Guideline 1. Provide equivalent alternatives to auditory and visual content.

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.

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

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. Refer to checkpoint 1.1 for information about textual equivalents for visual information.

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.

Guideline 2. Don't rely on color alone.

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

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.

Guideline 3. Use markup and style sheets and do so properly.

3.1 When an appropriate markup language exists, use markup rather than images to convey information.

For example, use MathML to mark up mathematical equations, and *style sheets* to format text and control layout. Also, avoid using images to represent text -- use text and style sheets instead.

3.2 Create documents that validate to published formal grammars.
For example, include a document type declaration at the beginning of a document that refers to a published DTD (e.g., the strict HTML 4.0 DTD).
3.3 Use style sheets to control layout and presentation.
For example, use the CSS 'font' property instead of the HTML FONT element to control font styles.
3.4 Use relative rather than absolute units in markup language attribute values and style sheet property values.
For example, in CSS, use 'em' or percentage lengths rather than 'pt' or 'cm', which are absolute units. If absolute units are used, validate that the rendered content is usable (refer to the section on validation).
3.5 Use header elements to convey document structure and use them according to specification.
For example, in HTML, use H2 to indicate a subsection of H1. Do not use headers for font effects.
3.6 Mark up lists and list items properly.
For example, in HTML, nest OL, UL, and DL lists properly.
3.7 Mark up quotations. Do not use quotation markup for formatting effects such as indentation.
For example, in HTML, use the Q and BLOCKQUOTE elements to markup short and longer quotations, respectively.

Guideline 4. Clarify natural language usage

4.1 Clearly identify changes in the natural language of a document's text and any *text equivalents* (e.g., captions).
For example, in HTML use the "lang" attribute. In XML, use "xml:lang".

Guideline 5. Create tables that transform gracefully.

5.1 For data tables, identify row and column headers.
For example, in HTML, use TD to identify data cells and TH to identify headers.
5.2 For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells.
For example, in HTML, use THEAD, TFOOT, and TBODY to group rows, COL and COLGROUP to group columns, and the "axis", "scope", and "headers" attributes, to describe more complex relationships among data.
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*).
Note. *Once user agents* support style sheet positioning, tables should not be used for layout.
5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting.
For example, in HTML do not use the TH element to cause the content of a (non-table header) cell to be displayed centered and in bold.

Guideline 6. Ensure that pages featuring new technologies transform gracefully.

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.

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

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 as per checkpoint 11.4.

6.4 For scripts and applets, ensure that event handlers are input device-independent.

Refer to the definition of device independence.

6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page.

For example, in HTML, use NOFRAMES at the end of each frameset. For some applications, server-side scripts may be more accessible than client-side scripts.

Guideline 7. Ensure user control of time-sensitive content changes.

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

Note. 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 quick changes from dark to light (like strobe lights).

7.2 *Until user agents* allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off).

7.3 *Until user agents* allow users to freeze moving content, avoid movement in pages.

When a page includes moving content, provide a mechanism within a script or applet to allow users to freeze motion or updates. Using style sheets with scripting to create movement allows users to turn off or override the effect more easily.

7.4 *Until user agents* provide the ability to stop the refresh, do not create periodically auto-refreshing pages.

For example, in HTML, don't cause pages to auto-refresh with "HTTP-EQUIV=refresh" until user agents allow users to turn off the feature.

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.

Guideline 8. Ensure direct accessibility of embedded user interfaces.

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

Guideline 9. Design for device-independence.

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.
9.2 Ensure that any element that has its own interface can be operated in a device-independent manner.
9.3 For scripts, specify logical event handlers rather than device-dependent event handlers.

Guideline 10. Use interim solutions.

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.

For example, in HTML, avoid using a frame whose target is a new window.

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.

The label must immediately precede its control on the same line (allowing more than one control/label per line) or be in the line preceding the control (with only one label and one control per line).

Guideline 11. Use W3C technologies and guidelines.

11.1 Use W3C technologies when they are available and appropriate for a task and use the latest versions when supported.
11.2 Avoid deprecated features of W3C technologies.

For example, in HTML, don't use the *deprecated* FONT element; use style sheets instead (e.g., the 'font' property in CSS).

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.

Guideline 12. Provide context and orientation information.

12.1 Title each frame to facilitate frame identification and navigation.

For example, in HTML use the "title" attribute on FRAME elements.

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

For example, in HTML, use "longdesc," or a *description link.*

12.3 Divide large blocks of information into more manageable groups where natural and appropriate.

For example, in HTML, use OPTGROUP to group OPTION elements inside a SELECT; group form controls with FIELDSET and LEGEND; use nested lists where appropriate; use headings to structure documents, etc.

12.4 Associate labels explicitly with their controls.
For example, in HTML use LABEL and its "for" attribute.

Guideline 13. Provide clear navigation mechanisms.

13.1 Clearly identify the target of each link.
Link text should be meaningful enough to make sense when read out of context -- either on its own or as part of a sequence of links. Link text should also be terse.
For example, in HTML, write "Information about version 4.3" instead of "click here". In addition to clear link text, content developers may further clarify the target of a link with an informative link title (e.g., in HTML, the "title" attribute).

13.2 Provide metadata to add semantic information to pages and sites.
For example, use RDF ([RDF]) to indicate the document's author, the type of content, etc.
Note. Some HTML *user agents* can build navigation tools from document relations described by the HTML LINK element and "rel" or "rev" attributes (e.g., rel="next", rel="previous", rel="index", etc.).

13.3 Provide information about the general layout of a site (e.g., a site map or table of contents).
In describing site layout, highlight and explain available accessibility features.

13.4 Use navigation mechanisms in a consistent manner.

Guideline 14. Ensure that documents are clear and simple.

14.1 Use the clearest and simplest language appropriate for a site's content.