COAX

WCAG Compliance Checklist for 2026

Overview

The European Union has introduced the **European Accessibility Act**[1] **(EAA)** to ensure that digital services are accessible to everyone, regardless of their abilities. At the heart of the EAA lie the Web Content Accessibility Guidelines (WCAG), international practices to make online content and applications inclusive. WCAG focuses on four principles: **Perceivability, Operability, Understandability,** and **Robustness**. There are three levels of WCAG compliance:

Level A (minimal)	Level AA (recommended)	Level AAA (enhanced)
covers the basic requirements for web content, ensuring overall usability for most users.	makes your digital presence work well for the vast majority of users, including those using assistive technology, which is what EAA demands to be completely implemented in 2026 and beyond.	targets the needs of users with very specific accessibility requirements. Not always necessary for all websites and applications.

Implementing the principles and multiple criteria is a great technical challenge. This checklist simplifies WCAG 2.1 AA requirements to help your team implement accessibility standards effectively. However, some criteria have exclusions not allowed on stricter levels (they will be marked *cursive*).

For exhaustive guidelines, visit the **WCAG Quick Reference**[2] and the **updated information**[3] on the 2026-2027 deadlines. To explore all recommended tools for testing and evaluation, check the **WAI Testing Tools List**[4]**.**

No matter what digital services you provide in the EU, you need to ensure your online presence meets the **WCAG 2.1 Level AA standards by April 24, 2026**. If you don't comply, your company will be subject to fines and corrective measures.

[1] https://ec.europa.eu/social/main.jsp?catId=1202
[2] https://www.w3.org/WAI/WCAG22/quickref/?versions=2.1
[3] https://wpvip.com/blog/ada-website-accessibility-deadline-2026/
[4] https://www.w3.org/WAI/test-evaluate/tools/list/

Fines and Corrective Measures by Country

Country	Fines range	Corrective measures
Austria	Up to €200,000	Higher fines and potential service suspensions for repeated violations
Belgium	€1,000 to €50,000	Business operation suspensions for continuous non-compliance
Bulgaria	Substantial fines	Restrictions on business operations until compliance is achieved
Croatia	€2,000 to €50,000	Public notices and further legal actions for persistent non-compliance
Cyprus	€1,000 to €20,000	Additional penalties for continuous violations
Czech Republic	Up to €100,000	Potential additional costs for corrective actions
Denmark	€10,000 for initial non-compliance	Increasing fines for repeated offenses
Estonia	€5,000 to €50,000	Businesses required to undertake corrective measures
Finland	Up to €150,000	Mandatory accessibility audits and documentation requirements
France	€5,000 to €250,000	Potential public exposure of non-compliant businesses
Germany	Up to €500,000	Corrective actions and potential service suspensions
Greece	€2,000 to €100,000	Additional penalties for continuous non-compliance
Hungary	€3,000 to €50,000	Potential service suspensions
Ireland	Up to €200,000	Mandatory accessibility audits
Italy	€5,000 to €150,000	Public notices and mandatory improvements
Latvia	Up to €100,000	Additional requirements for corrective measures
Lithuania	€2,000 to €50,000	Mandatory accessibility improvements
Luxembourg	Up to €150,000	Public notices about non-compliance
Malta	€1,000 to €50,000	Additional penalties for continuous violations
Netherlands	Up to €250,000	Service suspensions for non-compliance
Poland	Up to €200,000	Mandatory accessibility audits

Country	Fines range	Corrective measures
Portugal	€5,000 to €100,000	Required corrective actions
Romania	Up to €100,000	Enforcement actions for continuous non-compliance
Slovakia	€2,000 to €50,000	Additional corrective measures
Slovenia	Up to €100,000	Public notices and mandatory improvements for non-compliance
Spain	€5,000 to €300,000	Required corrective measures and potential public exposure
Sweden	Up to €200,000	Mandatory accessibility audits and corrective actions

Need Help with WCAG Compliance?

WCAG compliance is not the kind of thing you want to leave to chance. If you don't have technical experts in-house, you have the option to partner with professionals who specialize in accessibility solutions.

With over 15 years of experience working with clients from the EU and the UK, COAX understands how to tackle the complexities of various types of data handling and web accessibility compliance. Our expertise in **web development** and **quality assurance** sets the foundation for assisting you with the technical side of the WCAG compliance process.

COAX team can conduct end-to-end audits of your website, mobile applications, and digital content, evaluating them against the WCAG guidelines. Once we identify areas that need improvement, we provide detailed reports and outline an actionable plan. Our team will help you implement the necessary changes, ensuring your digital platforms are fully optimized for all users.

Web Accessibility Checklist (simplified)

PRINCIPLE 1

Make Content Easy to See, Hear, and Understand

1. Visibility and Comprehension of Content and Controls

Information and user interface components must be presented to the users so they can clearly and easily see, hear, and understand.

- **Screen Reader Ready:** Make sure all non-text content (e.g., images, audio, video) has a text description so people using screen readers or other accessibility tools can understand it.

Use a screen reader to check if non-text items have clear and understandable descriptions.

```
<img
    src="img_girl.jpg"
    alt="Young woman wearing a blue winter jacket standing outdoors"
    width="500"
    height="600"
    loading="lazy"
>
```

- **Check Control Descriptions:** If a user needs to click, type, or interact with a non-text element (like a button), give it a clear name that explains what it does.

Check the website code to find 'alt' text for images and 'aria-label' for icons. Use tools like WAVE or Axe to find any missing text descriptions.

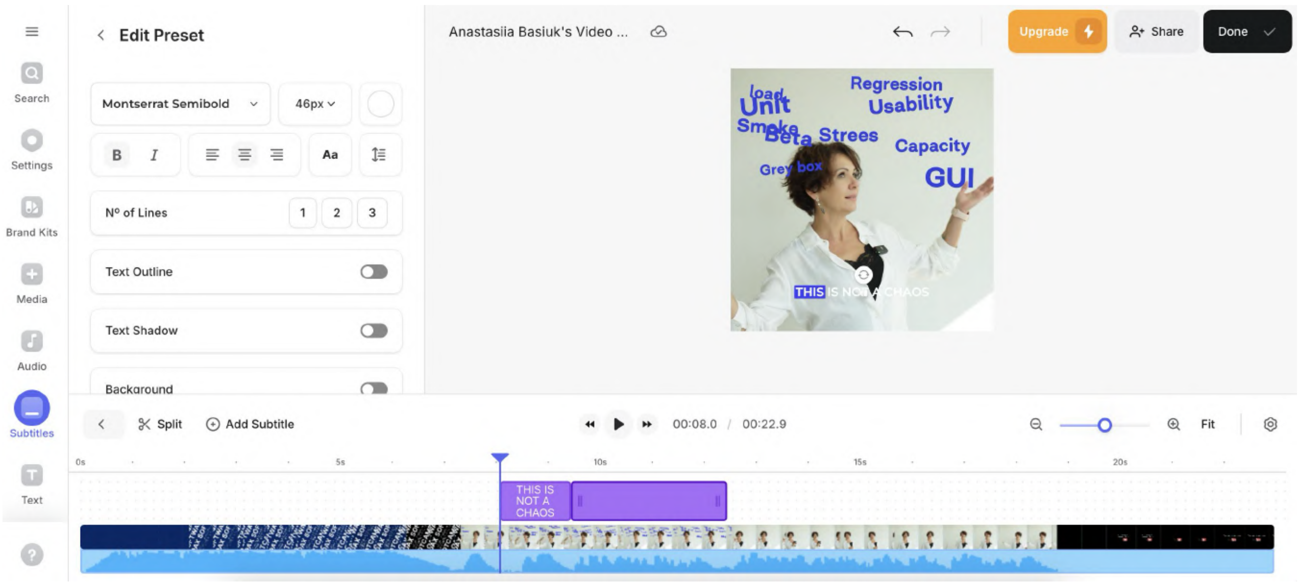
```
<button type="button" class="contact-button">
    <img src="submit.png" alt="Contact us" width="113" height="36" aria-hidden="true">
    <span class="visually-hidden">Contact us</span>
</button>
```

2. Captions and Transcripts for Video and Audio

Add captions to videos that have sound and written transcripts for audio-only content:

