the content of an A [p. 49] element. Like other d-links, it refers to a text equivalent of the associated image. Like other links, users can tab to it. Invisible d-links thus provide a (temporary) solution for designers who wish to avoid visible d-links for stylistic reasons.

7.3 Ascii art

Checkpoints in this section:

- 13.10 Provide a means to skip over multi-line ASCII art. [Priority 3]

Avoid ascii art (character illustrations) and use real images instead since it is easier to supply a text equivalent for images. 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 [skip over ascii figure [p. 28] or consult a description of chart]:

Example.

```
  %   __ __ __ __ __ __ __ __ __ __ __ __ __ __
100 |             *                             |
 90 |                *  *                       |
 80 |          *           *                    |
 70 |             @           *                 |
 60 |          @                 *              |
 50 |       *        @              *           |
 40 |                   @              *        |
 30 |    *  @              @  @           *     |
 20 |                                           |
 10 |    @                       @  @  @  @     |
      0  5 10 15 20 25 30 35 40 45 50 55 60 65 70
      Flash frequency (Hz)
```

End example.

Another option for smaller ascii art is to use an ABBR [p. 49] element with "title [p. 53] ".

Example.

```
<P><ABBR title="smiley in ascii art">:-)</ABBR>
```

End example.

If the ASCII art is complex, ensure that the text equivalent adequately describes it.

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

7.4 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 may be 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 [p. 50] element. 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 a text equivalent.

Content developers should create client-side image maps (with "usemap [p. 53] ") rather than server-side image maps (with "ismap [p. 54] ") because server-side image maps require a specific input device. If server-side image maps must be used (e.g., because the geometry of a region cannot be represented with values of the shape [p. 54] attribute), authors must 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. If you must use a server-side image map, please consult the section on server-side image maps [p. 31]

7.4.1 Text equivalents for client-side image maps

Checkpoints in this section:

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

Provide text equivalents for image maps since they convey visual information. As with other links, the link text should make sense when read out of context. Refer to the section on Link Text [p. 22] for information on writing good link text. Users may also want keyboard shortcuts to access frequently followed links. Refer to the section on Keyboard access to links [p. 25] .

If AREA [p. 49] is used to create the map, use the "alt [p. 53] " 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 following example illustrates the same idea, but uses OBJECT [p. 50] 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.

7.4.2 Redundant text links for client-side image maps

Checkpoints in this section:

- 1.5 Until user agents render text equivalents for client-side image map links, provide redundant text links for each active region of a client-side image map. [Priority 3]

In addition to providing a text equivalent, provide redundant textual links. If the A [p. 49] 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="276,0,276,28,100,200,50,50,276,0">
       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 older screen readers from reading several adjacent links as a single link as well as helps sighted users distinguish between links visually.

Content developers should make sure they include printable characters (such as brackets or a vertical bar (|)) surrounded by spaces between adjacent text links. The problem does not occur if images have been used as links; The alt-text will not be read as a single link because of the place-holding images that graphical browsers use when images are not loaded. Refer to the section Grouping and bypassing links [p. 24] for more information.

7.4.3 Client-side versus server-side image maps

Checkpoints in this section:

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

@@discuss reasoning why we suggest client- vs. server-side image maps. Discuss tools to use to convert server-side to client-side. Find a good example of a server-side image map.

7.4.4 Server-side image maps

Checkpoints in this section:

- 1.2 Provide redundant text links for each active region of a server-side image map. [Priority 1]

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

- Include the alternative links within the body of an OBJECT [p. 50] element (refer to the previous example illustrating links in the OBJECT element [p. 30]).
- If IMG [p. 50] is used to insert the image, provide an alternative list of links after it and indicate the existence and location of the alternative list (e.g., via that "alt [p. 53] " attribute).

