Illinois Information Technology Accessibility Act Implementation Guidelines for Web-Based Information and Applications 1.0

(formerly Illinois Web Accessibility Standards)

1. Coding

1.1 - Use valid, standard web programming code.

What:

The World Wide Web Consortium (W3C) sets and publishes standards for web programming languages including HyperText Markup Language (HTML/XHTML), and Cascading Style Sheets (CSS). Programming code is considered "valid" when it follows the rules and conventions specified in the published standards.

Why:

Valid code is the foundation for accessibility. Screen readers and other assistive technologies most reliably interpret and interact with web pages that are built using valid, standard code.

How:

For web pages, indicate the programming language you are using by starting your code with a standard document type declaration, such as:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

See the full list of W3C Recommended Document Type Declarations (http://www.w3.org/QA/2002/04/valid-dtd-list.html).
Use the W3C HTML Validation Service (http://validator.w3.org) to check your code.

Ref:

WCAG 3.2, 11.1, 11.2

1.2 - Use appropriate markup to convey document structure.

What:

HTML includes "markup" (programming code) to identify the structural elements of a document. For example, <p> identifies a paragraph and <h1> identifies a heading.

Why:

Screen readers use structural information to help make reading more efficient. For example, most screen readers can skip from heading to heading, announce the number of items in a list, and identify the current row and column in a data table.

How:

Identify headings, paragraphs, lists, quotations, etc., using the appropriate markup instead of relying solely on formatting. For example, use <h1> tags to identify the top-level heading rather than simply making its text large and bold. Do not misuse structural markup for formatting effects, such as using <blockquote> to indent a paragraph. Use Cascading Style Sheets (CSS) for formatting.

Ref:

WCAG 3.5, 3.6, 3.7, 5.4

1.3 - Provide meaningful page titles.

What:

Page titles appear in the title bar at the very top of the web browser window. In HTML, the page title is specified using the `<title>` tag.

Why:

Screen readers announce the page title whenever a new page is loaded. Screen reader users read the page title to confirm that they have reached the intended page and to identify the content and/or function the page.

How:

Every web page should have a title. The title should indicate both the name of the site and the topic of the page and should be unique within the site whenever possible.

Note: Titles should usually be 60 or fewer characters in length.

Ref:

WCAG 13.2

1.4 - Use headings to introduce sections and sub-sections, and use them in the correct order.

What:

Headings are used to introduce sections and sub-sections within a page. HTML includes six levels of headings (`<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, and `<h6>`), representing the different levels in an "outline" of the page. Web browsers normally display headings in large, bold font.

Why:

Screen readers identify headings so that users can easily skim through content and quickly skip to sections of interest.

How:

Start the main content area of each page with an `<h1>` heading that indicates the topic of the page as identified in the page title. If the page is long enough to be divided into sections, group each section in a `<div>` and begin each `<div>` with a heading of the appropriate level:

- `<h2>` for main sections,
- `<h3>` for sub-sections,
- `<h4>` for sub-sub sections, and so on.

Use Cascading Style Sheets (CSS) to control the size and appearance of headings.

Note: Headings (and other elements) may be hidden visually but revealed to assistive technologies using a style such as:

```
.invisible { position: absolute; left: -10000px; width: 1px;
height: 1px; overflow: hidden; }
```

Ref:

WCAG 3.5, 12.3

1.5 - Use lists to identify series of related items, including navigation menus.

What:

HTML includes list elements that are used to group and identify series of related items. Lists may be "ordered" (`<ol>`) or "unordered" (`<ul>`) and may contain "nested" sub-lists. Navigation menus are lists of links to the various pages within a site.

Why:

Screen readers identify the number and type of items in a list and enable users to easily skip all or part of the list if desired.

How:

Use ordered lists (`<ol>`) when the sequence of items is important; use unordered lists (`<ul>`) when the order does not matter. Use Cascading Style Sheets (CSS) to control the appearance of numbers and bullets, but do not use bullets on ordered lists or number-styles on unordered lists.

Note: Navigation menus should be preceded by headings of the appropriate level (usually <h2>).

Ref:

WCAG 3.6

2. Text

2.1 - Use text to display text, unless formatting that cannot be achieved with CSS is required.

What:

Web developers often use images of text to achieve a specific style, size, or special effect.

Why:

Users with limited vision rely on the ability to enlarge text or choose enhanced color combinations. However, most web browsers cannot change the size and color of images.

