Illinois Department of Innovation & Technology Accessibility Testing Standards

6/23/2023

Standards

As required by the Illinois Information Technology Accessibility Act (PA 095-0307/30 ILCS 587) accessibility testing will be performed to confirm conformance with:

- Web Content Accessibility Guidelines (WCAG) 2.0 Level AA (https://www.w3.org/TR/WCAG20)

WCAG 2.1 and Level AAA criteria may also be tested but should be identified as such in reports.

Testing Tools

Accessibility testing will be performed with the following tools/techniques:

- Keyboard Commands (https://doit.illinois.gov/initiatives/accessibility/guides/web/keyboard-testing)
- High Contrast & Zoom (https://doit.illinois.gov/initiatives/accessibility/guides/web/visual-testing)
- Colour Contrast Analyser (https://doit.illinois.gov/initiatives/accessibility/guides/color)
- Axe DevTools Browser Extension (https://www.deque.com/axe-devtools-accessibility-testing) and/or Accessibility Insights for Web (https://accessibilityinsights.io/downloads)

Additional testing may be performed by trained testers using the following assistive technology tools:

- NVDA screen reader (https://www.nvaccess.org)
- JAWS screen reader (https://www.freedomscientific.com/products/software/jaws)
- ZoomText screen magnifier (https://www.freedomscientific.com/products/software/zoomtext)
- Dragon Professional (https://www.nuance.com/dragon/business-solutions/dragon-professional)

Impact Ratings

If functional impact ratings are provided, the following scale will be used:

- Critical – Will prevent some users from completing essential tasks
- High – Will be very difficult or confusing to some users, but means of completing task exist
- Med – May be difficult or confusing to some users, but means of completing task exist
- Low – Violates accessibility standards but unlikely to significantly affect users

Testing Processes

Accessibility testing will include the following processes:

1. Keyboard Testing
2. Visual Testing
3. Automated Testing
4. Assistive Technology Testing

1. Keyboard Testing

Keyboard testing is performed using standard keyboard commands *only* – the mouse should *not* be used. Keyboard testing may be completed by following existing test scripts substituting keyboard commands for mouse actions. If test scripts are not available, keyboard testing may be performed a screen/page at a time, completing all the functions available on the screen/page.

1. Browse to the screen to be tested.
2. If necessary, explore the screen with the mouse to identify all interactive elements.
3. Click in the browser address bar; then set the mouse aside.
4. Press the Tab key several times to move focus past any browser toolbars and into the page.
5. Use the following keyboard commands to move to and operate all interactive elements:

Command	Function
Tab	Move to the next interactive element (link or control).
Shift + Tab	Move backwards to the previous interactive element (if necessary).
Alt + Down Arrow	Open a dropdown list.
Up/Down Arrow	Move vertically through options in a dropdown list, list box, radio button group, menu, or rows in a grid.
Right/Left Arrow	Move horizontally in a menu bar, tab list, or grid.
Spacebar	Check a checkbox or click a button.
Enter	Click a link or select an item in a list box or menu.
Other	For other types of controls, look up and use the Keyboard Interactions in the ARIA Authoring Practices Guide (APG) (https://www.w3.org/WAI/ARIA/apg/patterns)

6. At each step, check for and document any:
   - Interactive elements that do not receive keyboard focus. (WCAG 2.1.1)
   - Instances where focus doesn't follow a logical order. (WCAG 2.4.3)
   - Elements that cause unexpected changes when they receive focus. (WCAG 3.2.1)
   - Elements that do not show a visual indicator (e.g., outline) when focused. (WCAG 2.4.7)
   - Elements than cannot be operated using standard keyboard commands. (WCAG 2.1.1)
   - Elements that cause unexpected changes when values are entered/changed. (WCAG 3.2.2)
   - Elements that trap focus (do not allow it to move to the next element). (WCAG 2.1.2)
7. If any of the issues listed above are found, the keyboard test fails. In the ticket, clearly identify the element and which check it failed. Include steps to reproduce any unexpected behavior.

2. Visual Testing

Visual testing is performed using a combination of a Window High Contrast color theme and browser Zoom set to 200%:

1. Press left Alt + left Shift + PrtSc (Print Screen). If prompted to turn on High Contrast, click Yes. By default, Windows will activate a theme with a black background, white text, yellow links, green disabled text, and light blue selection highlights. (If High Contrast does not activate, check Windows Control Panel > Ease of Access Center > Make the computer easier to see, and confirm "Turn on or off High Contrast…" is checked.)