  Example.

  ```
   <A href="http://myserver.com/cgi-bin/imagemap/my-map">
   <IMG src="welcome.gif" alt="Welcome! (Text links follow)" ismap>
   </A>

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

  End example.

- If other approaches don't make the image map accessible, create an alternative page that is accessible.

Server-side and client-side image maps may be used as submit buttons in Forms. For more information, refer to the section Graphical buttons [p. 43] .

7.5 Color in images

Checkpoints in this section:

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

@@discuss contrast issues with regard to images

7.6 Animated images

Checkpoints in this section:

- 7.3 Until user agents allow users to freeze moving content, avoid movement in pages. [Priority 2]

@@discuss avoiding movement caused by animated gifs.

8 Applets and other programmatic objects

While applets may be included in a document with either the APPLET [p. 49] or OBJECT [p. 50] element, OBJECT [p. 50] is the preferred method.

8.1 Text and non-text equivalents for applets and programmatic objects

Checkpoints in this section:

- 1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes*: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. [Priority 1]
- 6.2 Ensure that equivalents for dynamic content are updated when the dynamic content changes. [Priority 1]
- 6.3 Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page. [Priority 1]

If OBJECT [p. 50] is used, provide a text equivalent in the content of the element:

Example.

```
<OBJECT classid="java:Press.class" width="500" height="500">
      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 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.

If APPLET [p. 49] is used, provide a text equivalent with the "alt [p. 53] " attribute *and* in the content in the APPLET element. This enables the content 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>
```

End example.

8.2 Directly accessible applets

Checkpoints in this section:

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

If an applet (created with either OBJECT [p. 50] or APPLET [p. 49]) 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.

If an applet creates motion, developers should provide a mechanism for freezing this motion (for an example, refer to *[TRACE] [p. 57]*). Also, please refer to the next section for information about making audio and video presentations accessible.

For more information about developing accessible applets, please refer to *[JAVAACCESS] [p. 57]* and *[IBMJAVA] [p. 57]* . These companies have been developing an Accessibility API as well as making the Java Swing classes accessible.

Related checkpoints:

- 3.4 Use relative rather than absolute units in markup language attribute values and style sheet property values. [Priority 2] ,
- 6.4 For scripts and applets, ensure that event handlers are input device-independent. [Priority 2] ,
- 6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page. [Priority 2] ,
- 7.1 Until user agents allow users to control flickering, avoid causing the screen to flicker. [Priority 1] ,
- 7.2 Until user agents allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off). [Priority 2] ,
- 7.3 Until user agents allow users to freeze moving content, avoid movement in pages. [Priority 2] ,
- 7.4 Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2] ,
- 9.2 Ensure that any element that has its own interface can be operated in a device-independent manner. [Priority 2] ,
- and 10.1 Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current

window without informing the user. [Priority 2] .

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

9 Audio and video

9.1 Audio information

9.2 Text equivalents for multimedia

Checkpoints in this section:

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

When necessary, a text equivalent should be provided for visual information to enable understanding of the page. For example, consider a repeating animation that shows cloud cover and precipitation as part of a weather status report. Since the animation is supplementing the rest of the weather report (that is presented in natural language - text), a less verbose description of the animation is necessary. However, if the animation appears in a pedagogical setting where students are learning about cloud formations in relation to land mass, then the animation ought to be described for those who can not view the animation but who also want to learn the lesson.

9.3 Embedding multimedia objects

Other objects, such as those requiring a plug-in, should also use the OBJECT [p. 50] element. However, for backward compatibility with Netscape browsers, use the proprietary EMBED element within the OBJECT element as follows:

Deprecated example.

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

    <NOEMBED>
        <IMG alt="Still from Movie"
```

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

  </OBJECT>
```

End example.

For more information refer to *[MACROMEDIA] [p. 57]* .

10 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 (with the FRAMESET [p. 50] , FRAME [p. 50] , and IFRAME [p. 50] elements) 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. 40] that uses HTML 4.01 and CSS and addresses many of the limitations of today's frame implementations.

10.1 Providing a frame title

Checkpoints in this section:

- 12.1 Title each frame to facilitate frame identification and navigation. [Priority 1]

Example.

Use the "title" attribute to name frames.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 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.

10.2 Describing frame relationships

Checkpoints in this section:

- 1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes*: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. [Priority 1]
- 12.2 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 4.01 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 text equivalent will no longer apply. Also, links to descriptions of a frame should be provided along with other alternative content in the NOFRAMES [p. 50] element of a FRAMESET [p. 50] .

10.3 Writing for browsers that do not support FRAME

Checkpoints in this section:

- 1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes*: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. [Priority 1]
- 6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page. [Priority 2]

Example.

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

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 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.

10.4 Frame sources

Checkpoints in this section:

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

Content developers must provide text equivalents 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. Thus, 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 4.01 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.

The following deprecated example should be avoided since it inserts IMG [p. 50] directly in a frame:

Deprecated example.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 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").

End example.

10.5 Using FRAME targets

Checkpoints in this section:

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

Content developers should avoid specifying a new window as the target of a frame with target="_blank".

10.6 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 [p. 50] markup:

   Example.

   ```
   <P>
   <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.