How:

Whenever possible, use actual text instead of images of text. Cascading Style Sheets (CSS) can be used to achieve specific sizes, colors, or effects. Text that requires exact formatting, such as logos and/or other branding elements, are appropriate exceptions.

Ref:

WCAG 3.1

2.2 - Use relative sizes for fonts.

What:

Font sizes can be set using "absolute" or "relative" units of measurement. Absolute units, notably pixels (`px`), points (`pt`), and inches (`in`), are based on fixed physical measurements; "relative" units, such as percentages (`%`) or keywords (e.g., `small`, `medium`, or `large`) are based on the user's default font size.

Why:

Users with limited vision often rely on the ability to enlarge text. Most web browsers allow users to easily change the size of text that has been set with relative units (or not set at all). Using absolute font sizes generally makes it more difficult for users to change text size to meet their needs.

How:

Set font sizes using relative measurements or avoid setting font sizes altogether.

Ref:

WCAG 3.4

2.3 - Identify the language of text.

What:

HTML uses the `lang` attribute to specify language in a web page. It can be set for any HTML element.

Why:

Words written in foreign languages can be unintelligible when spoken by a screen reader. Most screen readers are able to pronounce words in their appropriate language if it is specified.

How:

Use the lang attribute on the `<html>` element to identify the primary language of each document, for example, `<html lang="en">`, for English. Use the lang attribute on `<span>` or other elements to identify words or phrases in other languages. For example, a Spanish phrase within an English document could be coded as: `<span lang="es">se habla español</span>`.

Ref:

WCAG 4.1, 4.3

2.4 - Use images instead of "ASCII art."

What:

"ASCII art" is images created using special arrangements of text characters and symbols. For example, ":-)" is often used to create a smiley face, and "->" is often used as an arrow.

Why:

Screen readers read most ASCII art literally, which can be extremely confusing. For example, ":-)" reads as "colon dash right parenthesis" and "->" as "dash greater than."

How:

Use images with appropriate alternate text instead of ASCII art.

Ref:

WCAG 1.1; 508 a

3. Colors

3.1 - Do not convey information with color alone.

What:

Color is often used to indicate special functions or status. For example, required form fields are frequently indicated with red labels.

Why:

Users with blindness, limited vision, or color-blindness may miss information presented with color.

How:

Whenever color is used as an indicator, use a non-color-based indicator as well. For example, required form fields could be identified with an icon (an image with appropriate alternate text) in the label, as well as with color.

Ref:

WCAG 2.1; 508 c

3.2 - Use contrasting foreground and background colors.

What:

Web authors can set specific colors to be used for foregrounds (text) and backgrounds. Sometimes images are used as backgrounds.

Why:

Users with limited vision or color-blindness may have difficulty reading text that is similar in color to its background.

How:

For text, use dark colors on light backgrounds, or vice versa. Avoid combinations of red and green as well as busy background images. Text must have a contrast ratio of at least 3:1.

Ref:

WCAG 2.2

4. Images

4.1 - Provide appropriate "alternate text" for all images.

What:

The HTML image element (`<img>`) includes an "alternate text" attribute (`alt`) that is used to provide text that can be substituted when the image itself cannot be displayed. Alternate text is meant to be a concise replacement for an image and should serve the same purpose and convey the same meaning.

Why:

Individuals who are blind cannot perceive information presented in images; screen reading software reads alternate text instead.

How:

ALL images must have appropriate alternate text. As a rule of thumb, consider what you might say if you were reading the web page to someone over the telephone. You do not need to include the words "image of" or "graphic of." Specifically:

- For images that contain words or letters - use alternate text that includes the same words or letters.
- For images links - use alternate text that identifies the link's destination or function. You do not need to include the words "link to."
- For images that are invisible, purely decorative, do not convey meaning, or are completely redundant with text that is already on screen - use `alt=""` (empty string) to indicate that the image can be safely ignored by a screen reader.

- For CAPTCHA images, use alt text to indicate that the image is being used for CAPTCHA and provide an alternative test in a non-visual medium (e.g., an audio file).

Ref:

WCAG 1.1; 508 a

4.2 - Provide full descriptions for graphs, diagrams, and other meaningful images.

What:

"Meaningful" images are images that convey more information than can appropriately be expressed as alternate text.

Why:

A full description allows a user who cannot see or understand a meaningful image to receive the same information as a sighted individual.