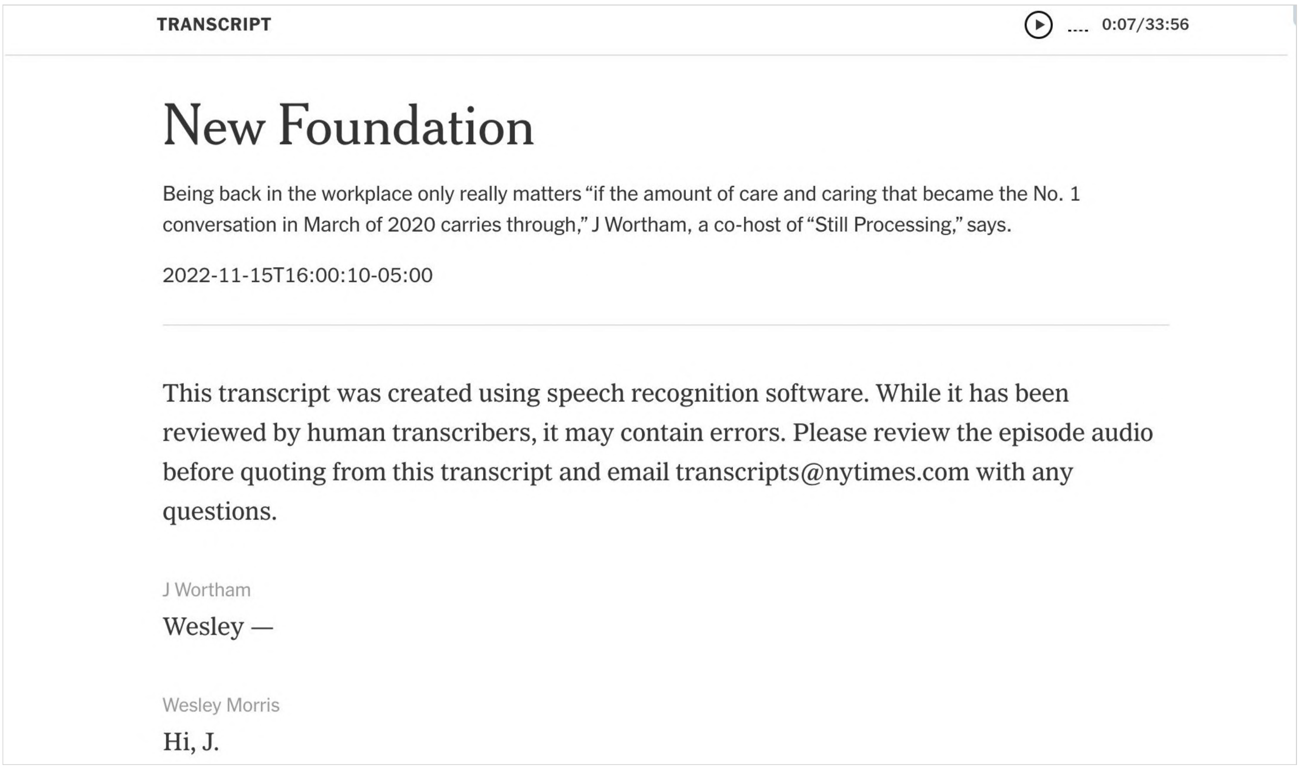
- **Video Captions:** For prerecorded videos, synchronized captions must match the spoken dialogue and relevant sounds (e.g., music, sound effects).

☑ Play each prerecorded video to make sure captions appear at the right time and accurately show all dialogue and key sounds.
☑ Use VEED to generate captions and evaluate video content for synchronized captions.



- **Audio Transcripts:** For audio-only content (podcasts, recordings), provide a written transcript both for the spoken content and any significant sounds.

☑ Check that the transcript is complete and includes all that's said and important sounds (e.g., laughter, background effects).
☑ Check for completeness of audio transcripts with AI tool Screenapp.io.



TRANSCRIPT

0:07/33:56

New Foundation

Being back in the workplace only really matters "if the amount of care and caring that became the No. 1 conversation in March of 2020 carries through," J Wortham, a co-host of "Still Processing," says.

2022-11-15T16:00:10-05:00

This transcript was created using speech recognition software. While it has been reviewed by human transcribers, it may contain errors. Please review the episode audio before quoting from this transcript and email transcripts@nytimes.com with any questions.

J Wortham

Wesley —

Wesley Morris

Hi, J.

Audio Transcript on the NY Times Website

- **Descriptions for Visuals:** Add audio descriptions for videos with important visual elements not mentioned in the audio. These descriptions can be part of the video or available separately.

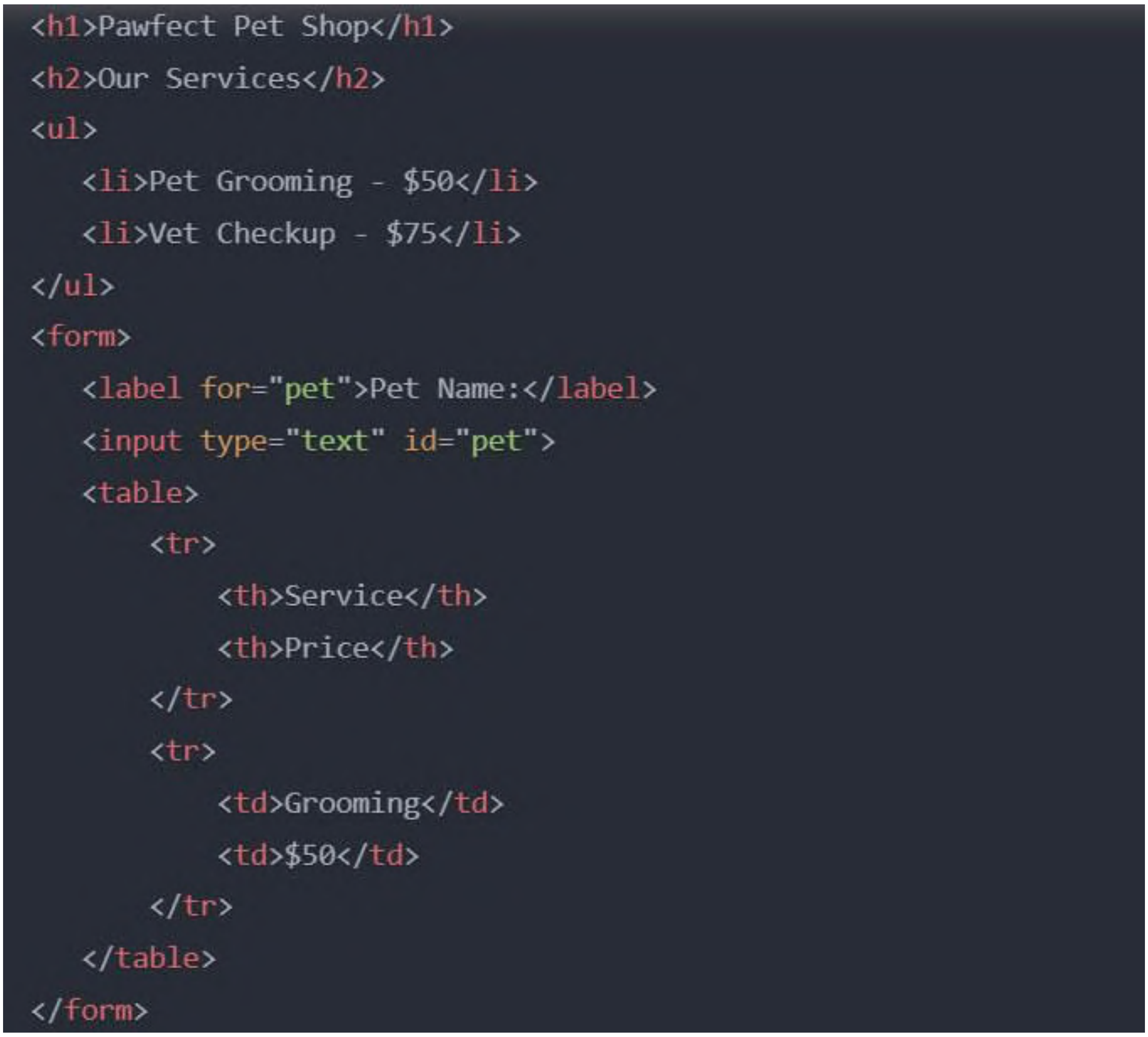
✓ Make sure the visual elements' descriptions are either included directly in the video or available as a separate option.

3. HTML Code Structure and Sequence

All webpage structure and information defined through HTML code must be organized in a way that assistive technologies can understand and interpret. This means using the right HTML elements in the right order to create a logical structure that maintains meaning regardless of how users access it.

- **Core Elements:** Use semantic HTML tags (<h1>, <h2>, <ul>, <ol>, <label>, <th>) to create proper structure and hierarchy.

✓ Check implementation by reviewing code and testing with accessibility tools.



```html
<h1>Pawfect Pet Shop</h1>
<h2>Our Services</h2>
<ul>
   <li>Pet Grooming - $50</li>
   <li>Vet Checkup - $75</li>
</ul>
<form>
   <label for="pet">Pet Name:</label>
   <input type="text" id="pet">
   <table>
       <tr>
           <th>Service</th>
           <th>Price</th>
       </tr>
       <tr>
           <td>Grooming</td>
           <td>$50</td>
       </tr>
   </table>
</form>
```

HTML for the pet shop website with the correct order of headers and lists elements, and the pricing table

- **Code Order:** Ensure HTML elements follow a logical sequence that matches the intended reading order.

☑ Use a screen reader to verify proper code structure and succession.
☑ Test the reading order of HTML code with ACHECKS, which can generate reports on the sequence of elements.

- **Code Testing:** Review that all relationships between elements are properly defined in HTML.

☑ Test content readability and meaning when CSS is disabled.

4. Content Access

All web page content must be accessible to users regardless of how they interact with it, whether through visual, audio, or assistive technologies. This means providing clear alternatives and maintaining meaningful structure across different ways of accessing content.