10.6.1 Sizing frames with relative units

Checkpoints in this section:

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

In the previous examples, note that frame sizes are specified in percentage. When a user resizes the window, the frames will adjust accordingly and remain readable.

11 Forms

This section discusses the accessibility of forms and form controls that one can put in a FORM [p. 50] element.

11.1 Keyboard access to forms

Checkpoints in this section:

- 9.4 Create a logical tab order through links, form controls, and objects. [Priority 3]
- 9.5 Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls. [Priority 3]

In the next example, we specify a tabbing order among elements (in order, "field2", "field1", "submit") with "tabindex [p. 53] ":

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.

This example assigns "U" as the accesskey (via "accesskey [p. 53] "). 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.

11.2 Grouping form controls

Checkpoints in this section:

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

Content developers should group information [p. 8] where natural and appropriate. When form controls can be grouped into logical units, use the FIELDSET [p. 50] element and label those units with the LEGEND [p. 50] element:

Example.

```
<FORM action="http://example.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.

11.2.1 Grouping menu options

Content developers should group information [p. 8] where natural and appropriate. For long lists of menu selections (which may be difficult to track), content developers should group SELECT [p. 51] items (defined by OPTION [p. 51]) into a hierarchy using the OPTGROUP [p. 51] element. Specifies a label for the group of options with the label [p. 53] attribute on OPTGROUP.

Example.

```
<FORM action="http://example.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.

11.3 Labeling form controls

Checkpoints in this section:

- 12.4 Associate labels explicitly with their controls. [Priority 2]
- 10.2 Until user agents support explicit associations between labels and form controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned. [Priority 2]

An example of LABEL [p. 50] used with "for [p. 53] " in HTML 4.01 is given in the previous section.

A label is implicitly associated with its form control either through markup or positioning on the page. The following example shows how a label and form control may be implicitly associated with markup.

Example.

```
<LABEL for="firstname">First name:
  <INPUT type="text" id="firstname" tabindex="1">
</LABEL>
```

End example.

11.4 Graphical buttons

Checkpoints in this section:

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

Using images to decorate buttons allows developers to make their forms unique and easier to understand. Using an image for a button (e.g., with the INPUT [p. 50] element or BUTTON [p. 49]) is not inherently inaccessible - assuming a text equivalent is provided for the image.

However, a graphical form submit button created with INPUT [p. 50] , type="image" creates a type of server-side image map. Whenever the button is clicked with a mouse, the x and y coordinates of the mouse click are sent to the server as part of the form submission.

In the Image and Image Maps [p. 26] section, we discuss why server-side images ought to be avoided, and suggest using client-side image maps instead. In HTML 4.01, graphical buttons may now be client-side image maps. To preserve the functionality provided by the server, authors have the following options, as stated in the HTML 4.01 Recommendation (*[HTML4] [p. 56]* , section 17.4.1):

> If the server takes different actions depending on the location clicked, users of non-graphical browsers will be disadvantaged.
> For this reason, authors should consider alternate approaches:
>
> - Use multiple submit buttons (each with its own image) in place of a single graphical submit button. Authors may use style sheets to control the positioning of these buttons.
> - Use a client-side image map together with scripting.

11.5 Techniques for specific controls

Checkpoints in this section:

- 10.4 Until user agents handle empty controls correctly, include default, place-holding characters in edit boxes and text areas. [Priority 3]

Example.