How:

Present a full description of a meaningful image either on the page on which the image appears or through a link immediately preceding or following the image. Use alternate text to provide a concise name for the image. For example, the alternate text of a graph should state its title and the full description should summarize its trends and/or present a table of its data.

Note: The `longdesc` attribute of the `<img>` element can also be used to provide a link to a full description. Because `longdesc` it is not yet supported by most web browsers, it should not be used as the only method of providing a full description.

Ref:

WCAG 1.1; 508 a

5. Image Maps

5.1 - Provide alternate text for each area in client-side image maps.

What:

Image maps are images divided into multiple "areas," with each area having its own hypertext link.

Why:

Just as images must have alternate text, each area of an image map must also have appropriate alternate text for use when the image is not displayed.

How:

Use alternate text that indicates the function or destination of the link for each area of a client-side image map. The image itself should have alternate text that indicates the overall function of the image map.

Ref:

WCAG 1.1; 508 a

5.2 - Use client-side image maps instead of server-side image maps unless areas cannot be defined with available shapes.

What:

While client-side image maps and server-side image maps look and operate similarly, they are technically very different. Because of the way server-side

image maps work, all information about the image and links is stored at the web server and is not available to the user's web browser or assistive technology.

Why:

Screen readers cannot identify or read the separate areas or links within server-side image maps.

How:

Whenever possible, use client-side image maps instead of server-side image maps. If server-side image maps must be used, provide an accessible alternative that includes the same content and functionality. In cases where it is impossible to create an equivalent accessible version, such as with some geographical imaging and mapping systems, exceptions may be necessary.

Ref:

WCAG 1.2, 9.1; 508 e, f

6. Sounds

6.1 - Do not convey information with sound alone.

What:

It is possible to use sound for a variety of purposes, including presenting warning signals, cues, or verbal instructions.

Why:

Users who are deaf or hard of hearing may miss information provided only through sound.

How:

Whenever significant information is provided by sound, include a visual indicator that provides the same information as well.

Ref:

WCAG 1.1; 508 a

6.2 - Do not automatically play audio.

What:

It is possible for a web page to automatically play sound or music when it loads.

Why:

Background sounds or music can make it difficult or impossible for screen readers user to hear their screen readers.

How:

Do not automatically play audio for more than 3 seconds. Provide a means for users to start audio playback when they desire (e.g., a "play" button).

Ref:

n/a

6.3 - Provide text transcripts for audio containing speech when it is provided to the public and/or required to be viewed by employees.

What:

"Audio containing speech" includes audio recordings or live broadcasts of speeches, seminars, conferences, etc. A text transcript is a word-for-word written record of the spoken content of such an event.

Why:

Individuals who are deaf or hard of hearing may require text transcripts to access audio information.

How:

Provide a link to an HTML or text transcript of any audio presented on a web site. Transcripts should be posted at the same time the audio is made available. Communication Access Realtime Translation (CART) providers can transcribe live events.

Ref:

WCAG 1.1; 508 a

7. Multimedia

7.1 - Provide synchronized captions for all multimedia that contains essential auditory information when it is provided to the public and/or required to be viewed by employees.

What:

Multimedia generally refers to recorded or live media containing both video and audio tracks. Captions are essentially a text transcript of the audio synchronized with the audio/video tracks of the presentation.

Why:

Individuals who are deaf or hard of hearing may require captions to access the audio information in multimedia.

How:

Whenever possible, captions should be implemented using Synchronized Multimedia Integration Language (SMIL) If multimedia contains essential audio but no essential video, a text transcript may be provided instead of captions.

Ref:

WCAG 1.4; 508 b

7.2 - Provide audio descriptions for all multimedia that contains essential visual information when it is provided to the public and/or required to be viewed by employees.

What:

Audio descriptions are verbal descriptions of the actions and images displayed in a video that are inserted during pauses in the regular dialogue or audio track. Audio descriptions are only necessary if significant information that is presented visually is not discernable from the dialogue or audio track.

Why:

Individuals who are blind or low-vision may require audio descriptions to access the visual information in multimedia.

How:

Carefully consider whether audio descriptions are necessary to present the significant information of a multimedia recording. Many speech-intensive events, such as speeches, lectures, or conferences, do not contain essential

video and, therefore, do not need audio description. When necessary, audio descriptions are usually best implemented by a professional "audio describer."

