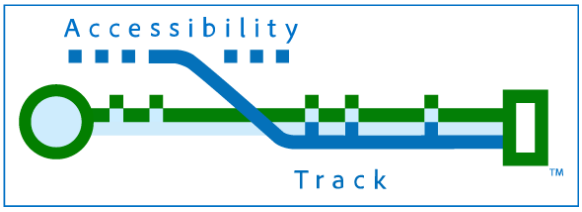
Proceedings of the 2018 ICT Accessibility Testing Symposium: Mobile Testing, 508 Revision, and Beyond

Arlington, VA, USA

November 7 & 8, 2018

George Mason University's Arlington Campus

Organized by Accessibility Track Consulting, LLC



www.ictaccessibilitytesting.org

Contents

Introduction from the Chair

Dr. Chris M. Law

Chair, The 2018 ICT Accessibility Testing Symposium

In this our third year of the Information and Communications Technology (ICT) Accessibility Testing Symposium, we continue to grow with expanded workshop offerings, a new competition, and a bigger planning committee. We have grown to have 19 papers, up from fourteen in 2017. From our first to second years we doubled attendance, and 97% of 2017 attendees surveyed were interested to return in 2018.

We are excited to have as our keynote speaker one of the most prominent lawyers in the ICT accessibility space, Lainey Feingold. Lainey's presentations are as entertaining as they are informative. Lainey will be discussing the importance of testing as it relates to legal compliance: if you aren't testing, how do you know where you stand?

Responding to a current gap in the accessibility testing field, the Symposium committee made Mobile Testing one of this year's themes. A mobile testing subcommittee put together a new, freely available Mobile Test Process that will be published on the Symposium website, and introduced in one of our five interactive workshops. The other theme for this year, the Section 508 Refresh is also featured in a number of paper presentations. We are also introducing a new annual competition, aiming to expand the knowledge-base in areas that have traditionally received less attention at accessibility conferences.

In this Symposium, we strive to foster an environment for networking and sharing professional practice and scientific analysis to collectively advance the ICT accessibility testing field.

Thank you for joining us this year in Arlington, Virginia.

—Dr. Chris M. Law

Keynote.
A Lawyer Goes into an Accessibility Testing Conference… (not a joke!)

Lainey Feingold

Disability Rights Lawyer

Lainey Feingold is a disability rights lawyer focusing on digital access, an international speaker, and the author of Structured Negotiation, A Winning Alternative to Lawsuits. Lainey's book is packed with win-win stories of accessibility advocacy with some of the largest organizations in the U.S., all without lawsuits. In 2017 Lainey was named one of the 13 Legal Rebels by the ABA Journal, the national magazine of the American Bar Association. That year she was also named the individual recipient of the John W. Cooley Lawyer as Problem Solver award, given annually by the Dispute Resolution Section of the ABA. Lainey has twice been recognized with a California Lawyer Attorney of the Year (CLAY) award (2000 and 2014) for her digital accessibility and Structured Negotiation legal work.

Workshop. Accessibility Testing 201

Gian Wild

AccessibilityOz
Melbourne, Victoria, Australia
gian@accessibilityoz.com

Sheri Byrne-Haber

Albertsons Companies
San Francisco, CA, USA
sheri@sheribyrne.com

Abstract

The Accessibility Testing workshop will cover a whole range of testing requirements: starting with an overview of testing – who, when, what and how. This will be followed by an in-depth discussion, with exercises, of how to develop a scope analysis for a site (based on the W3C Evaluation Methodology). The afternoon will be spent looking at testing tools – with a demonstration of the main automated accessibility testing tools such as Deque's WorldSpace, Level Access' AMP, SiteImprove and OzART – followed by a demonstration and discussion of one-page testing tools such as WebAIM's WAVE, the Paciello Group's Colour Contrast Analyser, mobile testing tools and readability tools. The workshop will be run by Gian Wild, CEO of AccessibilityOz and Sheri Byrne-Haber, Accessibility Director at Alberstons Companies, and formerly from McDonalds. Gian started in the accessibility industry in 1998 and spent six years with the W3C contributing to WCAG2 and has been testing web sites for almost twenty years. Sheri is an Accessibility evangelist with a history of successfully architecting strategic global accessibility programs in a Fortune 200 environment, including deploying corporate policies, accessible digital property design and remediation using LEAN and AGILE software development techniques, analytics, and maturity modeling. Multi-dimensional skillset includes degrees in information science, law, business, and professional certifications in ADA Coordination and Accessibility Professional Core Competencies.

Overview of presentation

In this half day workshop, the following will be discussed:

Choosing who will do the testing

Will the testing be done internally or externally? If internally, do staff have the correct skills? Are their other tasks going to be completed by someone else or do they need to complete the testing in conjunction with their daily job? If externally, will it be a consultancy or a contractor? How will the decision be made? What is the budget? Who will make the final decision? Who

will manage the external people? Will assistive technology testing be done? If so, it should be done by people with disabilities who are very familiar with their assistive technology. Can existing staff do this assistive technology testing?