Some legacy assistive technologies require initial text in form controls such as TEXTAREA [p. 51] in order to function properly.

```
<FORM action="http://example.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 a text equivalent for images used as "submit" buttons:

Example.

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

End example.

Also refer to the section on keyboard access since this applies to form controls.

11.6 Backward compatibility issues for forms

In some HTML 3.2 browsers,

- The BUTTON [p. 49] element does not appear
- INPUT [p. 50] with type="button"> will appear as a text input field

12 Scripts

This section discusses the accessibility of scripts included in a document via the SCRIPT [p. 51] element.

@@this section needs work. plus, is this the best place for this discussion? should we have a separate "non-w3c" module?

12.1 Graceful transformation of scripts

Checkpoints in this section:

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

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

- Avoid creating content on the fly on the client. If a user's browser does not handle scripts, no content will be generated or displayed. However, this is different than displaying or hiding already existing content by using a combination of style sheets and scripting; if there is no script, then the content is always shown. This also does not rule out generating pages on the fly on the server-side and delivering them to the client.
- Avoid creating links that use "javascript" as the URI. If a user is not using scripts, then they won't be able to link since the browser can't create the link content.

  Deprecated example. This is a dead-end link for a user agent where scripts are not supported or not loaded.

```
<A href="javascript:">...</A>
```

End example.

12.2 Scripts that cause flickering

Checkpoints in this section:

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

@@how to create and pause flicker with scripts

12.3 Scripts that cause movement and blinking

Checkpoints in this section:

- 7.2 Until user agents allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off). [Priority 2]
- 7.3 Until user agents allow users to freeze moving content, avoid movement in pages. [Priority 2]

@@how to create accessible scripts that cause movement and blinking. Part of the next section? directly accessible scripts?

12.4 Directly accessible scripts

Checkpoints in this section:

- 6.4 For scripts and applets, ensure that event handlers are input device-independent. [Priority 2]
- 6.3 Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page. [Priority 1]
- 8.1 Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important and not presented elsewhere, otherwise Priority 2.]
- 9.3 For scripts, specify logical event handlers rather than device-dependent event handlers. [Priority 2]

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 4.01, event handlers are attached to elements via event handler attributes [p. 54] (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 4.01, application-level event attributes are "onfocus", "onblur" (the opposite of "onfocus"), and "onselect". Note that these attributes are designed to be device-independent, but are implemented as keyboard specific events in current browsers.
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 4.01.
3. Do not write event handlers that rely on mouse coordinates since this prevents device-independent input.

12.5 Alternative presentation of scripts

Checkpoints in this section:

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

One way to accomplish this is with the NOSCRIPT [p. 50] element. The content of this element is rendered when 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.

12.6 Page updates and new windows

Checkpoints in this section:

- 7.4 Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2]
- 7.5 Until user agents provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects. [Priority 2]
- 10.1 Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user. [Priority 2]

@@this section needs work.

Index of HTML elements and attributes

Checkpoints in this section:

- 11.2 Avoid deprecated features of W3C technologies. [Priority 2]

Elements

Linear version of HTML 4.01 element index.