Ref:

WCAG 1.3; 508 b

8. Animation

8.1 - Provide a means of pausing any moving, blinking, scrolling, or auto-updating information.

What:

Animated graphics, Flash, Java, `<blink>` tags, `<marquee>` tags, and other techniques are often used to create a variety of animated effects.

Why:

Some users with disabilities are not able to read text that is moving. Animation can be distracting to users with visual or cognitive disabilities.

How:

Avoid animation and movement unless it provides significant additional information. If animation is used, provide a means of pausing the animation.

Ref:

WCAG 7.1, 7.2, 7.3

8.2 - Do not include content that flashes faster than 3 times per second.

What:

Animated graphics, Flash, Java, `<blink>` tags, and other techniques can be used to cause content to flash.

Why:

Flashing faster than 3 times per second can trigger epileptic seizures.

How:

Do not include content that flashes faster than 3 times per second.

Ref:

WCAG 7.1; 508 j

9. Links

9.1 - Ensure that links are understandable out of context.

What:

A link is understandable out of context when it clearly indicates its destination or function without requiring additional information.

Why:

Screen reader users often "tab" through links (skip from link to link by pressing the `Tab` key) in order to "scan" a page. Most screen readers also offer a "links list" feature to help speed the process of navigating to specific links. Links that are not understandable out of context, such as "click here" or "more," make these techniques much less efficient. Some screen readers can be configured to read link `title` attributes *instead* of link text, however most currently read only link text by default.

How:

Use link text that is clear and unambiguous. Link text should usually match the title of the page to which the link points. Ensure that links that point to the same URL use the same link text, and that links that point to different URLs use different link text. If `title` attributes are used, repeat the text of the link as the beginning of the `title`, followed by the additional information.

Note: Portions of links may be hidden visually but revealed to assistive technologies using a style such as:

```
.invisible { position: absolute; left: -10000px; width: 1px;
height: 1px; overflow: hidden; }
```

Ref:

WCAG 13.1

9.2 - Provide a means of skipping past repetitive navigation links.

What:

Navigation links are the lists or "menus" of links to all the sections of a site that are often repeated on every page.

Why:

Because navigation links are typically placed at the beginning (top left) of pages, screen reader users must read through all the navigation links before reaching the main area of the page. Individuals who use a keyboard instead of a mouse similarly must tab through all the navigation links before reaching the main area of the page. Providing a means of skipping these links can significantly improve efficiency and usability for screen reader and keyboard users.

How:

Provide a link at the beginning of navigation lists that points to a target at the beginning of the main content area of the page. This link must be visible to screen reader and keyboard users, but can be hidden from other users. It is also acceptable to design a page so that navigation links come at the end of the document.

Ref:

WCAG 13.6; 508 o

9.3 - Avoid using small links.

What:

The size of the "clickable" area of a link is limited to the size of the image or text that makes up the link.

Why:

Mouse-users with limited fine motor control may have difficulty pointing to and clicking on links that are small, especially if the links are close together.

How:

Make sure that images used for links are reasonably large, preferably 16 pixels by 16 pixels or larger. Use standard or enlarged font sizes for text links, and avoid using text links that are shorter than 4 characters in length. Avoid placing small links close together.

Ref:

n/a

9.4 - Ensure that same-page links move keyboard focus as well as screen focus.

What:

Same-page links are links that target another location on the same page. The target is usually indicated with a "named anchor" (e.g.,). When a same-page link is clicked, the screen should scroll and keyboard focus should move to the target location on the page.

Why:

Users with physical or visual impairments may not be able to use a mouse. Same-page links can enable these users to more quickly and efficiently move about a page. Unfortunately, Internet Explorer 5, 6 & 7 do not reliably move keyboard focus to the target of same page links.

How:

To ensure that same-page links work correctly in Internet Explorer 5, 6 & 7, set tabindex="-1" on same-page link targets.

Ref:

n/a

10. Forms

10.1 - Provide labels or titles for all form fields.

What:

HTML forms include "fields" such as text boxes (<input type="text">), check boxes (<input type="checkbox">), radio buttons (<input type="radio">), and drop-down lists (<select>). Each field is typically identified by a text label positioned above, before, or below the field. In HTML, labels are identified using the <label> tag.

Why:

Screen readers cannot always determine which label belongs to which field based on positioning alone. Proper use of the HTML <label> element and/or title attribute makes this association clear.