Choosing when to test

While building a web site accessibility should be considered when writing requirements; at wireframe and design; and template. Training should be conducted for staff and the final site should be tested.

If an organization has many web sites then a plan needs to be developed to ensure that all sites are tested in an efficient manner.

Choosing what to test

What should be tested? Automated testing means that the entire site can be tested, but there is still a lot of manual testing required. Which pages should be manually tested? When choosing pages consider templates, processes, popular pages, pages required by law, pages aimed at people with disabilities, pages with unusual technology or functionality, third-party widgets and standard pages like the home-page, contact us page and search feature. Utilizing the W3C Evaluation Methodology will assist in identifying pages to review.

Choosing how to test

There are a number of testing methods and all may be applicable. Sites should be tested with an automated testing tool as well as manually. However, there are other options as well – should the site be tested by users of assistive technologies, on multiple operating systems and browsers and / or on different mobile and tablet devices?

Testing tools

There are many tools available including WAVE, Web Developer Toolbar, Deque Worldspace, Level Access AMP, AccessibilityOz OzART, Paciello Group’s Colour Contrast Analyser.

Choosing how to present findings

There are numerous ways to present results. The presentation of findings will be dependent on whether the site is an existing site, if it is being rebuilt or if it is being retired. In most cases a Word document is provided with examples of results. Automated testing tools allow for more detailed results that can be handed directly to the developers to fix. A walkthrough of the results for the project manager and developers is always helpful in addressing thorny accessibility issues.

Copyright notice

Workshop. Mobile Testing Process

Gian Wild

AccessibilityOz
Melbourne, Victoria, Australia
gian@accessibilityoz.com

Sunish Gupta

MIT – Easy Alliance
Boston, MA, USA
sgupta@easyalliance.org

Abstract

Testing on mobile devices (iOS, Android) will be demonstrated with opportunities for hands-on activities (don't forget to bring your phone!). Mobile versions of websites will also be covered, some of which are also testable using PCs (laptops). Learn what you can test on your laptop and what needs to be tested on a device. The testing topics will include handling traps, keyboard use, standard user interface controls, JavaScript events, touch targets and interactive space, links, images, forms, and navigation aids. Participants will receive a copy of the test process. This workshop covers the accessibility of mobile web sites—responsive and m dot sites. It does not specifically cover mobile app accessibility, however many of these principles can be applied to native mobile apps.

Overview of presentation

In this half day workshop, the following will be discussed:

How is mobile testing different to desktop testing?

- Mobile-specific Critical errors: Hover trap, touch trap, VoiceOver swipe trap, On-screen keyboard trap, Zoom trap
- Mobile-specific errors: orientation, scroll-bars, pinch zoom, touch targets, inactive space, navigational aids
- Mobile and Desktop relationship errors: consistency, restriction of content, choice of content
- Non-specific mobile errors: alternatives for items only displayed in mobile (eg. Hamburger menus), underlined links, reference to attributes

Steps to undertake mobile testing

1. Choose the devices to test with
2. Simulators and when not to use them
3. Capturing errors
4. Test Mobile-specific Critical errors on devices
5. Test Mobile-specific errors on devices
6. Test Mobile and Desktop relationship errors on desktop and simulator (or device)
7. Test Non-specific mobile errors on mobile-sized window, simulator or device
8. Test functionality with VoiceOver on iOS devices
9. Test functionality with TalkBack on Android devices
10. Test functionality with keyboard on iOS devices
11. Test functionality with keyboard on Android devices and Switch Access
12. Test functionality with Zoom on iOS devices
13. Test functionality with Magnification on Android devices
14. Review site with iOS – Invert colors
15. Review site with iOS – Grayscale view
16. Review site with Android – Grayscale view

Resources

Mobile Accessibility at W3C (https://www.w3.org/WAI/standards-guidelines/mobile/)

BBC Mobile Accessibility Guidelines
(http://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile)

Copyright notice

Workshop. Vetting before Getting: Considerations for Procuring Accessible Information and Communication Technology

Kelsey Hall

The University of Massachusetts Amherst
W.E.B. DuBois Library, 154 Hicks Way, Amherst MA 01003, USA
kelseyh@umass.edu

Kristina England

The University of Massachusetts President's Office
333 South Street, Shrewsbury MA 01545, USA
kengland@umassp.edu

Abstract

Accessibility isn't a "nice to have" – it's the law!

Title II of the Americans with Disabilities Act (ADA) requires higher education to provide all individuals with equal and equitable access to the benefits of services, programs, or activities. In 2010, the Department of Justice's Dear Colleague Letter clarified technology is covered under Title II. One of the key components of any accessibility program is the procurement of technology. You can implement accessibility standards and provide training and education, but without the upfront vetting of products and services, purchasers will face an upward battle with vendors that never ceases. In addition, without the proper vetting, the purchaser can and will be held liable should a formal complaint be filed.

So, what is the proper vetting process?

