Developer Testing Checklist

Accessibility testing workflow for development teams

Prepared by: AccessGuard

Version: 1.0

Date: October 18, 2025

Standards: WCAG 2.2 AA, Section 508

Developer Testing Checklist

Table of Contents

[Back to TOC](#)

This checklist provides a systematic approach to accessibility testing throughout the development lifecycle. Use it during code reviews, before commits, and as part of your QA process.

Pre-Development

[Back to TOC](#)

Design Review

- ☐ Design includes accessibility annotations
- ☐ Color contrast meets WCAG AA requirements
- ☐ Focus states are defined for interactive elements
- ☐ Keyboard navigation flow is documented
- ☐ Screen reader announcements are specified

Technical Planning

- ☐ Semantic HTML structure planned

- ☐ ARIA patterns identified for custom components
- ☐ Accessibility requirements added to user stories
- ☐ Testing approach documented

During Development

Semantic HTML

- ☐ Using semantic HTML5 elements (`<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<footer>`)
- ☐ Heading hierarchy is logical (H1 → H2 → H3)
- ☐ Lists use `<ul>`, `<ol>`, or `<dl>` appropriately
- ☐ Forms use `<label>` elements for all inputs
- ☐ Tables have `<th>` elements with proper scope

Images and Media

- ☐ All images have `alt` attributes
- ☐ Decorative images use `alt=""`
- ☐ Complex images have long descriptions
- ☐ Videos have captions or transcripts
- ☐ Audio has transcripts

Forms and Inputs

- ☐ All inputs have associated labels
- ☐ Required fields are indicated with `aria-required`
- ☐ Error messages are associated with inputs via `aria-describedby`
- ☐ Error messages are announced to screen readers
- ☐ Autocomplete attributes are used where appropriate
- ☐ Form validation is accessible

Interactive Elements

- ☐ All interactive elements are keyboard accessible
- ☐ Focus indicators are visible and meet 3:1 contrast
- ☐ Focus order is logical

- [ ] No keyboard traps
- [ ] Skip links are provided for main content
- [ ] Touch targets are at least 44×44 pixels

ARIA Implementation

- [ ] ARIA labels provide accessible names
- [ ] ARIA roles are used appropriately
- [ ] ARIA states (`aria-expanded`, `aria-selected`, etc.) are updated
- [ ] ARIA live regions are used for dynamic content
- [ ] ARIA attributes are not used redundantly with semantic HTML

Automated Testing

Browser Extensions

- [ ] **axe DevTools** scan run and issues addressed
- [ ] **WAVE** evaluation completed
- [ ] **Lighthouse** accessibility audit passed
- [ ] **HeadingsMap** shows logical heading structure

CI/CD Integration

- [ ] Automated accessibility tests run on every commit
- [ ] Test failures block deployment
- [ ] Accessibility regression tests included
- [ ] Test results are visible to the team

Code Quality Tools

- [ ] ESLint accessibility rules enabled
- [ ] Stylelint checks color contrast
- [ ] HTML validator used
- [ ] No accessibility violations in console

Manual Testing

Keyboard Testing

- ☐ All functionality accessible via keyboard only
- ☐ Tab order is logical and intuitive
- ☐ Focus indicators are visible on all interactive elements
- ☐ No keyboard traps in modals or dropdowns
- ☐ Focus returns to trigger after closing modals
- ☐ Keyboard shortcuts are documented

Screen Reader Testing

- ☐ Tested with **NVDA** (Windows) or **VoiceOver** (macOS/iOS)
- ☐ Page structure announced correctly
- ☐ Links and buttons have descriptive names
- ☐ Form labels are announced
- ☐ Error messages are announced
- ☐ Dynamic content changes are announced

Visual Testing

- ☐ Page zoomed to 200% and remains usable
- ☐ Page zoomed to 400% and remains usable
- ☐ Text reflows without horizontal scrolling
- ☐ No content overlaps at high zoom levels
- ☐ Focus indicators remain visible at all zoom levels

Color and Contrast

- ☐ Text meets WCAG AA contrast requirements (4.5:1)
- ☐ UI components meet 3:1 contrast ratio
- ☐ Color is not the only means of conveying information
- ☐ Tested with color blindness simulator
- ☐ Tested in both light and dark modes