How:

Use a <label> element whenever possible to identify each form field's label. Set the label element's for attribute to match the corresponding field's id. Add a title attribute to any form field that cannot be associated with a <label>. Because screen readers typically recognize *either* the <label> *or* the title (not both), the title must provide the full label, and a <label> should *not* be associated with the field.

Note: In most web browsers, label associations can be checked by clicking the label with the mouse. If the label is properly associated, focus will move to the corresponding field.

Ref:

WCAG 12.4; 508 n

10.2 - Provide legends for groups of form fields.

What:

In some cases, several form fields may need to be grouped together. For example, a set of radio buttons may provide different "answers" to a single "question."

Why:

Screen readers need to be able to read the group name or "question" in addition to each field's individual label.

How:

If possible, group related fields within a `<fieldset>` element and provide the group name or "question" in the fieldset's `<legend>`. If a fieldset cannot be used, apply a `title` attribute to each field in the group including the group name and the field's label. Because screen readers typically recognize either the `<label>` *or* the `title` (not both), the `<label>` elements should *not* be associated with their fields when using this technique. Also note that groups of radio buttons (and sometimes checkboxes) can usually be replaced by a single drop-down list or list box (`<select>`) with the "question" as the `<label>` and the "answers" as `<option>` elements.

Ref:

WCAG 12.3; 508 n

10.3 - Ensure that form fields are in a logical tab order.

What:

Screen reader and keyboard users move between form fields (and links) using the `Tab` key. The order in which form fields receive focus is called the "tab order." By default, the tab order follows the order in which elements appear in a page's HTML code.

Why:

Depending on the design and layout of a page, the tab order may not match the visual (or logical) order of fields on a form. Reading fields out of their intended order can be disorienting for a screen reader or keyboard user.

How:

Make sure that fields appear in logical order in the HTML code or use the `tabindex` attribute on each field to set the appropriate order.

Note: Any element with a positive `tabindex` will come before all elements without `tabindex` in the tab order. As a result, `tabindex` must usually be applied to *all* or *none* of the focusable elements on a page.

Ref:

WCAG 9.4; 508 n

10.4 - Avoid placing non-focusable text between form fields.

What:

Special instructions for completing form fields may be listed after or between the fields to which they apply. Usually, this text is non-focusable, that is, it does not receive focus when a user tabs through the form fields.

Why:

When filling out a form, screen readers typically use a special "forms mode" in which they interact only with focusable elements, including form fields, associated labels, buttons, and links. In "forms mode," screen readers do not read non-focusable text.

How:

Instructions should be given within field labels if possible. If instructions are too long to fit within labels, they should be provided in an instructions section before the beginning of the form. If neither of these approaches works, consider using a technique to make the text elements focusable.

Note that disabled form fields (`disabled="disabled"`) are non-focusable; fields may be made read-only (`readonly="readonly"`) to keep them focusable.

Ref:

508 n

10.5 - Ensure that text in form fields can be enlarged.

What:

Most web browsers allow users to easily adjust the size of text on a page.

Why:

Users with limited vision often rely on the ability to enlarge text. However, Internet Explorer 5, 6 & 7 do not change the size of text within form fields to match settings selected from the View, Text Size menu.

How:

To ensure that text in form fields can be easily resized in Internet Explorer 5, 6 & 7, include a style rule such as the following:

```
input, select, textarea, button { font-size: 100%; }
```

Ref:

n/a

11. Tables

11.1 - Identify a header cell for each column and row in simple data tables.

What:

In HTML, the table element (`<table>`) is used to display data. "Simple" tables have a uniform number of columns and rows. Table header cells (`<th>`) are used to provide a label for each row and column.

Why:

A screen reader can use table headers to provide row and column information while a user explores the data cells within a table.

How:

Use `<th>` elements with `scope="col"` (for column headers) or `scope="row"` (for row headers) to identify column and row header cells. The cell that contains the most uniquely identifying information for the column or row should be the header; usually, this should be the first cell of each column and row. Do not include unnecessary columns or rows for formatting.

Ref:

WCAG 5.1; 508 g

11.2 - Identify relationships in complex data tables using id and headers attributes.

What:

Tables become complex when they have "spanned" columns or rows, multiple layers of headers, or are divided into multiple sections. The `id` and `headers` attributes can be used to associate data cells with multiple row and column headers in complex data tables.