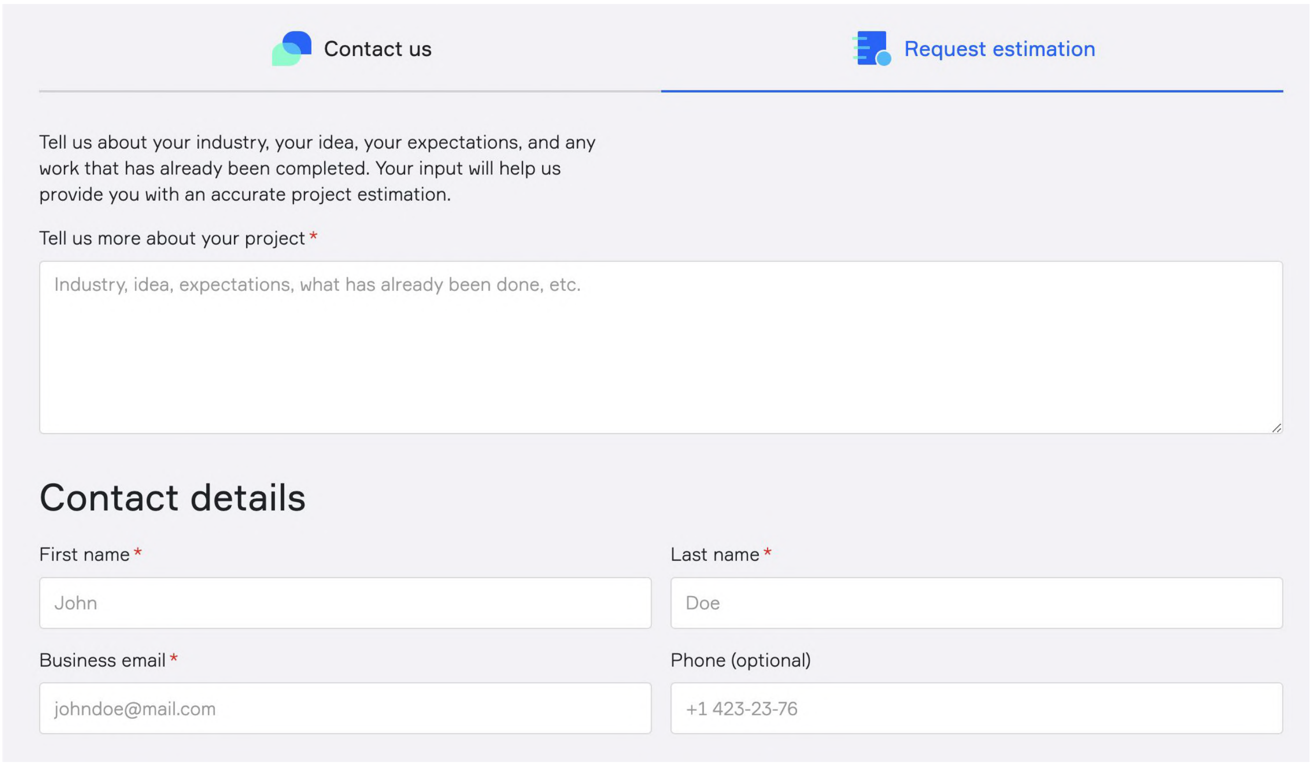
- **Reading Flow:** The order of content must be logical and consistent, as changing it could alter the meaning. Create a logical sequence that maintains meaning no matter what will be used for navigation.

✓ Test keyboard navigation to ensure users can tab through interactive elements in a way that makes sense contextually.

✓ Use Accessibility Cloud Lite or Axe-core NodeJS APIs to test tab order and interactive elements to verify that keyboard navigation follows a logical flow and is accessible without a mouse.

- **Clear Indicators:** Include multiple ways to convey important information.

✓ For example, mark required fields with both color and text indicators like "Required" or an asterisk (*).



Tell us about your industry, your idea, your expectations, and any work that has already been completed. Your input will help us provide you with an accurate project estimation.

Tell us more about your project *

Industry, idea, expectations, what has already been done, etc.

Contact details

First name *	Last name *
John	Doe
Business email *	Phone (optional)
johndoe@mail.com	+1 423-23-76

- **Non-Reliance on Sensory Cues:** Instructions should be clear and easy to understand without depending only on things like color, shape, or sound. For example, instead of saying "click the green button," it should say "click the 'Submit' button."

- Review instructions across your website to make sure they don't rely solely on sensory characteristics. Check for phrases that mention color, shape, or location without also providing a description in words.
- Use AudioEye Color Contrast Checker to verify that color-based indicators also include text cues (like "Required" or "*") and meet WCAG contrast requirements.

5. Inclusivity Characteristics

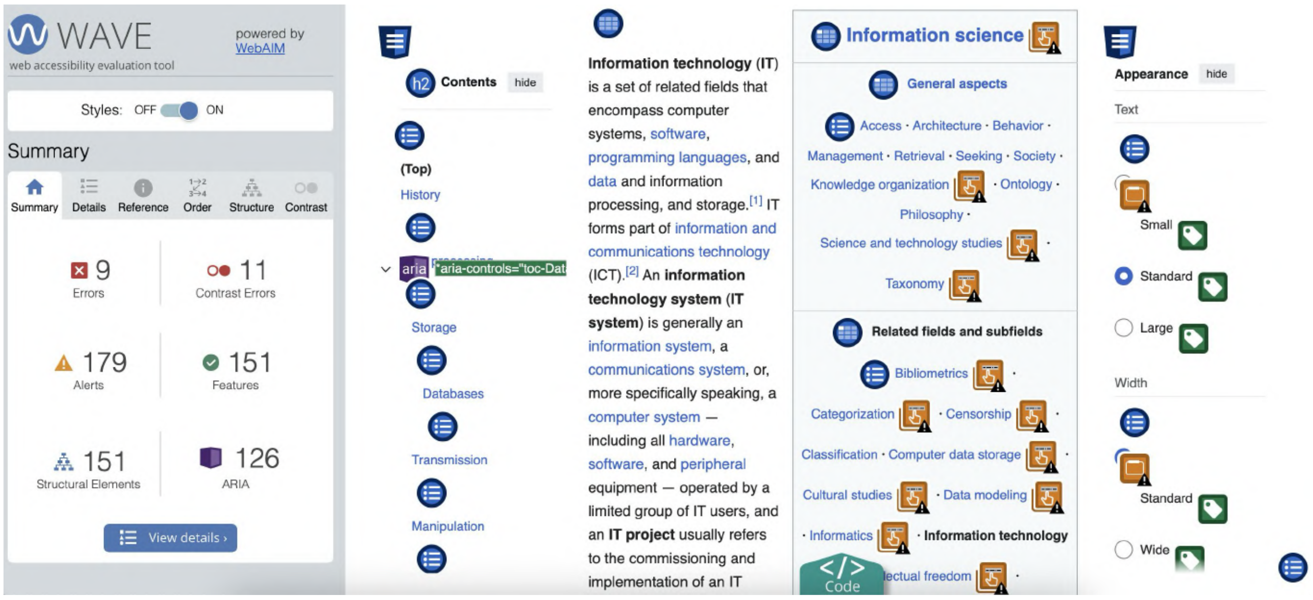
Instructions and information on web pages must be understandable without relying on how something looks, sounds, or where it's placed. This ensures users with visual impairments, color blindness, or cognitive disabilities can fully understand and use the content through various assistive technologies.

- **Multiple Indicators:** Use more than one way to convey important information and instructions.

- Test that all content makes sense without visual or audio cues. Verify text descriptions with tools like WAVE or screen readers.

- Alternative Instructions: Include text-based directions instead of relying on visual or audio cues.

- Ensure form controls have visible labels, not just placeholder text.


WAVE
web accessibility evaluation tool
powered by WebAIM
Styles: OFF ON
Summary
Summary Details Reference Order Structure Contrast
View details ›
Contents hide
(Top)
History
aria "aria-controls="toc-Dat
Storage
Databases
Transmission
Manipulation
Information science
General aspects
Access · Architecture · Behavior · Management · Retrieval · Seeking · Society · Knowledge organization · Ontology · Philosophy · Science and technology studies · Taxonomy
Related fields and subfields
Bibliometrics · Categorization · Censorship · Classification · Computer data storage · Cultural studies · Data modeling · Informatics · Information technology · lectual freedom ·
Code
Appearance hide
Text
Small
Standard
Large
Width
Standard
Wide

9 Errors
11 Contrast Errors
179 Alerts
151 Features
151 Structural Elements
126 ARIA

Information technology (**IT**) is a set of related fields that encompass computer systems, software, programming languages, and data and information processing, and storage.[1] IT forms part of information and communications technology (ICT).[2] An **information technology system** (**IT system**) is generally an information system, a communications system, or, more specifically speaking, a computer system — including all hardware, software, and peripheral equipment — operated by a limited group of IT users, and an **IT project** usually refers to the commissioning and implementation of an IT

Example of using WAVE Evaluation Tool on Wikipedia page

PRINCIPLE 2

Make UI Components and Navigation Easy to Operate

1. Enabling All Users to Operate With a Keyboard Only

This guideline makes sure that everything on a website can be accessed and operated using only a keyboard, without needing specific timing or input methods. This is crucial for users who can't use a mouse due to physical limitations, vision problems, or personal preferences. The guideline requires that all interactive elements, like links, buttons, and form fields, must be accessible through keyboard commands.

This guideline supports users with disabilities, older adults, and anyone who prefers keyboard navigation, ensuring equal access. It also improves the overall usability of the website for all users, no matter their input preferences or abilities.

