MATC Web Accessibility

The W3C (World Wide Web Consortium) Recommended Standards

A Brief Overview of Priority Level 1 Standards and How to Comply

February 1, 2006

Checklist of Checkpoints for Web Content Accessibility Guidelines 1.0 — Priority 1

It is recommended that you read the working draft from W3C web site title “Web Content Accessibility Guideline.” This is a working draft document dated November 23, 2005 found at http://www.w3.org/TR/WCAG20/.

The information in this document is taken from the W3C recommendations at www.w3.org first published in 1999.

It is an appendix to the W3C Web Content Accessibility Guidelines 1.0. It provides a list of all checkpoints from the Web Content Accessibility Guidelines 1.0, organized by concept, as a checklist for Web content developers. This list is a good reference to check accessibility on your web pages.

Included is a list version as well as a checkpoint grid for web authors.

Priority 1 means that these items **must** by satisfied. A web developer must satisfy this checkpoint. Otherwise, one or more groups will find it impossible to access information in the document. Satisfying this checkpoint is a basic requirement for some groups to be able to use Web document.

Priority 2 means a Web content developer **should** satisfy this checkpoint. Otherwise, one or more groups will find it difficult to access information in the document. Satisfying this checkpoint will remove significant barriers to accessing Web documents.

Priority 3 means a Web content developer **may** address this checkpoint. Otherwise, one or more groups will find it somewhat difficult to access information in the document. Satisfying this checkpoint will improve access to Web documents.

Priority 2 and 3 are not covered in this document.

More information regarding Priority 2 and 3 can be found on the www.w3.org web site.

Please use the Priority 1 Checklist for Web Authors along with the document for help in creating accessible web pages.

This document contains the following Priority 1 discussion points:

- General Information
- Images and Image Maps
- Tables
- Frames
- Applet and Scripts
- Multimedia
- If all else fails!

Priority 1 checkpoints — General

Checkpoint 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.

For example, in HTML:

- Use "alt" for the IMG, INPUT, and APPLET elements, or provide a text equivalent in the content of the OBJECT and APPLET elements.
- For complex content (e.g., a chart) where the "alt" text does not provide a complete text equivalent, provide an additional description using, for example, "longdesc" with IMG or FRAME, a link inside an OBJECT element, or a description link.

- For image maps, either use the "alt" attribute with AREA, or use the MAP element with A elements (and other text) as content.

Checkpoint 2.1

Ensure that all information conveyed with color is also available without color, for example from context or markup.

Core Techniques (Structure vs. Presentation)

When designing a document or series of documents, content developers should strive first to identify the desired structure for their documents before thinking about how the documents will be presented to the user. Distinguishing the structure of a document from how the content is presented offers a number of advantages, including improved accessibility, manageability, and portability. Learning how to distinguish between structure and presentation can be difficult at times. Content developers may consider that a horizontal line communicates structural division. This may be true for sighted users but to unsighted users without graphical browsers, a horizontal line may have next to no meaning. Content developers should use the HTML 4.01 heading elements (H1-H6) to identify new sections. These may be complemented by visual or other cues such as horizontal rules, but should not be replaced by them.

Content developers should not use structural elements to achieve presentation effects. For instance in HTML, even though the BLOCKQUOTE element may cause indented text in some browsers, it is designed to identify a quotation, not create a presentation side-effect. BLOCKQUOTE elements used for indentation confuse users and search robots alike, who expect the element to be used to mark up block quotations.

CSS Techniques (Ensuring information is not in color alone)

Ensure that information is not conveyed through color alone. For example, when asking for input from users, do not write “Please select an item from those listed in green.” Instead, ensure that information is available through other style sheets (e.g., a font affect) and through context (e.g., comprehensive text links).

For instances, in a document, examples could be styled by default (through style sheets) as follows:

- They are surrounded by a border.
- They use a different background color.
- They begin with the work “Example” or Deprecated Example”.
- They also end with the phrase “End Example”, but that phrase is hidden by default with ‘display: none’. For user agents that don’t support style sheets or when style sheets are turned off, this text helps delineate the end of an example for readers who may not be able to see the border around the example.

Checkpoint 4.1

Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions).

Text equivalents - Since text content can be presented to the user as synthesized speech, Braille, and visually-displayed text, these guidelines require *text equivalents* for graphic and audio information. Text equivalents must be written so that they convey all essential content.

HTML Techniques (Identifying changes in language)

If you use a number of different languages on a page, make sure that any changes in language are clearly identified by using the “lang” attribute. For example, <span lang=”fr”>je ne sais quoi</span>.

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 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.

Checkpoint 6.1

Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document.

When content is organized logically, it will be rendered in a meaningful order when style sheets are turned off or not supported.