Why:

Modern screen readers can use the information provided with `id` and `headers` attributes to interpret relationships in complex tables.

How:

Whenever possible, simplify complex tables by re-arranging or dividing them into separate tables. If a table cannot be simplified, use a unique `id` attribute on each header cell, for example: `<th id="header1">`, and a `headers` attributes on each data cell to list the `id` attributes of header cells that it relates to, for example: `<td headers="header1 header2 header3">`. Do not include unnecessary columns, rows, colspans, or rowspans for formatting.

See W3C's "Tables in HTML Documents" (http://www.w3.org/TR/html401/struct/tables.html#h-11.4.1) for more information on how to markup complex tables.

Ref:

WCAG 5.2; 508 h

11.3 - Provide summary attributes for data tables.

What:

The `summary` attribute provides the name and, possibly, a brief description of a table's content and structure. The summary is not displayed on the screen, but is read by most screen readers.

Why:

By reading a table's summary, a screen reader-user can quickly decide whether to take the time to read a table in detail; it also helps to let the screen reader-user know what to expect regarding the table's structure.

How:

Provide the name of the table in the `summary` attribute of the `<table>` tag. A brief description of the content and structure of the table may also be included in the summary. (No summary should be used if a table is used for layout.)

Ref:

WCAG 5.5

12. Frames

12.1 - Provide concise, unique, and understandable titles for frames.

What:

HTML frames and iframes are used to divide web pages into separate areas, each displaying a separate web page. Each frame is identified by its `title` attribute and each page contained within a frame is identified by its `<title>` element.

Why:

To navigate pages with frames, users who are blind must be able to identify the different frames and understand the purpose of each frame. Most screen readers identify frames by speaking the title of each frame.

How:

Give each frame and iframe a concise, unique, understandable `title` attribute that indicates the frame's function and is unique within the frameset. Do not include the word "frame." Set the `<title>` element of each page contained within a frame to match its `title` attribute or to identify the current content of that frame.

Ref:

WCAG 12.1; 508 i

12.2 - Avoid using hidden, empty, or non-essential frames.

What:

Frames or iframes are sometimes used for formatting, layout, or other technical purposes. For example, hidden frames may be used to hold information to be transmitted between pages.

Why:

Screen readers cannot judge whether the content of a frame is significant and identify every frame for the user. Having to read this extraneous information can be time consuming and confusing.

How:

Use frames sparingly. If a frame is not necessary for page content, eliminate it. Minimize the nesting of frames. Recognize that information in hidden frames may be read by screen readers.

Ref:

n/a

13. Scripts

13.1 - Ensure that scripted functions are usable with assistive technologies.

What:

Scripting languages such as JavaScript are simple programming languages that can be used within a web browser to automate tasks and enable pages to respond to user input. Scripts are often used to dynamically show or hide the content that appears on a web page or to perform important functions, such as checking that entries in form fields are valid.

Why:

Assistive technologies may interact with scripts in unexpected ways. For example, some assistive technologies may not be able to trigger some script events, and other assistive technologies may not recognize changes to content made using scripts. However, assistive technology users need to be able to access the same essential content and functionality whether scripts are fully, partially, or not supported. It is not safe to assume that users with disabilities will have scripting support turned off.

How:

Whenever scripts are used, it is the responsibility of the page developer to thoroughly test using assistive technologies to ensure that all information and functionality is accessible. Testing should confirm that content is usable with system large fonts and high contrast display settings, that interface elements are focusable and operable using the keyboard, that tab and reading order are appropriate, and that all content can be identified and operated with leading

assistive technology tools. If there is any doubt, err on the safe side by ensuring that the essential elements of the page do not rely on scripts.

Scripting features that are purely decorative and do not present any significant information or functionality do not need to be made accessible.

Ref:

WCAG 6.2, 6.3, 6.5, 8.1; 508 l

13.2 - Ensure that significant interactions can be performed with both keyboard and mouse.

What:

Scripts can trigger changes when the user performs specific actions. For example, an image can change color when the mouse pointer hovers over it (i.e., the `onmouseover` event). Different events are triggered by either mouse or keyboard actions.

Why:

Users with physical impairments may be able to use the keyboard but not the mouse. Individuals who cannot see the mouse pointer on the screen must use the keyboard for all interactions. Scripts that can only be triggered by the mouse are not accessible to these individuals.