- **Keyboard-Friendly Features:** Everything on the website must work with a keyboard, so users don't have to rely on a mouse. This rule helps people with disabilities, older people, and anyone who likes using a keyboard instead of a mouse to access websites.

- ✓ Verify that all website functions and interactive parts can be accessed and used with just a keyboard (e.g., using Tab, Enter, and other keyboard keys).
- ✓ Confirm that buttons, links, form fields, and other interactive items can be activated and controlled using keyboard commands.
- ✓ Use the keyboard (Tab, Shift + Tab, Enter, Escape) to go through all interactive elements and ensure there are no issues.
- ✓ Check that the focus moves properly between elements and users can exit any interactive part without getting stuck.

- **No Specific Timing:** Users shouldn't have to rush or meet any time limits when using the keyboard to move around.

- ✓ Make sure users don't have to do anything within specific time limits when navigating with the keyboard.
- ✓ Confirm users can see all visible changes in the interactive elements as they move through the website using their keyboard.

- **Clear Visual Focus:** Ensure the current keyboard focus is clearly visible as users navigate, so they can track where they are.

- ✓ Verify the focus indicator changes visibly as the user moves between elements.
- ✓ Check that all form fields are accessible and users can easily navigate away from them using the keyboard.
- ✓ Make sure modal dialogs don't trap the focus inside them and allow users to close them using the Escape key or by clicking outside.
- ✓ Utilize Accessibility Cloud Lite to confirm that focus indicators are clearly visible and move logically between elements

- **Exceptions and Alternatives:** Find any parts that need mouse movement and make sure there are other ways to use these features.

- ✓ Identify any components that may need specialized input (like freehand drawing) and provide alternative ways to access similar features.
- ✓ For any inherent keyboard traps in specialized content, offer alternative navigation options to maintain accessibility.

2. Making Keyboard Navigation Easier With Shortcuts

If a website uses keyboard shortcuts that only involve regular keys like letters, numbers, or punctuation, then at least one of these things must be true:

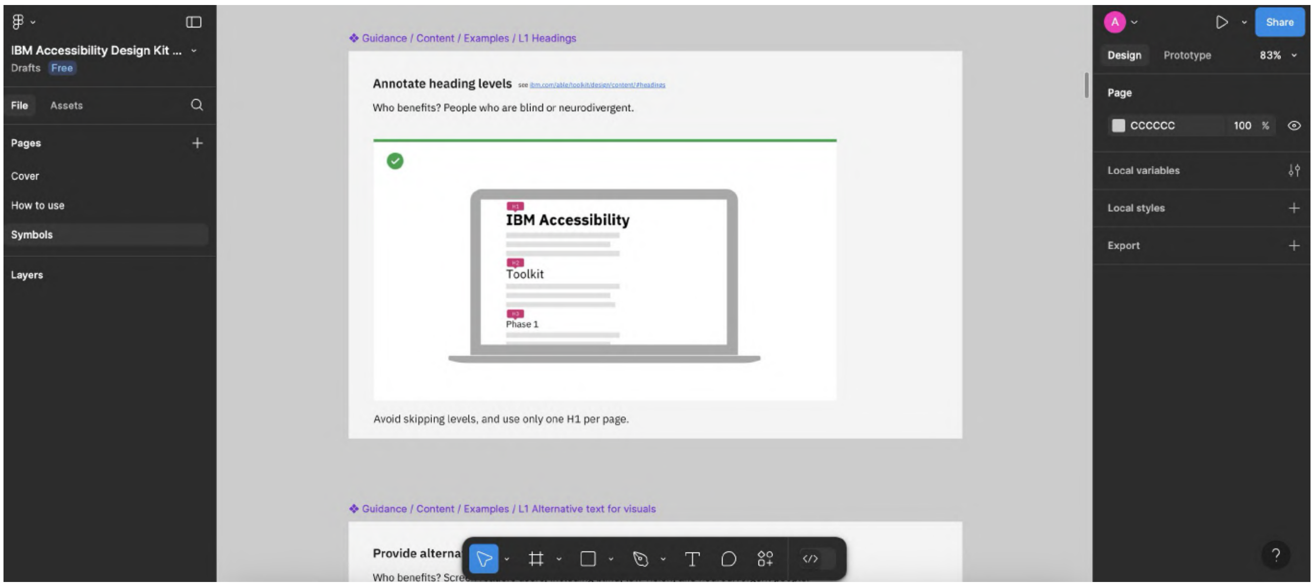
- **Users can turn the shortcut off completely.**
- **Users can change the shortcut to include special keys like Ctrl or Alt.**
- **The shortcut only works when the specific part of the page using it is focused.**

This helps prevent accidental activation of shortcuts, which can be really frustrating for users who rely on speech input or have trouble with dexterity. It makes the keyboard shortcuts more usable and accessible for everyone. This helps users with speech input and those who may accidentally hit keys while navigating with a keyboard.

- **Turn Off Shortcuts:** Users must be able to disable character key shortcuts to prevent unintended actions.

- ✓ Verify that all character key shortcuts can be turned off through user settings or preferences.

- **Remap Shortcuts:** Users should be able to change character key shortcuts to include special keys (like Ctrl, Alt) to reduce accidental triggers.



Annotate heading levels

Who benefits? People who are blind or neurodivergent.

Avoid skipping levels, and use only one H1 per page.

Provide alterna

Who benefits? Scre

IBM Accessibility Design Kit for Figma

- Ensure that users can change character key shortcuts to include non-printable keys and that these changes are saved.
- Ensure instructions for disabling and remapping shortcuts are clearly documented and accessible to users.
- Use IBM Equal Access cypress-accessibility-checker in a testing pipeline to verify that shortcuts can be customized with additional modifier keys like Ctrl or Alt and that they activate only when focused on relevant elements.

- **Active Only on Focus:** Shortcuts should only work when the relevant user interface component is focused, so they don't interfere with other input methods.

- Confirm character key shortcuts are only active when the relevant component is focused, to prevent unintended actions.

3. Giving Your Users Enough Time

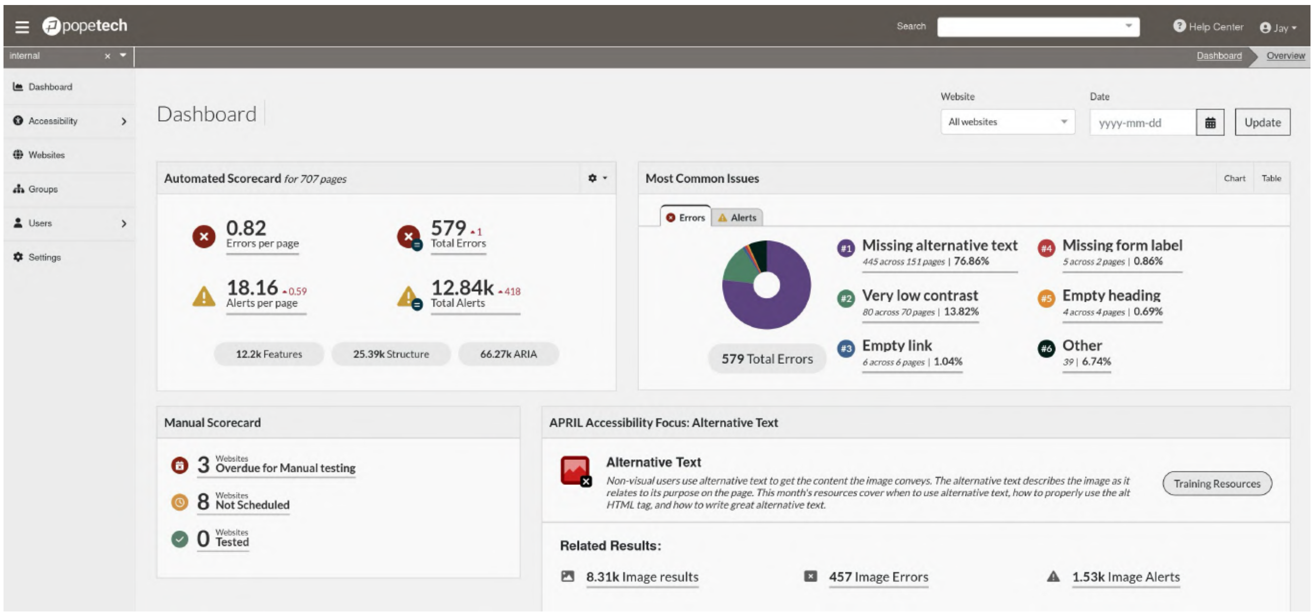
This guideline requires that if a website has time limits, users must be able to turn off, adjust, or extend those limits. This is essential for users who need extra time due to disabilities, ensuring they can complete tasks without time pressure. This flexibility helps users with various disabilities complete tasks without undue time pressure, creating a more inclusive experience.

- **Turn Off or Adjust Time Limits:** Users must have the ability to completely turn off any time limits before they start using the content. Also, users should be able to adjust the time limit to at least 10 times the default setting, giving them enough time to complete tasks.

- ☑ Verify that users can easily find and use controls to turn off or adjust time limits.
- ☑ Ensure that users receive a warning before the time limit expires and can extend it with a simple action, like pressing a button.
- ☑ Check that any adjustable time limits allow users to set them to at least 10 times the default duration.

- **Exceptions for Essential Timing:** Time limits that are essential for real-time events (like auctions) or longer than 20 hours are exempt from these requirements.