This index lists all elements in HTML 4.01. The first column of this table links to the definition of the element in the HTML 4.01 specification (*[HTML4] [p. 56]*). Elements that are deprecated in HTML 4.01 are followed by an asterisk (*). Elements that are obsolete in HTML 4.01 or don't exist in a W3C specification of HTML (2.0, 3.2, 4.01) do not appear in this table.

The second column indicates other W3C specifications for HTML that included each element. The third column indicates the element's role.

The last column lists the sections in the current document where the element is discussed. An entry of "N/A" means that the element is not discussed in this document.

Element name	Defined also in	Role	Techniques
A	2.0, 3.2	Structure	N/A
ABBR		Structure	N/A
ACRONYM		Structure	N/A
ADDRESS	2.0, 3.2	Metadata	N/A
APPLET*	3.2	Replaced	N/A
AREA	3.2	Structure	N/A
B	2.0, 3.2	Presentation	N/A
BASE	2.0, 3.2	Processing	N/A
BASEFONT*	3.2	Presentation	N/A
BDO		Processing	N/A
BIG	3.2	Presentation	N/A
BLOCKQUOTE	2.0, 3.2	Structure	N/A
BODY	2.0, 3.2	Structure	N/A
BR	2.0, 3.2	Presentation	N/A
BUTTON		Structure	N/A
CAPTION	3.2	Structure	N/A
CENTER*	3.2	Presentation	N/A
CITE	2.0, 3.2	Structure	N/A
CODE	2.0, 3.2	Structure	N/A
COL		Structure	N/A
COLGROUP		Structure	N/A
DD	2.0, 3.2	Structure	N/A
DEL		Metadata	N/A
DFN	3.2	Structure	N/A
DIR*	2.0, 3.2	Structure	N/A
DIV	3.2	Structure	N/A
DL	2.0, 3.2	Structure	N/A

DT	2.0, 3.2	Structure	N/A
EM	2.0, 3.2	Structure	N/A
FIELDSET		Structure	N/A
FONT*	3.2	Presentation	N/A
FORM	2.0, 3.2	Structure	N/A
FRAME		Replaced	N/A
FRAMESET		Presentation	N/A
H1	2.0, 3.2	Structure	N/A
HEAD	2.0, 3.2	Structure	N/A
HR	2.0, 3.2	Presentation	N/A
HTML	2.0, 3.2	Structure	N/A
I	2.0, 3.2	Presentation	N/A
IFRAME		Replaced	N/A
IMG	2.0, 3.2	Replaced	N/A
INPUT	2.0, 3.2	Structure	N/A
INS		Metadata	N/A
ISINDEX*	2.0, 3.2	Structure	N/A
KBD	2.0, 3.2	Structure	N/A
LABEL		Structure	N/A
LEGEND		Structure	N/A
LI	2.0, 3.2	Structure	N/A
LINK	2.0, 3.2	Metadata	N/A
MAP	3.2	Structure	N/A
MENU*	2.0, 3.2	Structure	N/A
META	2.0, 3.2	Metadata	N/A
NOFRAMES		Alternative	N/A
NOSCRIPT		Alternative	N/A
OBJECT		Replaced	N/A

OL	2.0, 3.2	Structure	N/A
OPTGROUP		Structure	N/A
OPTION	2.0, 3.2	Structure	N/A
P	2.0, 3.2	Structure	N/A
PARAM	3.2	Processing	N/A
PRE	2.0, 3.2	Presentation	N/A
Q		Structure	N/A
S*		Presentation	N/A
SAMP	2.0, 3.2	Structure	N/A
SCRIPT	3.2 (DTD)	Processing	N/A
SELECT	2.0, 3.2	Structure	N/A
SMALL	3.2	Presentation	N/A
SPAN		Structure	N/A
STRIKE*	3.2	Presentation	N/A
STRONG	2.0, 3.2	Structure	N/A
STYLE	3.2 (DTD)	Processing	N/A
SUB	3.2	Presentation	N/A
SUP	3.2	Presentation	N/A
TABLE	3.2	Structure	N/A
TBODY		Structure	N/A
TD	3.2	Structure	N/A
TEXTAREA	2.0, 3.2	Structure	N/A
TFOOT		Structure	N/A
TH	3.2	Structure	N/A
THEAD		Structure	N/A
TITLE	2.0, 3.2	Metadata	N/A
TR	3.2	Structure	N/A
TT	2.0, 3.2	Presentation	N/A

U*	3.2	Presentation	N/A
UL	2.0, 3.2	Structure	N/A
VAR	2.0, 3.2	Structure	N/A

Attributes

Linear version of HTML 4.01 attribute index.

This index lists some attributes in HTML 4.01 that affect accessibility and what elements they apply to. The first column of this table links to the definition of the attribute in the HTML 4.01 specification (*[HTML4] [p. 56]*). Attributes and elements that are deprecated in HTML 4.01 (*[HTML4] [p. 56]*) are followed by an asterisk (*). Attributes and elements that are obsolete in HTML 4.01 or don't exist in a W3C specification of HTML (2.0, 3.2, 4.01) do not appear in this table. Attributes that apply to most elements of HTML 4.01 are indicated as such; please consult the HTML 4.01 specification for the exact list of elements with this attribute.

The second column indicates other W3C specifications for HTML that included each attribute. The third column indicates the elements that take each attribute. The fourth column indicates the attribute's role.

The last column lists the sections in the current document where the attribute is discussed. An entry of "N/A" means that the attribute is not discussed in this document.

Attribute name	Applies to elements	Role	Techniques
abbr	TD, TH	Alternative	N/A
accesskey	A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA	User Interface	N/A
alt	APPLET, AREA, IMG, INPUT	Alternative	N/A
axis	TD, TH	Structure	N/A
class	Most elements	Structure	N/A
dir	Most elements	Processing	N/A
for	LABEL	Structure	N/A
headers	TD, TH	Structure	N/A
hreflang	A, LINK	Metadata	N/A
id	Most elements	Structure	N/A
label	OPTION	Alternative	N/A
lang	Most elements	Metadata	N/A