How:

Whenever using a mouse-only event (e.g., `onmouseover`, `onmouseout`) to trigger a significant script action, use the corresponding keyboard event (e.g., `onfocus`, `onblur`). Ensure that scripts are attached to elements that can receive keyboard focus, such as links and form fields. If necessary, use a technique to make elements focusable, such as by using script to set `tabindex="0"`.

Also, make sure that keyboard events do not unintentionally trigger script actions. For example, when a keyboard user arrows through the options in a drop-down list (`<select>`), the onchange event fires once for each option; if a script attached to this event reloads the page (or causes some other significant change), it may be impossible for a keyboard user to operate the control.

Ref:

WCAG 6.4, 9.2, 9.3; 508 1194.21 a

13.3 - Avoid changing focus unexpectedly.

What:

Scripting can be used to move focus from one element to another, load a new page, or open a new window.

Why:

Users may be disoriented by changes that they do not expect.

How:

In most cases, changes to focus, location, or the current window should be initiated by a user activating (clicking) a link or button. If changes are initiated by other actions, make sure that there is an obvious cause-and-effect relationship between the action and the change. If changes are not likely to be expected, it may be necessary to provide explicit instructions to users before the changes occur.

Ref:

WCAG 8.1; 508 l

13.4 - Avoid changing content unexpectedly.

What:

Scripting can be used to add or change content on a web page.

Why:

Users, especially those using screen readers and screen magnifiers, may not notice that content has changed on a different part of the page. If changes occur in parts of the page that have already been read, the user is not likely to "back up" to find the new content.

How:

In most cases, content should only be changed or added after the current point of focus in the tab/reading order. For example, if entering a value in a field causes text to appear, the text should appear after that field in the tab/reading order. It is also important to ensure that changes happen quickly enough that they are complete before the user reaches the changed element. For example, if selecting an item from a list box changes the value in a subsequent field, the change must be complete before the focus moves to the subsequent field. If content is changed above the user's focus, or if changes occur slowly enough that a user may move past the changed element before the change is complete, it may be necessary to provide an alert box or other instruction to direct the user to the changed element.

Ref:

WCAG 8.1; 508 l

14. Embedded Objects

14.1 - Use accessible embedded objects whenever possible.

What:

"Objects" include a variety of non-HTML technologies, such as Java and Flash, that can be embedded within web pages. These technologies require additional software to be downloaded, installed, and run before the content can be viewed or used. Embedded objects operate with their own user interfaces, which are separate and different from that of standard web pages.

Why:

Because embedded objects have their own interfaces, they must be accessible in and of themselves. If essential content or functionality is presented using an object that is not accessible, it will not be usable by individuals with disabilities.

How:

Check with the manufacturer and/or developer to determine if and how the technology can be made accessible.

Ref:

WCAG 8.1; 508 m

14.2 - If an inaccessible embedded object must be used, provide an accessible alternative that includes the same content and functionality.

What:

If an embedded object cannot be made accessible, it may be possible to provide both the original object and an equivalent accessible alternative.

Why:

The same features that make an embedded object inaccessible to some users may actually improve accessibility or usability for others. By providing both the original and accessible versions, the same content and functionality can be available to all users.

How:

Wherever an inaccessible object is embedded, also provide or link to an accessible version. Make sure that the information and functionality is completely equivalent and up-to-date. Be sure to consider whether the inaccessible version is actually necessary. In cases where it is impossible to create an equivalent accessible version, such as with some geographical imaging and mapping systems, exceptions may be necessary.

Ref:

WCAG 6.2, 11.4; 508 k

15. Downloadable Documents

15.1 - Provide natively accessible downloadable documents whenever possible.

What:

Downloadable documents are often provided in formats such as Adobe Acrobat Portable Document Format (PDF), Microsoft Word, Microsoft PowerPoint, etc. Each document format has its own accessibility issues and may or may not provide appropriate techniques to address accessibility. A document is considered natively accessible if it follows the accessibility techniques for its format.

Why:

Because non-HTML documents cannot implement HTML accessibility techniques, they must be accessible in and of themselves. If essential content or functionality is presented in a downloadable document that is not accessible, it will not be usable by individuals with disabilities.

How:

Check with the manufacturer and/or publisher of the format of a downloadable document to determine if and how it can be made accessible. If accessibility techniques exist, ensure that the downloadable document fully implements these techniques and meets the functional performance criteria defined in these standards.

