Web for All: Enhancing Quality Intelligence with Automated Accessibility Testing using Playwright and axe-core

Rodrigo Silva Ferreira

rodrigosf672@gmail.com

Abstract

As organizations strive to build more inclusive digital experiences, accessibility (A11Y) has become a crucial component of software quality. Yet, A11Y testing is often overlooked or performed too late in the software development life cycle (SDLC), resulting in costly rework and missed compliance. This paper addresses the challenge of embedding A11Y checks early and continuously by integrating Playwright, a modern browser automation tool, with axe-core, an open-source A11Y testing engine.

Herein, a practical, low-maintenance approach to automated A11Y testing is presented, with the goal of showcasing how it can be seamlessly integrated into existing test pipelines and CI/CD workflows (*e.g.,* through GitHub Actions). Using real-world examples and axe-core-embedded Playwright tests, it is demonstrated how this integration helps detect issues like missing ARIA labels, poor color contrast, and incorrect heading structures. These checks can be performed without requiring specialized A11Y expertise. Furthermore, this approach can significantly improve detection of WCAG violations during nightly test runs, for instance, in addition to providing details and screenshots of potential WCAG violations.

By building A11Y into the fabric of automated testing, teams can enhance quality intelligence, reduce manual testing burden, and uphold their commitment to inclusive design. This paper shows that with the right tools and strategy, A11Y can be tested as reliably, systematically, and throughout the SDLC, as any other aspect of software quality.

Biography

Rodrigo Silva Ferreira is a QA Engineer at Posit PBC, where he contributes to the quality and usability of open-source tools that empower data scientists working in R and Python. He focuses on both manual and automated testing to ensure reliability, performance, and a seamless user experience across platforms.

Rodrigo holds a Bachelor of Science in Chemistry with minors in Applied Math and Arabic from NYU Abu Dhabi and a Master of Science in Analytical Chemistry from the University of Pittsburgh. His scientific background informs his analytical mindset, investigative thinking, and passion for quality.

Rodrigo enjoys working at the intersection of data, science, and technology, especially when building tools that help people better understand and navigate the world through its increasingly complex data.

1 Introduction

1.1. The Importance of Digital Accessibility (A11Y)

- The web is a critical gateway to information, services, and opportunities
- Ensuring A11Y is not only a legal requirement (*i.e.,* it's a moral imperative)
- Inclusive digital experiences benefit everyone, not just people with disabilities
- A11Y contributes to usability, SEO, and overall product quality

1.2. Challenges in Current Industry Practices

- A11Y is often considered late in the SDLC, if at all
- Manual A11Y audits are resource-intensive and may be deprioritized
- Lack of training or expertise among QA and development teams
- As a result, many teams miss critical violations until after release

1.3. Purpose and Scope of This Paper

- Introduce a practical, low-maintenance approach to automated A11Y testing
- Demonstrate how to integrate Playwright and axe-core into existing QA pipelines
- Show that A11Y can be tested early, often, and systematically, like any other aspect of software quality
- Emphasize how automation empowers teams to build inclusive products from the ground up

2 Background and Motivation

2.1. What is Accessibility (A11Y) in Web Development?

- Definition of digital A11Y
- The principle of inclusive design: ensuring websites/apps are usable by people with disabilities (*e.g.*, visual, auditory, cognitive, motor)
- Examples of common barriers (*e.g.*, unlabeled buttons, low contrast, inaccessible forms)

2.2. The Role of WCAG Guidelines

- Introduction to the Web Content A11Y Guidelines (WCAG)
  - Versions (***e.g.***, 2.0, 2.1, 2.2)
  - Conformance levels: A, AA, AAA
- Relevance of WCAG in legal, technical, and ethical contexts
- Industry expectations and compliance needs (e.g., Section 508, ADA, EN 301 549)

2.3. The Cost of Neglecting A11Y

- Business and reputational consequences
  - Examples of lawsuits and publicized A11Y failures
- Technical debt from late-stage A11Y testing
- Real-world impact: exclusion of users and missed markets

2.4. Why Automate A11Y Testing?

- Manual A11Y testing is essential but doesn't scale
- Automated tools help catch high-impact, low-effort issues early
- Automation empowers QA engineers and developers to contribute without needing to be A11Y experts
- Helps embed A11Y into CI/CD pipelines and shift-left quality strategies

3 Tools and Technologies

3.1. Overview of A11Y Testing Landscape

- Brief landscape of automated A11Y tools
  - Examples: axe-core, Lighthouse, pa11y, WAVE
- Manual vs. automated testing: limitations and complementary roles
- Rationale for choosing an automation-first approach integrated into existing QA pipelines

3.2. Playwright

- Introduction to Playwright
  - Developed by Microsoft
  - Supports Chromium, Firefox, WebKit
  - Cross-browser testing and modern JavaScript/TypeScript compatibility
- Key features relevant to A11Y testing:
  - Headless execution
  - Robust selectors
  - Easy integration with CI/CD
  - Native screenshot and tracing capabilities

3.3. axe-core

- Introduction to axe-core
  - Open-source engine by Deque Systems
  - Industry-standard ruleset based on WCAG guidelines
- Features:
  - Fast in-browser scanning
  - Granular rule tagging (*e.g.*, WCAG 2.0/2.1 AA)
  - Detailed violation reports (HTML elements, descriptions, fixes)
- Integration compatibility: browser extensions, JavaScript frameworks, Playwright