- ☑ Identify any exceptions and ensure they are clearly communicated in the documentation.
- ☑ Use Pope Tech to test that warnings are visible and actionable before time expiration, allowing users to easily extend time without rushing.



Alternative Text

Non-visual users use alternative text to get the content the image conveys. The alternative text describes the image as it relates to its purpose on the page. This month's resources cover when to use alternative text, how to properly use the alt HTML tag, and how to write great alternative text.

Pope Tech

4. Controlling Automatically Moving or Updating Content

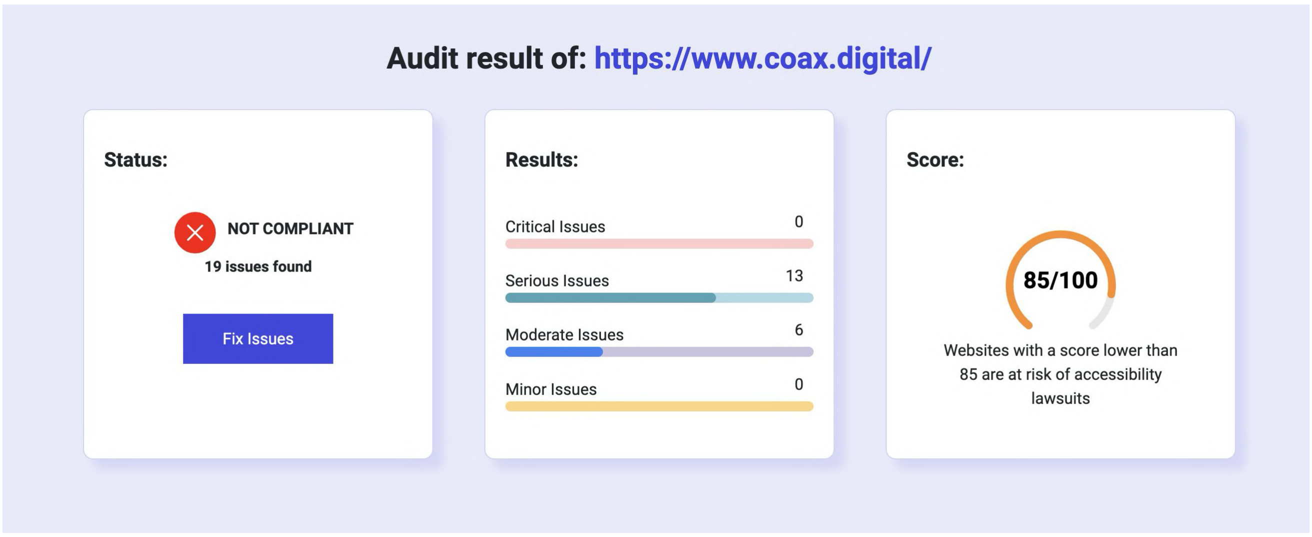
This guideline requires that users have control over content that moves, blinks, scrolls, or updates automatically. This is crucial for users who may find such motion distracting or disorienting. This allows users to consume content comfortably without distractions, especially those with attention issues or visual impairments.

- **Control Over Motion:** Users must have a mechanism to pause, stop, or hide automatically moving, blinking, scrolling, or auto-updating content that lasts more than 5 seconds.

- Verify that all moving or auto-updating content includes clearly labeled controls to pause, stop, or hide it.
- Ensure the controls are clearly labeled and easily accessible to all users. Confirm the controls are keyboard-operable and accessible.

- **Moving Content Timing and Exceptions:** Automatically moving content must last more than 5 seconds before requiring user intervention to pause or stop it.

- If the movement or blinking is part of an essential activity (like a loading indicator), the requirement for user controls may not apply.
- Identify any essential animations and document them as exceptions.
- For essential animations, use AEL Accessibility Checker to document exceptions clearly and verify that essential content is exempt only when necessary.



Audit result of: https://www.coax.digital/

Status:

NOT COMPLIANT

19 issues found

Results:

Critical Issues	0
Serious Issues	13
Moderate Issues	6
Minor Issues	0

Score:

85/100

Websites with a score lower than 85 are at risk of accessibility lawsuits

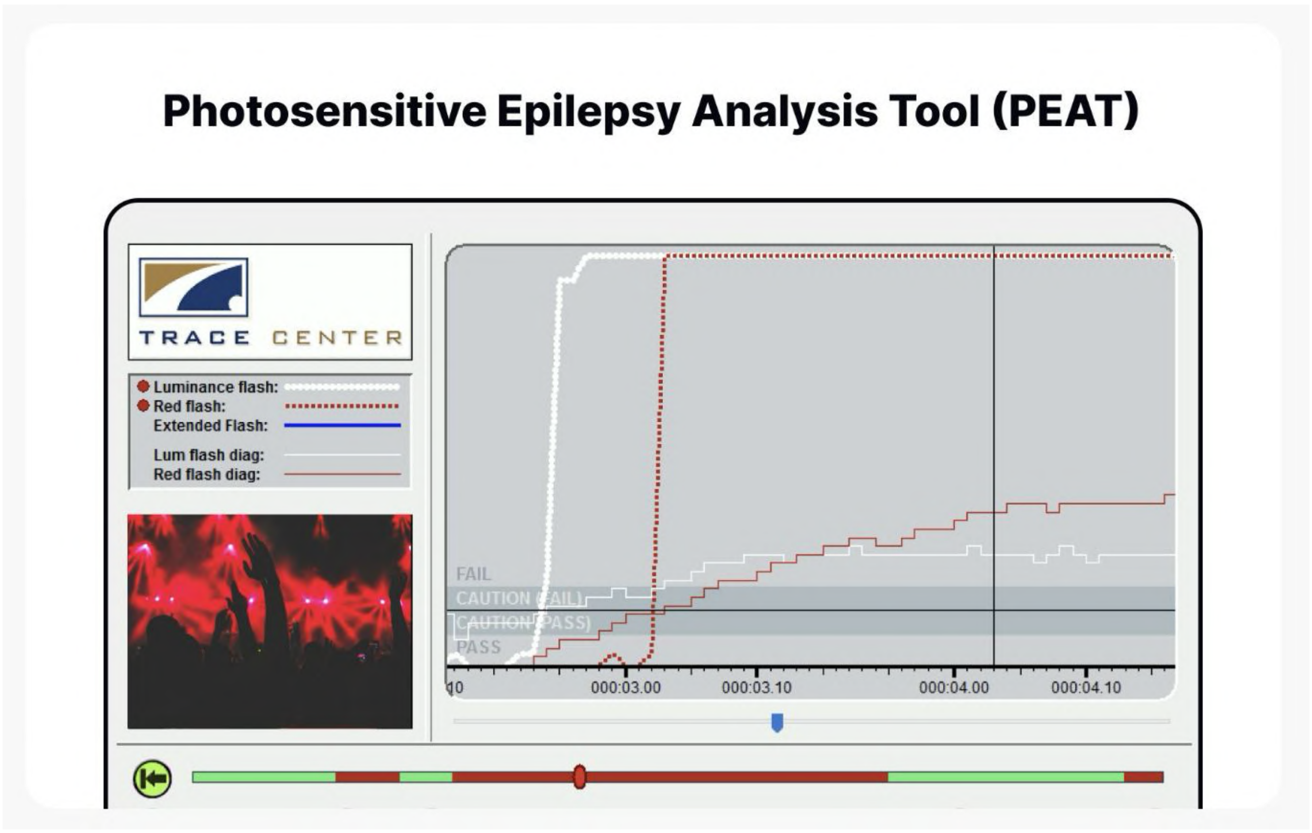
Audit Result from AEL Accessibility Checker

5. Avoiding Seizure-Triggering Flashes

The Three Flashes or Below Threshold guideline is designed to protect users from web content that could potentially cause seizures due to flashing or blinking elements. It requires that web pages do not contain anything that flashes more than three times in any one second, or the flashing must be below specific thresholds.

- **Flashing Content Limitations:** Ensure that animations, videos, and interactive elements are designed to avoid rapid flashing or blinking.

- Web content must not flash more than three times in any one-second period to prevent triggering seizures.
- Use tools to analyze the visual content and ensure no element flashes more than three times per second.
- Utilize PEAT (Photosensitive Epilepsy Analysis Tool) to assess whether flashing content meets the required thresholds.



Photosensitive Epilepsy Analysis Tool

- **Safe Flash Thresholds:** If flashing does occur, it must be below specified thresholds to minimize risk. If flashing is essential for communication or functionality, warnings, and alternatives should be provided to users who might be affected (Note that this exception is not permitted for the more stringent 2.3.2 Three Flashes Level AAA).

