Deque Web Accessibility Checklist

Based on WCAG 2.1 AA

Background

Deque recommends using a combination of software tools and informed human analysis – using a checklist such as the one in this document – to test web accessibility.

Web Content Accessibility Guidelines

This checklist is based on the Web Content Accessibility Guidelines 2.1, W3C World Wide Web Consortium Recommendation 05/06/2018 (https://www.w3.org/TR/2018/REC-WCAG21-20180605/, Latest version at https://www.w3.org/TR/WCAG21/).

Accessibility Training

- Find detailed online accessibility courses and reference materials on Deque University.
- Experienced Deque web accessibility instructors are also available

Deque Accessibility Evaluation Software

axe DevTools is an automated and guided accessibility testing solution for Component Developers, Front-End Developers, Native Mobile App Developers and Test Engineers that allows you to easily find and fix 76-84% of accessibility errors before your applications get out of development. Solutions for Unit Testing, Test Automation, and integration with popular testing frameworks such as Selenium and QUnit, combined with in-browser testing using Deque's Attest extensions for Chrome and Firefox, allow you incorporate accessibility testing into your existing local development and testing environment.

axe Auditor is a manual accessibility testing solution for Accessibility Testing Specialists that allows you to optimize your accessibility testing, tracking, and reporting with step-by-step guidance and report building. Based on The Deque Way of accessibility testing, your QA team can run both manual and automated tests in the same interface, and without needing to be accessibility experts themselves, they can create precise, consistent accessibility issue reports for developers to resolve, then rerun tests on new builds and across multiple browsers to verify requirements have been met.

axe Monitor is an enterprise-level scanning and reporting tool that can evaluate the web accessibility of entire web sites, including password-protected areas. axe Monitor can implement use case scripts, allow custom reporting options based on WCAG or Section 508 guidelines, and allow scheduled monitoring of accessibility on projects of any scale or complexity. It is designed to work in conjunction with the axe DevTools Browser Extension for Chrome and Firefox.

axe, the Accessibility Engine, is Deque's free, open-source JavaScript accessibility rules library and browser extension. It was developed to be fast and to return zero false errors or duplicate results, and it is available as a GitHub repository, browser plugin, or framework integration.

Section 1: Structure and Semantics

General Notes About Semantic Markup:

If the technology used provides semantic structure to convey the relationships between information:

- Use semantic markup to mark emphasized or special text.
- Use text to convey information that is conveyed by variations of presentation of text.
- Separate information and structure from presentation. Do not use style to convey structure.

Page Title

Topic	Technique	WCAG AA
Markup	Correct markup: The page `<title>` MUST be present and MUST contain text.	Required WCAG 2.4.2
Meaningful Text	Accurate and informative: The page `<title>` MUST be accurate and informative.	Required WCAG 2.4.2
	Dynamic pages: The page `<title>` of dynamic pages (e.g. in single page apps) MUST be updated when the purpose of the page changes.	Required WCAG 2.4.2
	User Actions: If a page is the result of a user action or scripted change of context, the text of the `<title>` SHOULD describe the result or change of context to the user.	Best practice
	Concise: The `<title>` SHOULD be concise.	Best practice
	Unique: The page `<title>` SHOULD be unique, if possible.	Best practice
	Unique info first: Unique information SHOULD come first in the `<title>`.	Best practice
	Match heading: The page `<title>` SHOULD match (or be very similar to) the top heading (ideally marked as `<h1>`) in the main content.	Best practice

Page Language

Topic	Technique	WCAG AA
Page Language	Page Language: The primary language of the page MUST be identified accurately, using a standard language code, on the `<html>` element (e.g. `<html lang="en">` or `<html lang="fr">`).	Required WCAG 3.1.1
Language of Parts	Language of Parts: Inline language changes MUST be identified with a valid `lang` attribute.	Required WCAG 3.1.2
Language Code	Two-character language code: The language code SHOULD be designated with a standard two-character ISO 639-1 code (e.g. `lang="en"`) to achieve maximum support across screen readers and browsers, though other codes (e.g. `lang="en-us"`) are technically allowable.	Best practice