CSS Techniques (Generated Content)

- Provide a text equivalent for any important image or text generated by style sheets (e.g., via the ‘background-image’, ‘list-style’, or ‘content’ properties)
- Ensure that important content appears in the document object. Text generated by style sheets is not part of the document source and will not be available to assistive technologies that access content through the Document Object Model Level 1 (DOM1).

CSS2 includes several mechanisms that allow content to be generated from style sheets:

- The :before and :after pseudo-elements and the ‘content’ property. When used together, these allow authors to insert markers (e.g., counters and constant strings such as “End Example” in the example below) before or after the element’s content.
- The ‘cue’, ‘cue-before’, and ‘cue-after’ properties. This properties allow users to play a sound before or after an element’s content.
- List styles, which may be numbers, glyphs, or images (usually associated with the LI element in HTML). CSS2 adds international list styles to the styles defined in CSS1.

Generate content can serve as markers to help users navigate a document and stay oriented when they can’t access visual clues such as proportional scrollbars, frames with tables of content, etc.

For instance, the following user style sheet would cause the words “End Example” to be generated after each example marked up with a special class value in the document.

CSS Techniques (Rules and Borders)

Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document.

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

Use these CSS properties to specify border styles:

- ‘border’, ‘border-width’, border-style’, ‘border-color’
- ‘border-spacing’ and ‘border-collapse’ for tables
- ‘outline’, ‘outline-color’, ‘outline-style’, and ‘outline-width’ for dynamic outlines.

Authors should use style sheets to create rules and borders.

Here the H1 element will have a top border that is 2px thick, red and separated from the content by 1em.:

```
<style type=”tex/css”>h1 {padding-top: 1em; border-top; 2px red}</style>
```

If a rule (e.g., the HR element) is used to indicate structure, be sure to make sure to indicate the structure in a non-visual way as well. (e.g., by using structural markup).

Here the DIV element is used to create a navigation bar, which includes a horizontal separator:

```
<DIV class=”navigation-bar”><hr><a rel=”Next” href=next.html”>[Next Page]</a></DIV>
```

CSS Techniques (Using Style Sheet Positioning and Markup to Transform Gracefully)

Using the positioning properties of CSS2, content may be displayed at any position on the user’s viewport. The order in which items appear on a screen may be different than the order they are found in the source document. The following example demonstrates a few principles:

1) The text appears visually in the browser in a different order than in the markup.
2) CSS positioning may be used to create tabular effects. A TABLE element could have been used to create the same effect.

Note that a class is defined for each object that is being positioned. The use of “id” could be substituted for “class” in these examples. “Class” was used because in the live example, the objects are replicated and thus not unique.

Deprecated example

```
<head>
<style type=”text/css”>
.menu1 {position:….. }
.item1 {position:…}
</style>
</head>
<body>
<div class=”box”>
<span class=”menu1”>Products</span>
<span class=”item1”>product1</span>
</div>
</body>
```

When style sheets are applied, the text appears in two columns. Elements of class “menu1” appear as column headings. “product1” is listed under Products. When style sheets are not applied, all the text appears in one line of words. They appear in the order in which they are written in the source.

The following example shows that the same visual appearance may be created in a browser that support style sheets as well as creating a more meaningful presentation when style sheets are not applied. Structural markup (definition lists) have been applied to the content. Note that the margins have been set to 0 since in HTML browsers, definition lists are displayed with a margin set on the DD element.

Example

```
<head><style type=”text/css”>
.menu1 { position: …..}
.item1 {position:…}
#box {position:…}
</style></head><body>
<div class=”box”>
<dl>
<dt class=”menu1”>Products</dt>
<dd class=”item1”>product1</dd>
</dl>
```

```
</div>
</body>
```

When the style sheets are applied, it looks like before. However, when the style sheet is not applied, the text appears in a definition list rather than a string of words. What appears as column headings now appear as a defined terms when the style sheet is not applied.

Checkpoint 6.2

Ensure that equivalents for dynamic content are updated when the dynamic content changes.

HTML Techniques (Text and non-text equivalents for applets and programmatic objects)

OJECT
If OBJECT is used, provide a text-equivalent in the content of the element. See Checkpoint 1.1.

Example:

```
<OBJECT classid=”java:Press.class” width=”500” height=”500”>As temperature increases, the molecules in the
balloon…</OBJECT>
```

A more complex example takes advantage of the fact the OBJECT element 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 the temperature increases….
</OBJECT>
</OBJECT>
</OBJECT>
```

APPLET
If APPLET is used, provide a text equivalent with the “alt” attribute and in the content in the APPLET element. This enables the content to transform gracefully for those users 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>
```

HTML Techniques (Frame Sources)

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 files. Images may be inserted into the HTML file and their text alternatives will evolve correctly.