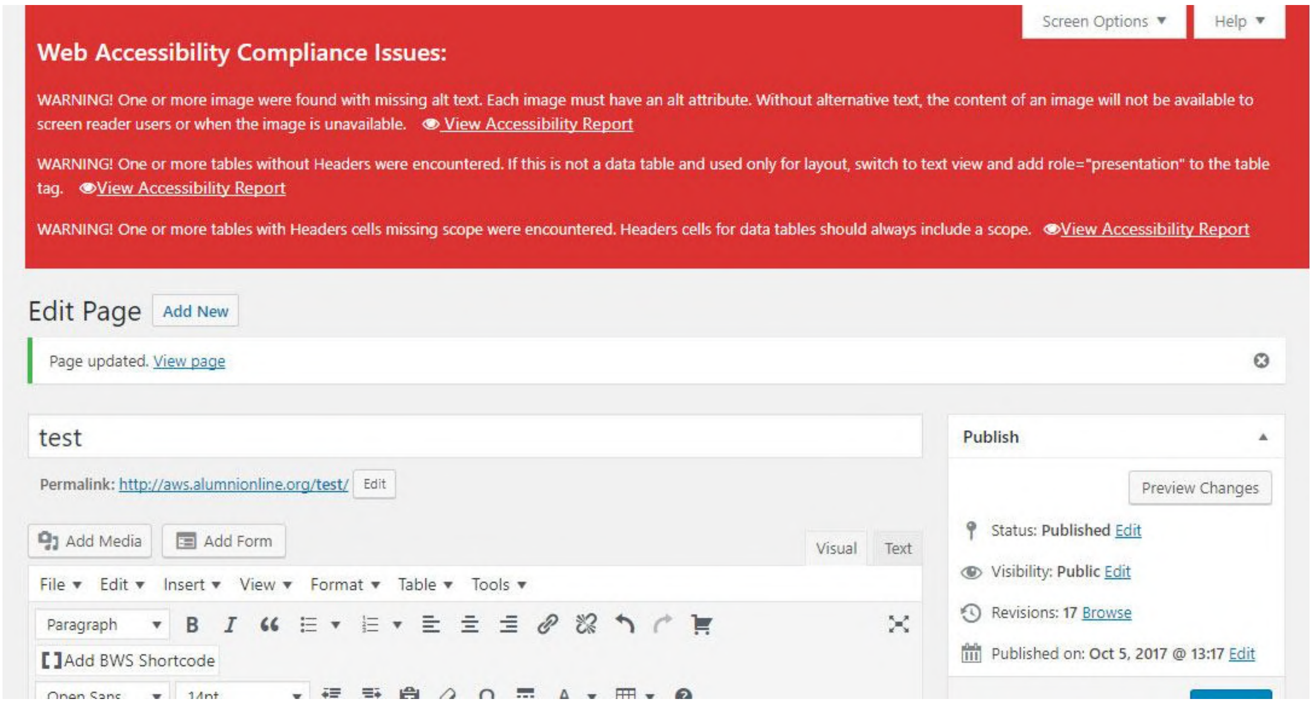
- ✓ Check if the flashing area small and dimmed.
- ✓ Check that the combined area of flashes occurring concurrently occupies no more than a total of 006 steradians within any 10-degree visual field.
- ✓ Identify any instances where flashing content is essential and ensure these cases are well documented with appropriate warnings.

6. Providing Efficient Navigation

- **Bypass Blocks:** Allow users to bypass repetitive elements, reducing the time and effort needed to reach desired content. This criterion supports users with disabilities — especially those using screen readers or keyboard navigation — by minimizing repetitive interactions.

- ✓ Verify that each page includes a clearly labeled link (e.g., 'Skip to Content') that allows users to bypass repeated content.
- ✓ Ensure that the bypass link is visible when focused and can be easily accessed using keyboard navigation.
- ✓ Confirm that clicking the bypass link successfully directs users to the main content area without any errors.

- Use WP ADA Compliance Check Plugin to ensure a "Skip to Content" link is accessible and functional.


Web Accessibility Compliance Issues:

WARNING! One or more image were found with missing alt text. Each image must have an alt attribute. Without alternative text, the content of an image will not be available to screen reader users or when the image is unavailable. View Accessibility Report

WARNING! One or more tables without Headers were encountered. If this is not a data table and used only for layout, switch to text view and add role="presentation" to the table tag. View Accessibility Report

WARNING! One or more tables with Headers cells missing scope were encountered. Headers cells for data tables should always include a scope. View Accessibility Report

Edit Page

Page updated. View page

test

Permalink: http://aws.alumnionline.org/test/

Status: Published Edit

Visibility: Public Edit

Revisions: 17 Browse

Published on: Oct 5, 2017 @ 13:17 Edit

WP ADA Compliance Check Plugin

- **Exception:** Minimalist websites or those without repetitive blocks may not require a bypass mechanism, as there may be no need for such functionality.

- Identify any sections of the website where content is repeated across multiple pages and ensure appropriate bypass mechanisms are in place.

- **Page Titles:** Each web page must have a unique and descriptive title that clearly indicates the topic or purpose of the page, aiding user understanding and navigation, particularly for users with screen readers or assistive technologies.

- Ensure that each web page includes a <title> element in the HTML <head> section that accurately describes the page's purpose.
- Avoid using identical titles for different pages; create variations that reflect the specific content of each page.
- Verify that screen readers can correctly announce the page title and that it is meaningful in the context of the content.
- Verify page titles with Silktide accessibility checker to confirm unique and descriptive titles.

- **Exception:** In single-page applications (SPAs) or dynamic content scenarios, titles should be updated dynamically to reflect the current view or context.

- ✓ For such, implement methods to update the page title based on content changes to maintain context for users.

7. Focus and Link Accessibility

The focus order and linking should reflect the visual layout and logical relationships of the content, allowing users to navigate through elements in a way that makes sense. Efficient link usage ensures the users know exactly where each link is taking them.

- **Understandable Focus Order:** A consistent focus order helps users form a mental model of the content, reducing confusion and enhancing usability for keyboard users and those with disabilities.

- ✓ Navigate through the website using the keyboard (Tab key to move forward, Shift + Tab to move backward) to ensure the focus order follows a logical sequence.
- ✓ Verify that there is a clear visual indication of which element is currently focused (e.g., border highlight or background color change).
- ✓ Examine the arrangement of interactive elements (links, buttons, form fields) to confirm they follow a logical order consistent with their visual presentation.

- **Exception:** If the navigation does not affect meaning or operation, different focus orders may be acceptable; however, care should be taken to ensure clarity for all users.

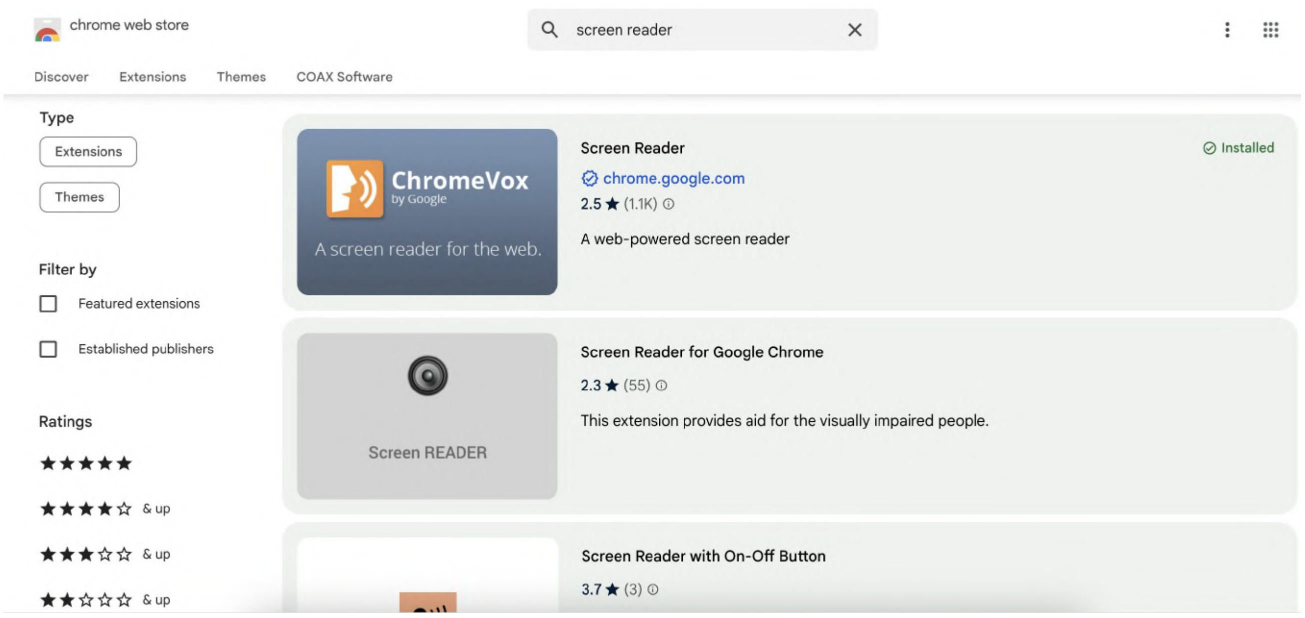
- ✓ Use Accessibility Insights for Web to confirm the focus order follows a logical sequence, especially in dynamic content updates (e.g., AJAX), and check that the focus state is clearly visible across elements.

- **Link Purpose (in context):** Each link on a webpage should have clear and descriptive text that conveys its purpose. This criterion aids users with cognitive disabilities and those using assistive technologies by reducing confusion and ensuring they can navigate effectively.

- ✓ Review all links on the webpage to ensure the text clearly indicates the purpose of each link without needing additional context.
- ✓ Examine surrounding content to confirm that it provides meaningful context for links where the text alone may not suffice.
- ✓ Test the page with screen readers to ensure users can easily understand the purpose of each link based on the text and context provided.

- **Exception:** Links designed to be ambiguous for all users (e.g., in games) may not require clear purposes as their intent is to create suspense.