Headings

Topic	Technique	WCAG AA
Headings to Bypass Blocks of Content	Bypass blocks: Screen readers allow users to navigate by headings, so headings are an effective way to bypass blocks of content, as required by WCAG 2.4.1. Note: Headings are not absolutely required by WCAG to pass 2.4.1, but are highly recommended, along with landmarks and skip links.	Required WCAG 2.4.1
Meaningful Text	Accurate, informative section labels: Headings MUST be accurate and informative, as labels for the sections of text they describe.	Required WCAG 1.3.1 WCAG 2.4.6
	Brevity: Heading text SHOULD be concise and relatively brief.	Best practice
Heading Markup	Use real headings: Text that acts as a heading visually or structurally SHOULD be designated as a true heading (`<h1>`, `<h2>`, etc.) in the markup.	Best practice
	Heading Markup for Headings Only: Text that does not act as a heading visually or structurally SHOULD NOT be marked as a heading.	Best practice
Outline/Hierarchy of Content	Content outline: Headings SHOULD convey a clear and accurate structural outline of the sections of content of a web page.	Best practice
	Consecutive levels: Headings SHOULD NOT skip hierarchical levels.	Best practice
	First heading in the main content: The beginning of the main content SHOULD start with `<h1>`.	Best practice
	One <h1>: Most web pages SHOULD have only one `<h1>`.	Best practice

Form Validation and Feedback

Topic	Technique	WCAG AA
Labels and Instructions for Error Prevention	See the requirements and recommendations for Form Inputs, Labels, and Instructions, including: • Labels for inputs • Labels for groups of inputs • Instructions about inputs • Instructions about an entire form, a group, or a section • Required fields (in the full list of recommendations) • Data input restrictions (in the full list of recommendations) • Disabled fields • Time limits	Required (Multiple)
	Context-Sensitive Help: Context-sensitive help SHOULD be available.	Best practice
Critical Error Prevention	Web pages that process user input for any of the following: • legal commitments, • financial transactions, • user-controllable data (e.g. user profile, social media posts, OR • test/quiz responses • MUST implement at least one of the following error prevention techniques: • Reversible: Submissions are reversible. • Checked: Data entered by the user is checked for input errors and the user is provided an opportunity to correct them. • Confirmed: A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.	Required WCAG 3.3.4
Error Prevention (All Circumstances)	All web pages that process user input SHOULD implement at least one of the following error prevention techniques: • Reversible: Submissions are reversible. • Checked: Data entered by the user is checked for input errors and the user is provided an opportunity to correct them. • Confirmed: A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.	Best practice

Topic	Technique	WCAG AA
Error Detection on Submit	Error Identification: If an error is automatically detected, the input with the error MUST be identified. Valid techniques include the following: Add aria-invalid="true" to the input Identify the input (referencing the label): • in a simple JavaScript alert • with information associated with the input via `aria-describedby` (widely supported) or `aria-errormessage` (not yet widely supported) • with error text added to the input's label (other techniques are more semantically correct, but this is a reliable method) • with text on the web page (it may be appropriate to move the keyboard focus to the error message) • with an aria-live or `role="alert"` announcement • with information about the error in the page `<title>` if the submission causes a page reload or a new page load.	Required WCAG 3.3.1
Dynamic Error Detection	Visible Real-Time Error Messages: Real-time error messages MAY be scripted to show on the screen for sighted users, but attempts to announce the real-time messages to screen reader users can be problematic (see the next two rows below). It is usually acceptable to wait to announce real-time errors until after form submission, assuming that no data has been saved yet.	Best practice
	Live Announcements per Keystroke: ARIA live error messages SHOULD NOT be scripted to occur with every keystroke (to avoid overwhelming screen reader users), unless there is a delay built into the script to avoid announcements while the user is actively typing.	Best practice
	Live Announcements on Leaving a Field: ARIA live error messages SHOULD NOT be scripted to occur when a user leaves a field, because the aria-live announcement will may conflict with the screen reader's attempt to read the next element which receives focus, causing some information to be interrupted or not announced at all.	Best practice
Error Message Characteristics	Error Suggestion: If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text.	Required WCAG 3.3.3
	Programmatically-Associated: Error feedback SHOULD be programmatically-associated with the appropriate element.	Best practice
	Meaningful Error Message: Error feedback MUST clearly and accurately describe the error and/or how to fix the error.	Best practice
	Visible Error Message: Error feedback MUST be visible.	Required WCAG 3.3.3
Success Messages	Success Confirmation: The web page SHOULD confirm successful submission of data. Possible techniques include the following: • a simple JavaScript alert • with confirmation text on the web page (it may be appropriate to move the keyboard focus to the error message) • with an aria-live or `role="alert"` announcement • with the confirmation message in the page `<title>` if the submission causes a page reload or a new page load.	Best practice