Component Testing

Buttons

- ☐ Accessible name provided
- ☐ Keyboard accessible (Enter and Space)
- ☐ Loading states announced
- ☐ Disabled state communicated
- ☐ Icon buttons have accessible names

Links

- ☐ Link text is descriptive (not "click here")
- ☐ Links are distinguishable from text
- ☐ External links are indicated
- ☐ Links are keyboard accessible

Forms

- ☐ All inputs have labels
- ☐ Required fields are indicated
- ☐ Error messages are associated with inputs
- ☐ Success messages are announced
- ☐ Form can be submitted via keyboard

Modals and Dialogs

- ☐ Focus trapped within modal
- ☐ Focus returns to trigger on close
- ☐ Modal has accessible name (`aria-label` or `aria-labelledby`)
- ☐ Escape key closes modal
- ☐ Background content is not keyboard accessible
- ☐ Modal is announced to screen readers

Navigation

- ☐ Navigation is keyboard accessible
- ☐ Current page is indicated
- ☐ Breadcrumbs are provided
- ☐ Skip links are present
- ☐ Multiple navigation methods available

Tables

- ☐ Table has `<caption>` or accessible name
- ☐ Headers use `<th>` elements
- ☐ `scope` attribute used on headers
- ☐ Complex tables have `headers` attribute
- ☐ Table is keyboard navigable

Responsive Testing

Mobile Accessibility

- ☐ Touch targets are at least 44×44 pixels
- ☐ Content is readable without zooming
- ☐ Forms are usable on mobile
- ☐ Navigation is accessible on mobile
- ☐ Gestures have keyboard alternatives

Different Screen Sizes

- ☐ Tested on small screens (320px)
- ☐ Tested on tablets (768px)
- ☐ Tested on desktop (1920px)
- ☐ Content reflows appropriately
- ☐ No horizontal scrolling required

Performance and Accessibility

Loading States

- ☐ Loading indicators are announced
- ☐ Skeleton screens are accessible
- ☐ Progress indicators have accessible names
- ☐ Timeouts are configurable

Dynamic Content

- ☐ Dynamic updates are announced via ARIA live regions
- ☐ Infinite scroll has "load more" button
- ☐ Auto-updating content can be paused
- ☐ Status messages are announced

Documentation

Code Comments

- ☐ Complex accessibility features are documented
- ☐ ARIA usage is explained
- ☐ Keyboard shortcuts are documented
- ☐ Known issues are documented

User Documentation

- ☐ Accessibility features documented for users
- ☐ Keyboard shortcuts documented
- ☐ Screen reader users have guidance
- ☐ Contact information for accessibility issues

Testing Tools Reference

Browser Extensions

- **axe DevTools**: Comprehensive accessibility testing
- **WAVE**: Visual accessibility evaluation
- **Lighthouse**: Built-in Chrome accessibility audit
- **HeadingsMap**: Visual heading structure
- **Color Contrast Analyzer**: Contrast ratio testing

Screen Readers

- **NVDA** (Windows, free): https://www.nvaccess.org/
- **JAWS** (Windows, paid): https://www.freedomscientific.com/
- **VoiceOver** (macOS/iOS, built-in): Enable in System Preferences
- **TalkBack** (Android, built-in): Enable in Settings

Online Tools

- **WAVE**: https://wave.webaim.org/
- **axe DevTools**: https://www.deque.com/axe/devtools/
- **Contrast Checker**: https://webaim.org/resources/contrastchecker/
- **HTML Validator**: https://validator.w3.org/

Pre-Deployment Checklist

- ☐ All automated tests passing
- ☐ Manual keyboard testing completed
- ☐ Screen reader testing completed
- ☐ Visual testing at multiple zoom levels completed
- ☐ Mobile accessibility verified
- ☐ Color contrast verified
- ☐ No console errors or warnings
- ☐ Accessibility statement updated if needed
- ☐ Code review completed with accessibility focus

Sign-off

Developer: _____ Date: _____
QA Engineer: _____ Date: _____
Accessibility Lead: _____ Date: _____

Deployment Approved: [ ] Yes [ ] No