- ☑ Identify any links that may be intentionally ambiguous and ensure they are appropriate for their context.
- ☑ Test with screen readers to ensure each link's purpose is clear in isolation and that contextual cues around ambiguous links (e.g., "click here") are understandable for users with cognitive or vision impairments.



Screen Reader
chrome.google.com
2.5 ★ (1.1K)
A web-powered screen reader
Installed

Screen Reader for Google Chrome
2.3 ★ (55)
This extension provides aid for the visually impaired people.

Screen Reader with On-Off Button
3.7 ★ (3)

Screen reader extensions available in Chrome

8. Complex Interaction Requirements

All features that use multi-finger gestures, specific movement patterns, or device motion must provide simple alternatives unless the complex interaction is absolutely necessary. This means ensuring accessibility through basic controls and input methods.

- **Multi-Finger Gestures:** Features requiring two or more fingers must have single-pointer alternatives.

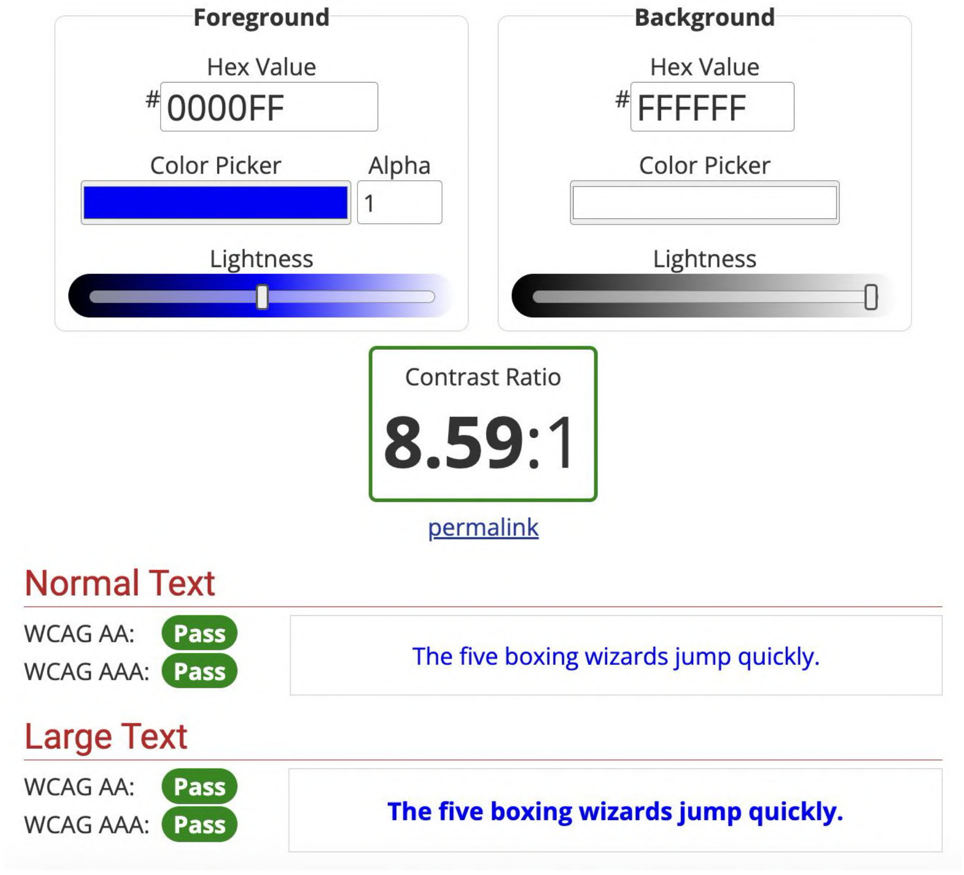
- ☑ Test pinch-zoom features to verify presence of button controls (+/-).

- **Movement Patterns:** Actions requiring specific movements must offer simple alternatives.

- ☑ Review swipe interactions to confirm presence of button controls (arrows).

- **Device Motion:** Features controlled by device movement must have standard controls.

- ☑ Verify motion features can be disabled through settings and have button alternatives.
- ☑ Use WebAIM's accessibility checker to ensure motion-based features (e.g., device tilting) are accessible via button controls and can be turned off through settings.


Foreground

Hex Value

#0000FF

Color Picker — Alpha: 1

Lightness

Background

Hex Value

#FFFFFF

Color Picker

Lightness

Contrast Ratio

8.59:1

permalink

Normal Text

WCAG AA: Pass

WCAG AAA: Pass

The five boxing wizards jump quickly.

Large Text

WCAG AA: Pass

WCAG AAA: Pass

The five boxing wizards jump quickly.

WebAIM's accessibility checker

- **Exceptions:** Only truly necessary gestures (like signatures) or motions (like pedometers) may omit alternatives.

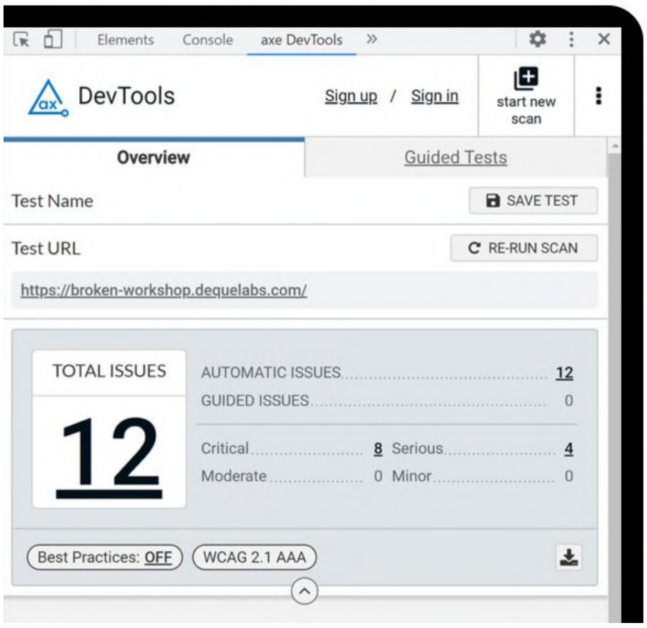
- ✓ Validate that exceptions are genuinely required for functionality.
- ✓ Test functions across phones, tablets, and computers for consistency.
- ✓ Test all alternatives work with eye-tracking and voice control tools.
- ✓ Review motion-controlled features across various phones and tablets.

9. Accessibility Labeling and Interaction Requirements

All interactive elements must have proper activation behavior and matching visible and programmatic labels. This means implementing proper event handling and ensuring consistency between visual and assistive technology output.

- **Activation Timing:** Actions should trigger on pointer up, not pointer down, unless immediate activation is essential.

- ✓ Test all interactive elements to verify proper activation timing.
- ✓ Test interactive elements with Axe DevTools to confirm actions only trigger on pointer-up events unless immediate activation is essential.



Elements Console axe DevTools

DevTools Sign up / Sign in start new scan

Overview | Guided Tests

Test Name SAVE TEST

Test URL RE-RUN SCAN

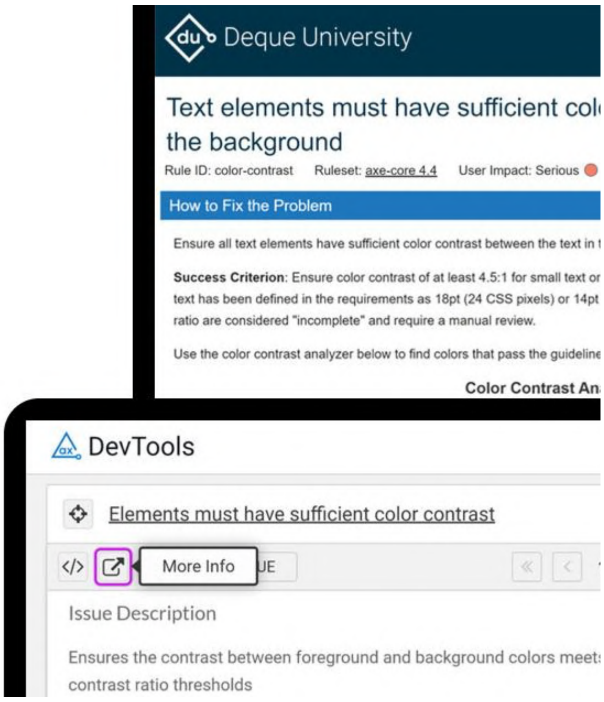
https://broken-workshop.dequelabs.com/

TOTAL ISSUES 12

AUTOMATIC ISSUES 12
GUIDED ISSUES 0

Critical 8 Serious 4
Moderate 0 Minor 0

Best Practices: OFF WCAG 2.1 AAA



Deque University

Text elements must have sufficient col
the background

Rule ID: color-contrast Ruleset: axe-core 4.4 User Impact: Serious

How to Fix the Problem

Ensure all text elements have sufficient color contrast between the text in t

Success Criterion: Ensure color contrast of at least 4.5:1 for small text or
text has been defined in the requirements as 18pt (24 CSS pixels) or 14pt
ratio are considered "incomplete" and require a manual review.

Use the color contrast analyzer below to find colors that pass the guideline

Color Contrast An

DevTools

Elements must have sufficient color contrast

More Info

Issue Description

Ensures the contrast between foreground and background colors meet
contrast ratio thresholds

- **Cancellation Options:** Users must be able to prevent or reverse accidental activations.