3.4. @axe-core/playwright Integration

- Purpose-built NPM package to combine Playwright and axe-core
- How it works:
  - Injects axe-core into the page context
  - Runs A11Y scans after navigation or interaction
  - Returns machine-readable results
- Advantages:
  - Minimal boilerplate
  - Seamless integration into existing Playwright tests
  - No need for deep A11Y domain knowledge to get started

4 Methodology

4.1. Tool Selection Rationale

- Justification for choosing Playwright over other browser automation tools (*e.g.*, Cypress, Selenium)
- Benefits of integrating axe-core as the A11Y engine (vs. other A11Y tools)
- Open-source nature, community support, and developer experience

4.2. Test Setup and Environment

- Project structure: where and how A11Y tests live within the codebase
- Required dependencies (Playwright, @axe-core/playwright, Node.js, etc.)
- Configuration steps: setting up Playwright with axe-core
- Example *beforeEach()* or global setup for A11Y scanning

4.3. Writing A11Y Tests with Playwright and axe-core

- Code example: basic A11Y test that runs axe-core scan
- Targeting key user flows and components (*e.g.*, navigation, forms, modals)
- Filtering rules/tags (*e.g.*, only WCAG 2.1 AA violations)
- Capturing violation details and screenshots

4.4. CI/CD Integration

- Integrating A11Y tests into GitHub Actions (or similar pipeline)
    - When the tests run (push, PR, nightly, etc.)
    - Caching, parallelization, test result uploads
- Configuring fail/pass thresholds (*e.g.*, continue on warning vs. hard fail)
- A11Y reports as build artifacts (HTML, JSON, screenshots)

4.5. Reporting and Monitoring

- Logging and surfacing violations to dev/QA teams
- Storing and reviewing results over time
- Highlighting regressions and missed issues

4.6. Test Coverage Strategy

- Page prioritization: choosing templates and user-critical pages
- Component-level vs. page-level testing
- Reusability of helper functions for scan automation

5 Results

5.1. Accessibility Coverage Improvement

- Increase in A11Y test coverage across key user flows and pages

- Types of issues detected more reliably (*e.g.*, missing ARIA labels, low contrast, invalid landmarks)

5.2. Early Defect Detection

- Shift-left impact: A11Y issues identified during development or QA, not post-release
- Reduction in time-to-detection and cost of fixing A11Y bugs

5.3. Integration into CI/CD Workflows

- Stability and frequency of nightly/PR-triggered A11Y scans
- Reporting integration (*e.g.,* HTML/JSON reports, screenshots, dashboarding)
- Developer feedback loops: how A11Y results are surfaced and addressed

6 Discussion

6.1. Interpreting the Gains in Accessibility Coverage

- What broader value does increased test coverage bring to the organization?
- How does automated detection of common A11Y issues support compliance and usability goals?
- Limitations of automated coverage: what still requires manual review (*e.g.*, keyboard traps, logical reading order)?

6.2. Impact of Early Defect Detection on Team Workflow

- How early detection changed development and QA practices
- The role of automation in shifting accessibility “left” in the SDLC
- Developer behavior: increase in issue ownership, fewer regressions over time

6.3. CI/CD Integration and Developer Trust

- How seamless integration affected test reliability and adoption
- Challenges faced in tuning sensitivity (*e.g.*, false positives, noisy failures)
- Developer perception: were results actionable, or did it require training/support?

6.4. Broader Organizational Implications

- Can the use of Playwright + axe-core influence accessibility culture?
- Scalability: can this approach support growing teams and codebases?
- How this fits into a larger digital quality or DevOps initiative?

6.5. Limitations and Areas for Future Work

- Limitations of current implementation (*e.g.*, dynamic content, complex interactions)
- Potential next steps: integrating visual diffing, combining with manual audits, extending test libraries
- Opportunities to refine rule sets or prioritize issues by impact (*e.g.*, user-facing severity)

7 Conclusion

7.1. Summary of Contributions

- Recap of the paper's main contribution: a practical approach to integrating automated accessibility (A11Y) testing using Playwright and axe-core
- Reinforcement of the key benefits: improved coverage, early detection, seamless CI/CD integration

7.2. Implications for Software Quality

- Emphasize that accessibility is a core part of quality (*i.e.,* not a separate effort)
- Automation enables teams to uphold A11Y standards continuously, not just during audits
- Integrating A11Y testing into the SDLC reduces rework and improves the end-user experience for all

7.3. Final Thoughts and Call to Action

- Encourage teams to start small: prioritize key user flows and incrementally build A11Y coverage
- Promote a mindset shift: accessibility should NOT be an afterthought; it should be a design and quality principle, fully integrated into the SDLC
- With the right tools and commitment, A11Y can be democratized and continuously tested across QA, dev, and product teams

References

Deque Systems. *axe-core Accessibility Engine*. GitHub. https://github.com/dequelabs/axe-core (accessed June 27, 2025).

Microsoft. 2020. *Playwright: Fast and Reliable End-to-End Testing for Modern Web Apps*. https://playwright.dev/ (accessed June 27, 2025).

U.S. Department of Justice. 2022. *Guidance on Web Accessibility and the ADA*. https://www.ada.gov/resources/web-guidance/ (accessed June 25, 2025).

World Wide Web Consortium (W3C). 2025. *Web Content Accessibility Guidelines (WCAG) 2.1*. https://www.w3.org/TR/WCAG21/ (accessed June 25, 2025).

World Wide Web Consortium (W3C). *How People with Disabilities Use the Web*. https://www.w3.org/WAI/people-use-web/ (accessed June 25, 2025).