Topic	Technique	WCAG AA
	Tab Panel widgets SHOULD conform to WAI-ARIA Authoring Practices for Tab Panels.	Best practice
	Table widgets SHOULD conform to WAI-ARIA Authoring Practices for Tables.	Best practice
	Table (Responsive, Collapsible) widgets SHOULD maintain data relationships through table structure, list hierarchy, and/or headings.	Best practice
	Table (Sortable) widgets SHOULD be constructed of a standard HTML table, if possible, and make full use of ARIA sort attributes.	Best practice
	Toolbar widgets SHOULD conform to WAI-ARIA Authoring Practices for Toolbars.	Best practice
	Tooltip widgets SHOULD conform to WAI-ARIA Authoring Practices for Tooltips.	Best practice
	Tooltip Dialog widgets SHOULD conform to WAI-ARIA Authoring Practices for Dialogs.	Best practice
	Tree View widgets SHOULD conform to WAI-ARIA Authoring Practices for Tree Views.	Best practice
	Window Splitter widgets SHOULD conform to WAI-ARIA Authoring Practices for Window Splitters.	Best practice

CAPTCHA

Avoid CAPTCHAs if Possible

CAPTCHAs require human problem-solving skills — which can be difficult or impossible for users with cognitive disabilities — and often require sensory abilities — which can be difficult or impossible for users who are blind, deafblind, or deaf. It is best to avoid CAPTCHAs altogether, and instead implement smart algorithms that do not require human input.

Topic	Technique	WCAG AA
Text Alternatives	**Text alternative describing the purpose:** If the CAPTCHA is not text-based (e.g. image or audio), a text alternative MUST communicate the purpose of the CAPTCHA, so that the user knows that this task must be completed before proceeding to the next step. **Note 1:** Ideally the alternative text would allow non-visual users to complete the task, but at a minimum it should inform users of the purpose of the CAPTCHA. **Note 2:** If there is an alternative CAPTCHA (e.g. in text elsewhere on the page, or in audio), the user SHOULD be notified that the alternative exists, by mentioning it either in the alternative text for the original CAPTCHA, or in the surrounding text.	Required WCAG 1.1.1
	Text-based CAPTCHA: A method SHOULD be available in a text-based format (either as the main CAPTCHA or as an alternative) that can be converted by a screen reader to braille output. Note: Although WCAG does not require a text-based CAPTCHA, deafblind users *require* a text-based method, because neither visual nor audio methods will be sufficient.	Best practice
Sensory Alternatives	**Sensory alternative:** If a non-visual user cannot pass the original CAPTCHA, an alternative method MUST be provided in another sensory modality (e.g. audio).	Required WCAG 1.1.1
Keyboard Accessibility	User input controls in a CAPTCHA (or in an alternative representation) must meet all the keyboard functionality requirements.	Required (Multiple)
Dynamic Content	Any dynamic content in a CAPTCHA (or in an alternative method) must meet all the dynamic content (JavaScript, AJAX) requirements.	Required (Multiple)
Custom Widgets	Any custom widgets in a CAPTCHA (or in an alternative method) must meet all the custom widgets (JavaScript, ARIA) requirements.	Required (Multiple)
Color and Contrast	Any visual elements in a CAPTCHA (or in an alternative method) must meet all the color and contrast requirements.	Required (Multiple)