longdesc	IMG, FRAME, IFRAME	Alternative	N/A
scope	TD, TH	Structure	N/A
style	Most elements	Processing	N/A
summary	TABLE	Alternative	N/A
tabindex	A, AREA, BUTTON, INPUT, OBJECT, SELECT, TEXTAREA	User Interface	N/A
title	Most elements	Metadata	N/A
usemap	IMG, INPUT, OBJECT	Processing	N/A

The following is the list of HTML 4.01 attributes not directly related to accessibility. Content developers should use style sheets instead of presentation attributes. For even handler attributes, please refer to the section on device-independent event handlers [p. 46] for more detail.

Other structural attributes:
start*, value*, rowspan, colspan, span

Other presentation attributes:
align*, valign*, clear*, nowrap*, char, charoff, hspace*, vspace*, cellpadding, cellspacing, compact*, face*, size*, background*, bgcolor*, color*, text*, link*,

alink*, vlink*, border, noshade*, rules, size (deprecated according to element), marginheight, marginwidth, frame, frameborder, rows, cols

Other processing instruction attributes:
ismap, coords, shape

Other user interface attributes:
target, scrolling, noresize

Other metadata attributes:
type, cite, datetime

Event handler attributes:
onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onload, onload, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onreset, onselect, onsubmit, onunload

---

Acknowledgments

Web Content Guidelines Working Group Co-Chairs:

Jason White, University of Melbourne

Gregg Vanderheiden, Trace Research and Development

W3C Team contact:

Wendy Chisholm

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 Hansen, Phill Jenkins, Leonard Kasday, George Kerscher, Marja-Riitta Koivunen, Josh Krieger, Chuck Letourneau, 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, and Jaap van Lelieveld

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

13 References

For the latest version of any W3C specification please consult the list of W3C Technical Reports at http://www.w3.org/TR.

[CSS1]
"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. This CSS1 Recommendation is http://www.w3.org/TR/1999/REC-CSS1-19990111. The latest version of CSS1 is available at http://www.w3.org/TR/REC-CSS1.

[CSS2]
"CSS, level 2 Recommendation", B. Bos, H. Wium Lie, C. Lilley, and I. Jacobs, eds., 12 May 1998. This CSS2 Recommendation is http://www.w3.org/TR/1998/REC-CSS2-19980512. The latest version of CSS2 is available at http://www.w3.org/TR/REC-CSS2.

[HTML4]
"HTML 4.01 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds., 24 December 1999. This HTML 4.01 Recommendation is http://www.w3.org/TR/1999/REC-html401-19991224.

[MATHML]
"Mathematical Markup Language", P. Ion and R. Miner, eds., 7 April 1998, revised 7 July 1999. This MathML 1.0 Recommendation is http://www.w3.org/TR/1998/REC-MathML-19990707. The latest version of MathML 1.0 is available at http://www.w3.org/TR/REC-MathML.

[WCAG10]
"Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, and I. Jacobs, eds., 5 May 1999. This WCAG 1.0 Recommendation is http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505.

[WCAG10-TECHS]
"Techniques for Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, I. Jacobs, eds. This document explains how to implement the checkpoints defined in "Web Content Accessibility Guidelines 1.0". The latest draft of the techniques is available at http://www.w3.org/WAI/GL/WCAG10-TECHS/.

14 Resources

Note: *W3C does not guarantee the stability of any of the following references outside of its control. These references are included for convenience. References to products are not endorsements of those products.*

14.1 Operating system and programming guidelines

[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.

[MACROMEDIA]
: Flash OBJECT and EMBED Tag Syntax from Macromedia.

14.2 User agents and other tools

A list of alternative Web browsers (assistive technologies and other user agents designed for accessibility) is maintained at the WAI Web site.

[ASTER]
: For information about ASTER, an "Audio System For Technical Readings", consult T. V. Raman's home page.

[HYPERMEDIA]
: IBM's techexplorer Hypermedia Browser.

14.3 Accessibility resources

[TRACE]
: The Trace Research & Development Center. Consult this site for a variety of information about accessibility, including a scrolling Java applet that may be frozen by the user.

[WAI-ER]
: The WAI Evaluation and Repair Working Group