2. In the browser, open the Settings menu (Alt + F) and set Zoom to 200%.

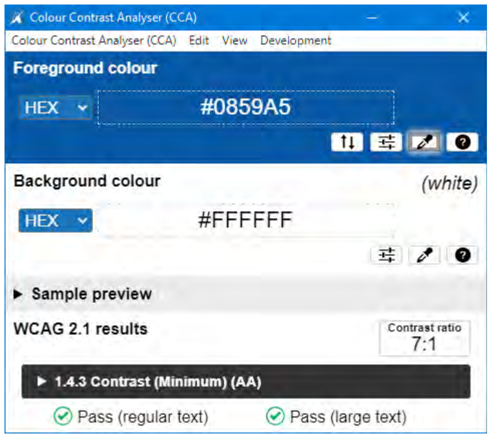
3. Browse to the screen to be tested. If there are elements of the screen that are not initially displayed, such as collapsed sections, display as many of them as possible.

4. Visually review all elements of the screen *except* logos, decorative (meaningless) images, or images of text that are duplicative of actual text provided elsewhere on the screen. Check for and document:

   - Text that does not take on high-contrast colors. (WCAG 1.4.5)
   - Information that was conveyed by color that is now not shown. (WCAG 1.4.1)
   - Text that did not increase in size by 200%. (WCAG 1.4.4)
   - Text that was truncated, overlapped, or otherwise became unreadable. (WCAG 1.4.4)

5. For any text that did not take on high-contrast colors (*except* logos, decorative, or duplicative images), use the Colour Contrast Analyser (https://www.tpgi.com/color-contrast-checker) to sample the foreground (text) color and background colors. Check for and document:

   - Regular-sized text with a contrast ratio less than 4.5:1. (WCAG 1.4.3)
   - Large text (24px or 19px bold or larger) with a contrast ratio less than 3:1. (WCAG 1.4.3)


Colour Contrast Analyser (CCA)
Colour Contrast Analyser (CCA) Edit View Development
Sample preview
1.4.3 Contrast (Minimum) (AA)

Foreground colour
HEX #0859A5

Background colour (white)
HEX #FFFFFF

WCAG 2.1 results
Contrast ratio 7:1
Pass (regular text) Pass (large text)

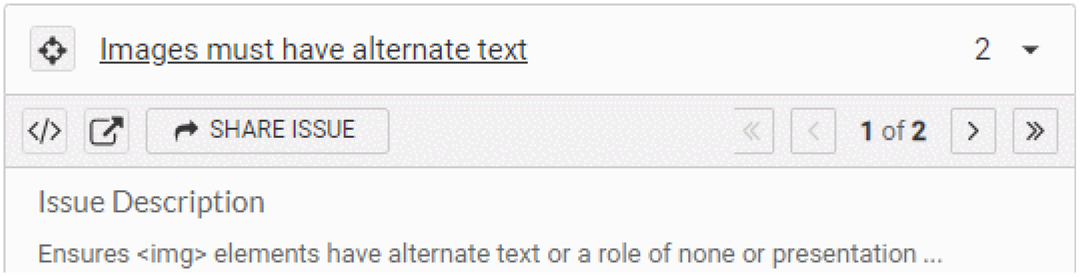
   Note: The Colour Contrast Analyser may also be used to check the contrast of text elements with High Contrast turned *off*, however text contrast issues will also be identified by automated testing. (Automated testing will *not* find contrast issues in images.)

6. If any of the issues listed above are found, the visual test fails. In the ticket, clearly identify the element and which check it failed. In the case of color contrast failures, include the hexadecimal color codes of the foreground and background colors and the computed color contrast ratio.

3. Automated Testing

If automated accessibility testing is not included as part of a continuous integration and delivery (CI/CD) pipeline, it should be performed using a browser extension tool. Automated testing is performed one screen/page at a time. All findings for a given screen may be reported in a single ticket, although, if a screen has multiple states, such as expanding/collapsing sections or pop-up dialogs, it may be necessary to run the test multiple times in the different states.