Note: See Adobe's Accessibility Resource Center (http://www.adobe.com/accessibility) for information about PDF accessibility.

Ref:

n/a

15.2 - If a downloadable document cannot be made natively accessible, provide an accessible alternative that includes the same content and functionality.

What:

If a downloadable document cannot be made natively accessible, it may be possible to provide both the original document and an equivalent accessible alternative.

Why:

By providing both the original and accessible versions, the same content and functionality can be available to all users.

How:

Wherever a link is provided to an inaccessible document, also provide a link to an accessible version, preferably in HTML format. Make sure that the information and functionality is completely equivalent and up-to-date. Be sure to consider whether the inaccessible version is actually necessary. If an accessible alternative cannot be provided electronically, provide information on how to obtain an alternate format (e.g., large print, Braille, audio recording) in a timely manner.

Ref:

WCAG 11.1, 11.4; 508 k

16. Timing

16.1 - Notify users of time limits and provide a means to extend time if possible.

What:

Some web pages, frequently those that require a user to log in, time-out after a certain period of inactivity and require the user to start over.

Why:

Users with visual, physical, or cognitive disabilities may require more time than average to read and interact with a web page.

How:

Provide a clear explanation of any time limits. If possible, offer the user a way to extend or remove the limits. Avoid using time limits unnecessarily.

Ref:

508 p

16.2 - Do not automatically refresh the current page.

What:

It is possible to cause web pages to automatically re-load their content on a certain interval. For example, a page containing news headlines might refresh every few minutes to present the most current items.

Why:

When a page automatically refreshes, it can cause a screen reader to re-start reading from the beginning of the page.

How:

Do not use `http-equiv="refresh"`. If necessary, provide a link or control to allow the user to refresh a page at his or her discretion.

Ref:

WCAG 7.4, 7.5

17. Page Layout

17.1 - When using tables for layout, ensure that reading order is logical.

What:

HTML tables are sometimes used to lay out web pages in multiple columns and sections (instead of actually presenting data). "Reading order" refers to the order in which a screen reader would read through the table.

Why:

Screen readers read through tables in the order in which cells are defined in the HTML code, which can be very different from the order that someone reading visually would follow. It is essential that the reading order match the logical flow of the document so that a screen reader user would hear the document in the same order that a visual reader would read it.

How:

Avoid using tables for layout whenever possible. If tables are used for layout, check the reading order by following the order in which the table cells appear in the code. If necessary, restructure the table to achieve an appropriate reading order.

Ref:

WCAG 5.3

17.2 - When using style sheets for layout, ensure that reading order is logical.

What:

The positioning features of Cascading Style Sheets can be used to position elements visually almost anywhere on a web page.

Why:

Screen readers read through the elements on a web page in the order in which they appear in the page code, regardless of how they are positioned using style sheets. It is essential that the reading order match the logical flow of the document so that a screen reader user would hear the document in the same order that a visual reader would read it.

How:

Check the reading order by following the order in which elements appear in the HTML code. Adjust the reading order by rearranging the order in which elements are defined in the code.

Ref:

WCAG 6.1; 508 d

17.3 - Avoid horizontal scrolling.

What:

If a web page is wider than the window or screen in which it is viewed, most browsers will display a horizontal scroll bar and require the user to manually scroll to see the entire page.

Why:

When a screen magnifier enlarges a web page, it also reduces the field of view so that the user must pan (scroll) to see the entire page. When the web page being viewed also requires horizontal scrolling, the combination can be awkward or unusable. Keyboard users may also find repetitive scrolling fatiguing and inefficient.

How:

Design pages so that they can resize to fit the width of the user's browser. Use relative widths and check for horizontal scrolling using a screen resolution of

800 by 600 pixels. If scrolling cannot be avoided, place the least important content in the rightmost part of the page.

Ref:

n/a

18. Alternate Accessible Versions

18.1 - Use separate accessible versions only as a last resort.

What:

Separate accessible or "text-only" versions are often offered instead of providing a single accessible site.

Why:

In practice, "text-only" versions are rarely kept complete or up-to-date. Given advances in accessibility techniques and assistive technologies, "text-only" sites are simply not necessary in most cases.

How:

Follow these standards to develop a single site that is universally accessible and efficient to maintain.

Ref:

WCAG 11.4; 508 k

Source: http://www.dhs.state.il.us/IITAA/IITAAWebImplementationGuidelines.html