☑ Verify pointer movement away cancels pending actions.
☑ Ensure actions can be reversed after completion.

- **Label Matching:** Visible text must match programmatic labels used by assistive technologies.

☑ Review all aria-label, aria-labelledby, and alt attributes against visible text.
☑ Test with screen readers to verify proper label announcement.

- **Consistent Behavior:** All interactions and labels must work similarly across devices and tools.

☑ Verify speech recognition responds to visible label text.
☑ Test all functionality across different browsers and devices.

- **Exceptions:** Immediate activation (like piano keys) must be clearly necessary.

☑ For press-down actions, verify release provides appropriate undo functionality.
☑ If pressing opens a menu, releasing should close it consistently across all devices.

PRINCIPLE 3

Make Sure Web Content is Presented in a Consistent and Predictable Way

1. Readable Content

Every webpage needs to tell computers what language it's written in. It's important because it helps assistive tools (like screen readers) to read text properly, correctly pronounce, and handle special characters correctly.

- **Right Language Tag:** The main language of each webpage must be marked in its code using a special tag called 'lang'.

✓ In the page code, find the HTML code that starts the webpage and make sure it includes the language tag. It should look like this: <html lang="en"> for English and <html lang="fr">' for French (for instance).

✓ Confirm that the language code used is correct according to ISO standards (e.g., en-US for American English, fr-FR for French).

- **Multiple Languages:** On pages with multiple languages, you should set the main language first., then mark any other languages used on the page separately.

✓ If your page uses different languages in places, mark these sections with their own language tags Example: Use '<span lang="es">Hola</span>' for Spanish words.

✓ Test with NVDA screen reader to confirm that marked language changes are recognized in assistive technologies, especially in multilingual pages where specific words or phrases are marked in different languages.

2. No Unexpected Change by Focus

Focus should not trigger context change. When someone moves to a clickable item on a website (such as buttons, links, or form fields), the context should not change automatically. The website shouldn't submit forms, open new windows, or change what's shown until the user actually clicks or presses a key. This helps people know what to expect and stay in control, which is especially helpful for people with vision, thinking, or movement challenges using keyboard navigation or special tools to browse.

- **Automatic Form Sending:** Forms should not send automatically when you move to a field.

- ☑ Make sure important actions (like sending forms or opening modals) only happen when users actually do something - moving to an item shouldn't trigger any actions on its own.
- ☑ Use Silktide accessibility checker to ensure no forms auto-submit.

- **Windows Opening:** Windows should not be opening just by moving to a link.

- ☑ Make sure changes only happen when users take action (like clicking or pressing Enter).

- **Focus Jumping to Different Places Unexpectedly:** Help boxes or hints should not be popping up uninvited.

- ☑ Use the Tab key to move through all clickable items and make sure nothing changes unexpectedly when moving to different items.
- ☑ Open the webpage and watch for any new windows or pop-ups that open by themselves, and check that these don't steal attention from the main page.
- ☑ Check that no pop-ups appear just because you moved to something and make sure the focus doesn't jump to different places on its own.

3. No Unexpected Change by Fill-In Actions

Changes in form fields shouldn't trigger automatic changes. When users fill out forms or change settings (like checkboxes, dropdowns, or text fields), nothing big should change automatically unless they're told about it first. This helps everyone, especially people with cognitive challenges or those using assistive tools, know what to expect when using a website.

- **No Automatic Changes Without User Action:** Forms and settings should not cause changes unless users take a clear action, like clicking a submit button. Changes should be intentional, not automatic.

- ☑ Check all form fields to verify that no actions occur unexpectedly. Complete the form without it auto-submitting, and ensure dropdowns don’t trigger page navigation on selection.
- ☑ Use Accessibility Insights for Web to confirm that form fields and settings don’t cause automatic page changes or submit forms without clear user action.

- **Warn Users About Any Automatic Changes:** If an automatic change is expected, users must be notified ahead of time to prevent surprises.

- ☑ Review each form field and setting for instructions about any automatic actions, ensuring they are clear and visible to users.

- ✓ Use Axe DevTools to verify that any elements with automatic behaviors (if present) have accompanying text warnings.

- **Avoid Unnecessary Focus Jumps:** Tabbing through form fields should follow a logical, predictable order. Focus should not jump between fields unexpectedly unless absolutely necessary and clearly explained.

  - ✓ Use the Tab key to navigate through all interactive elements, verifying that focus remains stable and predictable.

- **No Surprises from Sliders, Toggles, or Options:** Adjusting sliders, toggles, or selecting options (like checkboxes and radio buttons) should not trigger unexpected changes.

  - ✓ Change each slider, toggle, checkbox, and radio button option to ensure no automatic actions or visual changes happen without notice.

- **Provide Clear Instructions for Expected Behaviors:** When specific actions or changes will occur based on user input, make sure instructions are provided to inform users of the expected outcomes.

  - ✓ Verify that all fields or controls with potential changes include clear instructions explaining what will happen when users make adjustments.
  - ✓ Use WAVE to check that all fields and interactive controls contain help text or tooltips explaining what users can expect when they interact.

- **Exceptions Only for Expected or Pre-Informed Changes:** Automatic changes are acceptable if they are either a normal part of the feature or users are clearly informed in advance.

  - ✓ Identify all features that may include automatic changes, confirming they align with expected behaviors and that users are adequately informed.

4. Helping Understand What to Input

Input errors must be clearly identified and explained. When the website finds a mistake in what users type or select, it must tell them exactly what's wrong and how to fix it in clear text. This helps everyone, especially people using screen readers or those with vision or cognitive challenges.

- **Automatic Error Detection and Specific Feedback:** The website should automatically identify common errors, such as incorrect email formats, and give specific feedback.

Each error message should point directly to the field needing correction and be clear, such as “please enter a valid email address” or “please enter a phone number with 10 digits.”

✓ Submit forms with incorrect or missing information. Confirm that error messages appear next to each relevant field, provide specific instructions on how to fix the issue, and are accessible to screen readers. Check that messages can be understood without relying solely on color.

- **Error Message Placement and Accessibility:** Error messages should be located near the field where the error occurred and should use visual markers, like color, alongside text explanations.

✓ Ensure that each error message is both next to the corresponding field and easily readable.

✓ Confirm that messages are visible for users who rely on screen readers or cannot see color, and verify that a summary of all errors appears at the top of the form.

- **Clear Labels and Instructions for Input Fields:** Each input field should have a clear, descriptive label, such as “email address” instead of just “email,” to help guide users, especially those with cognitive or visual challenges.

✓ Check that every input field has a clear label that tells users what to write.

✓ Use Axe DevTools or WAVE to verify that each field has a descriptive label connected with <label> tags and for attributes.

- **Provide Additional Help Text:** Include short, easy-to-understand help text as needed, such as examples or hints, to support user understanding of each field.

✓ Review all fields for additional help text where needed. Ensure that the help text is concise and effectively clarifies the expected input.

- **Indicate Required Fields Clearly:** Required fields should be clearly marked, typically with a Star (*) Or other indicator, along with text explaining the requirement.

✓ Confirm that users can easily distinguish required fields from optional ones and that indicators are accessible to screen readers.

Ensure Your Website is Robust and Works Well with Various Tools

1. HTML Structure and Accessibility Requirements

All HTML markup must follow specific syntactical rules and validation requirements to ensure reliable parsing by browsers and assistive technologies. This means implementing correct HTML structure and following standard coding practices that support accessibility.

- **Complete Structure:** Every HTML element must have proper opening and closing tags, with correct nesting order. Standard HTML elements must be used whenever possible instead of creating custom solutions.

✓ Use W3C Markup Validator to check for structural and syntactical errors.
✓ Test rendering across browsers to verify consistent element behavior.

- **Unique Attributes:** Each ID must be unique within the document, and elements cannot have duplicate attributes.

✓ Review code to verify ID uniqueness and proper attribute implementation.
✓ Use Accessibility Insights for Web to confirm unique ID assignments and proper attribute usage.

- **Code Validation:** HTML structure must be checked for technical correctness and proper implementation. The overall code structure must maintain consistency across different platforms.

✓ Use automated tools to verify syntax and structure compliance.
✓ Test across various browsers and devices to verify consistent behavior.

2. Interface Accessibility Requirements

All user interface components must provide clear programmatic information about their purpose, role, and state to ensure proper interpretation by assistive technologies. This means implementing proper attributes and properties that convey component functionality and support various access methods.

- **Component Names:** Every interactive element must have a programmatically-determined name that describes its purpose.

☑ Verify that each interactive component has a programmatically defined name using Axe DevTools and screen readers (e.g., JAWS, NVDA).

- **Role Definitions:** All components must have clearly defined roles that indicate their function.

☑ Validate role implementation using accessibility testing tools.

- **State Information:** Current states and values must be programmatically available to assistive technologies.

☑ Review all interactive components to verify state information is properly conveyed.

- **Interactive Testing:** All user interface elements must work properly with different input methods.

☑ Verify all interface elements function with various inputs, including keyboard navigation and speech recognition tools (e.g., Dragon NaturallySpeaking).

- **User Experience:** Content and functionality must remain meaningful regardless of access method.

☑ Review presentation and interaction across different access scenarios.