1. If necessary, install axe DevTools browser extension from Chrome Web Store or Edge Add-ons.
2. Open the browser, display Developer Tools (F12), and select axe DevTools from the main toolbar.
3. Browse to the screen to be tested. If there are elements of the screen that are not initially displayed, such as collapsed sections or pop-up dialogs, display as many of them as possible. (If there are elements that cannot be displayed simultaneously, re-run the test in each state.)
4. In the axe DevTools panel, click "Scan ALL of my page" (or "Re-Run Scan" if re-testing).
5. If axe reports any Critical, Serious, Moderate, or Minor issues, the automated test fails. If there are no issues, or only Needs Review or Best Practices, the automated test is considered a pass.
6. If there are any issues regarding ARIA, or any other issues that are unclear or questionable, refer them to the DoIT Office of Information Accessibility (DoIT.Accessibility@Illinois.gov) before reporting.
7. For any issues that do *not* require review by the DoIT Office of Information Accessibility, copy and paste the top line issue descriptions into a single ticket per screen.
8. Developers should follow the steps above to use axe DevTools to see the specific details of each issue.



Issue Description

Ensures <img> elements have alternate text or a role of none or presentation ...

Use the following buttons in the axe interface to identify the details of the issue:

Toggle Highlights (upper left) will highlight the issue in the screen/page.

Inspect Issue will show the issue HTML code in the Developer Tools Elements panel.

More Info will open a web page explaining the issue and possible fixes.

Next Issue will show additional occurrences of the issue if more than one was found.

4. Assistive Technology Testing

Assistive technology testing should *only* be performed by testers who have been trained to use assistive technology tools. Testers must be certain not to “cheat,” for example by using the mouse or visually reading information on the screen when testing with a screen reader. Do not test with assistive technology unless you are *completely* confident in your ability to use it *as it would be used by a user*.

Assistive technology testing should be performed *after* automated, visual, and keyboard testing has been completed and *passed*. Assistive technology testing is normally performed only on a *representative sample* of screens/pages. The representative sample should be selected by someone with knowledge of the scope and functions of the application and should include:

- Screens required for the most essential functions of the application, and
- Examples of each design patterns or interface component not present on the essential screens.

All findings for a given screen may be reported in a single ticket OR findings may be reported individually, including steps to reproduce, depending on complexity and requirements of the system developer/vendor.

Screen Reader Testing

Screen reader testing should be performed using the browser and screen reader most likely to be used by users of the system, i.e., Edge & JAWS for internal and Chrome & NVDA for public applications. For details, see Screen Reader Testing Quick Reference (https://onenet.illinois.gov/page.aspx?item=135963)

Screen Magnifier Testing

Screen magnifier testing should be performed using ZoomText with magnification set to 4 x, color enhancement active, and speech disabled.

Speech Recognition Testing

Speech recognition should be performed using Dragon Professional or Windows Speech Recognition. Voice training should be completed, and recognition accuracy confirmed before testing. Preference should be given to commands targeting specific elements, such as “click first name.” Mouse movement commands, including mouse grid, should be used only if specific commands do not work.

Documentation

When reporting multiple issues in a single ticket, provide a list of all the issues in a format such as:

Issue	Standard	Impact
Concise description including element(s) affected; recommendation (opt.)	x.x.x (ver lvl)	see scale

For example:

Issue	Standard	Impact
Language of the page is not specified; add html lang="en"	3.1.1 (2.0 A)	Low
Country dropdown causes page to reload when selected option is changed; add a 'select' button or attach to blur instead of change event	3.2.2 (2.0 A)	High
Submit button does not receive keyboard focus; remove tabindex="-1"	2.1.1 (2.0 A)	Critical

When issues must be reported individually, use the following format:

Title: [Application - Screen - Issue (YYYY-MM-DD)]

URL:

Impact: [see scale]

Testing Environment

OS: [name and version]
Browser: [name and version]
Testing Tools: [name(s) and version(s)]

Screen Shot:

WCAG Violations:

[x.x.x (ver lvl): short name]

Steps to Reproduce:

1.
2.
3…

Expected Behavior:

1.
2.
3…

Notes: [optional, including code snippets and/or recommended corrections]

If necessary, a screen recording of the Steps to Reproduce in MP4 format may also be included.