HTML Techniques (Alternative Presentation of Scripts)

See checkpoint 1.1. One way to accomplish this is with the NOSCRIPT 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>
```

Checkpoint 7.1

Until user agents allow users to control flickering, avoid causing the screen to flicker.

User agent is software used to access web content, including desktop graphical browsers, text browsers, voice browsers, mobile phones, multimedia players, plug-ins, and some software assistive technologies used in conjunction with browsers such as screen readers, screen magnifiers, and voice recognition software. Not all user agents or assistive technologies provide the accessibility control users require. Checkpoints that contain the phrase “until user agents..” require content developers to provide additional support for accessibility until most user agents readily available to their audience include the necessary accessibility features.

Note: The W3C WAI Web site provides information about user agent support for accessibility features.

People with photosensitive epilepsy can have seizures triggered by flickering or flashing in the 4 to 59 flashes per second (Hertz) range with a peak sensitivity at 20 flashes per second as well as a quick change from dark to light (like strobe lights).

Core Techniques (Screen flicker)

A flickering or flashing screen may cause seizures in users with photosensitive epilepsy and content developers should thus avoid causing the screen to flicker.

Core Techniques (Visual Information and Motion)

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

Note: If there is no important visual information, for example, an animated talking head that describes (through pre-recorded speech) how to use this site, then an auditory description is not necessary. For movies, provide auditory descriptions that are synchronized with the original audio.

HTML Techniques (Directly Accessible Applets)

If an applet (created with either OBJECT or APPLET) requires user interaction (e.g., the ability to manipulate a physics experiment) than 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 (learn about TRACE).

Checkpoint 14.1

Use the clearest and simplest language appropriate for a site's content. Ensure that documents are clear and simple.

Core Techniques (Comprehension)

Writing Style — The following writing style suggestions should help make the content of your site easier to read for everyone, especially people with reading and/or cognitive disabilities.

1) Strive for clear and accurate headings and link descriptions. This includes using link phrases that are terse and that make sense when read out of context or as part of a series of links (Some users browse by jumping from link to link and listening only to link text). Use informative headings so that users can scan a page quickly for information rather than reading it in detail.
2) State the topic of the sentence or paragraph at the beginning of the sentence or paragraph (this is called 'front-loading'). This will help both people who are skimming visually, but also people who use speech synthesizers. "Skimming" with speech currently means that the user jumps from heading to heading, or paragraph to paragraph and listens to just enough words to determine whether the current chunk of information (heading, paragraph, link, etc.) interests them. If the main idea of the paragraph is in the middle or at the end, speech users may have to listen to most of the document before finding out what they want. Depending on what the user is looking for and how much they know about the topic, search features may also help users locate content more quickly.
3) Limit each paragraph to one main idea.
4) Avoid slang, jargon, and specialized meanings of familiar words, unless defined within your document.
5) Favor words that are commonly used. For example, use "begin" rather than "commence" or use "try" rather than "endeavor."
6) Use active rather than the passive verbs.
7) Avoid complex structures.

Multi-Media Equivalents — For people who do not read well or not at all, multimedia (non-text) equivalents may help facilitate comprehension. Beware that multimedia presentations do not always make text easier to understand. Sometimes, multimedia presentations may make it more confusing.

Examples of multimedia that supplement text:
1) A chart of complex data, such as a sales figures of a business for the past fiscal year.
2) A translation of the text into a sign language movie clip. Sign language is a very different language than spoken languages. For example, some people who may communicate via American Sign Language may not be able to read American English.
3) Pre-recorded audio of music, spoken language, or sound effects may also help non-readers who can perceive audio presentations. Although text may be generated as speech through speech synthesis, changes in a recorded speaker's voice can convey information that is lost through synthesis.

Priority 1 Checkpoints — Images and image maps

Checkpoint 1.2

Provide redundant text links for each active region of a server-side image map.

Core Techniques (Text equivalents)

Text is considered accessible to almost all users since it may be handled by screen readers, non-visual browsers, and Braille readers. It may be displayed visually, magnified, synchronized with a video to create a caption, etc. As you design a document containing non-textual information (images, applets, sounds, multimedia presentations, etc.), supplement that information with textual equivalents wherever possible.

When a text equivalent is presented to the user, it fulfills essentially the same function (to the extent possible) as the original content. For complex content (charts, graphs, etc.), the text equivalent may be longer and include descriptive information.

Text equivalents must be provided for

- logos
- photos
- submit buttons
- applets
- image bullets in lists
- ASCII art
- All links within an image map
- Invisible images used for page layout

HTML Techniques (Server-Side Image Maps)

When a server-side image map must be used, content developers should provide an alternative list of image map choices. There are 3 techniques:

1) Include the alternative links within the body of an OBJECT element
2) If IMG 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 the "alt" attribute).
   Example:<img src="welcome.gif" alt="Welcome! (text links follow)" ismap>
3) 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.

Checkpoint 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.

HTML Techniques (Client-side versus server-side image maps)

See Checkpoint 1.2 HTML Techniques.

Priority 1 Checkpoints — Tables

Checkpoint 5.1

For data tables, identify row and column headers.

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

- Provide summary information
- Identifying row and column information

HTML Techniques (Identifying rows and column information)

For data tables, identify row and column headers. For example, in HTML, use TD to identify data cells and TH to identify headers.

Checkpoint 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.

- Identify structural groups of rows (use THEAD for repeated table headers, TFOOT for repeated table footers, and TBODY for other groups of rows) and groups of columns (COLGROUP and COL).
- Label table elements with the 'scope', 'headers', and 'axis' attributes so that future browsers and assistive technologies will be able to select data from a table by filtering on categories.
- Do not use PRE to create a tabular layout of text, use the TABLE element so that assistive technologies may recognize that it is a table.

Example of using the 'scope' attribute:

```
<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>
```

Example using the 'axis' attribute

```
  <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>
```

Priority 1 Checkpoints — Frames

Checkpoint 12.1

Title each frame to facilitate frame identification and navigation.

For example, in HTML, use ‘title’ attribute on FRAME elements.

Example for using ‘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 col=”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 Electronic Library </a>
</NOFRAMES>
</FRAMESET>
```

Priority 1 Checkpoints — Applets and Scripts

Checkpoint 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.

For example, ensure that links that trigger scripts work when scripts are turned off or not supported (e.g., do not use "javascript" as the link target). If it is not possible to make the page usable without scripts, provide a text equivalent with the NOSCRIPT element, or use a server-side script instead of a client-side script, or provide an alternative accessible page.

HTML Techniques (Text and non-text equivalents for applets and programmatic objects

See Checkpoint 6.2 above.

HTML Techniques (Directly accessible scripts)

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 (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) User application-level event triggers rather than user interaction triggers. In HTML 4.01, application-level event attributes are "onfocus", "onblur" (opposite of "onfocus), and "onselect". Not 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 not 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.

Priority 1 Checkpoints — Multimedia

Checkpoint 1.3

Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation.

Synchronize the auditory description with the audio track as per Checkpoint 1.4 (see below). An auditory description is an non-text equivalent of the key visual element of the presentation.

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

Note: if there is not important visual information, for example, an animated talking head that describes (through prerecorded speech) how to use the site, then an auditory description is not necessary.

For movies, provide auditory descriptions that are synchronized with the original audio.

Checkpoint 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.

Core Techniques (Audio Information)

Auditory presentations must be accompanied by text transcripts, textual equivalents of auditory events. When these transcripts are presented synchronously with a video presentation they are called captions and are used by people who cannot hear the audio track of the video material.

Some media formats (e.g., QuickTime and SMIL) allow captions and video descriptions to be added to the multimedia clip. SAMI allows captions to be added. The following example demonstrates that captions should be include speech as well as other sounds in the environment that help viewers understand what is going on.

Example

Caption for a scene. The phone rings 3 times, then it is answered.

[phone rings]

[ring]

[ring]

Hello?

Until the format you are using supports alternative tracks, two versions of the movie could be make available, one with captions and descriptive video, and one without. Some technologies, such as SMIL and SAMI, allow separate audio/visual files to be combined with text files via a synchronization file to create captioned audio and movies.

Some technologies also allow the user to choose multiple sets of captions to match their reading skills.

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

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

HTML Techniques (Directly Accessible Applet)

Refer to Checkpoint 7.1 above.

Priority 1 Checkpoints — If all else fails

Checkpoint 11.4

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

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

1) Allow users to navigate to a separate page that is accessible, contains the same information as the inaccessible page, and is maintained with the same frequency as the inaccessible page.
2) Instead of static alternative pages, set up server-side scripts that generates accessible versions of a page on demand.
3) Provide a phone number, fax number, e-mail or postal address where information is available and accessible, preferably 24 hours a day.

Here are two techniques for linking to an accessible alternative page:

1. Provide links at the top of both the main and alternative pages to allow a user to move back and forth between them. For example, at the top of a graphical page include a link to the text-only page, and at the top of a text-only page include a link to the associated graphical page. Ensure that these links are one of the first that users will tab to by placing them at the top of the page, before other links.
2. Use meta information to designate alternative documents. Browsers should load the alternative page automatically based on the user's browser type and preferences.

Note: Content developers should only resort to alternative pages when other solutions fail because alternative pages are generally updated less often than “primary” pages. An out-of-date page may be frustrating as one that is inaccessible since, in both cases, the information presented on the original page is unavailable. Automatically generating alternative pages may lead to more frequent updates, but content developers must still be careful to ensure that generated pages always make sense, and that users are able to navigate a site by following links on primary pages, alternative pages, or both. Before resorting to an alternative page, reconsider the design of the original page; making it accessible is likely to improve it for all users.