The recent 508 refresh, combined with the current working group addressing updates to the Web Content Accessibility Guidelines (WCAG) standards, sheds light on the need to establish policies, procedures, and consistency when procuring accessible Information and Communication Technology (ICT). However, not everyone shares the same ideas regarding accessibility. It is critical for accessibility allies to help others establish their "why" for procuring accessible ICT. Depending on the audience, their "why" may be related to moral, ethical, or legal/financial impact. A clear understanding of audience awareness will establish more success in terms of putting forth and executing solid policies and procedures, setting the institution up for accessibility success.

Preparing your institution

This workshop will cover how the University of Massachusetts is vetting products, software, websites and more to ensure WCAG compliance. We'll address the following considerations:

- Implementation of formal procurement standards
- Prioritization of products
- Example RFP and Contract language – what to avoid and what to emphasize

As part of the procurement discussion, we'll address contract renewals and expanded rollout of existing products.

Getting Hands On

Accessible means much more than "available". In order to understand what "accessible" means, we'll dive into actual testing of products with various assistive technologies. You should bring a laptop and a current product that you have questions regarding. You will have twenty minutes to try to get through a key process with only your keyboard and appropriate shortcuts. You will also use a mobile device to simulate additional experiences.

Preparing for this workshop

As you prepare for this workshop, you may be asking, “What is an accessibility compliance review? How do we prioritize products and services? How do I validate products?”

Join us to learn about the accessibility testing process, as well as the responsibilities of both the vendor and the procuring institution in ensuring accessibility for all.

Hands-on Activities Include

- Exposure to assistive technology
- Finding your “why” prompt
- Group collaboration on procurement language
- Question/answer session on organizational impact and procedures

Learning Objectives:

1. Participants will be able to discuss foundational laws and standards dictating the need for equal and equitable access to ICT.
2. Participants will practice and engage with basic testing principles to ensure vendor product accessibility.
3. Participants will be able to describe the inclusion of accessibility (when, where, how, why) as part of the overall procurement process, as well as the collaborative process in working with vendors on creating accessible ICT.

Copyright notice

Workshop. Assembling a Robust Accessibility Toolkit

Carol Gardiner

CGI Federal
12601 Fair Lakes Cir
Fairfax, VA 22033
carol.gardiner@cgifederal.com

Dominique Wheeler

Sevatec, Inc.
2815 Old Lee Hwy
Fairfax, VA 22031
dwheeler@sevatec.com

Abstract

A robust Accessibility Toolkit (A11y Toolkit) can provide organized and focused resources for a broad assortment of an organization's team members, i.e. project managers, business analysts, graphic artists, instructional designers, technical writers, developers, quality assurance specialists, and testers. The goal of an A11y Toolkit is to improve the accessibility and accountability of Information and Communications Technology (ICT) supported by a well-rounded set of tools and resources. This hands-on interactive workshop will guide learners in improving Revised Section 508 compliance through hands-on activities that will cover a sample set of free accessibility/testing tools and help learners organize existing resources currently in use. Learn how an ICT organization reviewed existing tools, identified new tools, determined "best meets" criteria, determined "current state" and "future state" of accessibility testing, and how to work with IT and governing boards to acquire new tools, both with and without access restrictions to browsers and networks.

Overview of Workshop

In this half-day workshop, participants are encouraged to bring their own laptops or mobile devices to install the sample toolkit. Information on the toolkit will be provided before the symposium. The following will be reviewed and discussed based on a representative set of available or open-source resources.

Identifying components of an A11y Toolkit based on your organization

1. Policy
2. Process
3. Reporting Requirements/Repository
4. Manual Tools
5. Automated Tools
6. Training Materials
7. Credentials

Tools — Budgets, approvals and use

1. Private industry vs education vs government (fed, state, local)
2. Identifying stakeholders
3. Resource requirements
4. In-house vs outsourcing

Accessibility Standards — what is mandated for your organization and why

1. Revised Section 508
2. WCAG 2.0 as Included by Reference (IBR) in Revised Section 508
3. Federal agency specific accessibility guidelines, i.e. the Department of Homeland Security's Trusted Tester process adopted by additional federal agencies and is being updated for the Revised Section 508
4. Section 504
5. International Standards

Team Structure and Onboarding

1. What is your current team structure?
2. Who on the team is responsible for accessibility?
3. Who on the team is responsible for testing?
4. Are there developer(s) on the team? If yes, are developer and testing duties separate?
5. Who on the team makes final decisions and resolves differences of opinion?
6. When new team members are on-boarded, is accessibility part of their standard training and expectations from the beginning?

OS Platform and browser implications

1. Windows
2. macOS
3. Mobile – Android/iOS/other
4. Other

Resources

1. Training materials
2. Easy one-page job aides
3. Code of Federal Regulations (CFR), if applicable
4. Section508.gov, if federal
5. Many spot-on webinars from commercial Accessibility Vendors

Tools

1. ANDI
2. Tota11y
3. WAFs/Toolbar Extensions
4. Developer rulesets (Lighthouse, aXe, etc.)

Sample Toolkit Practice

Copyright notice

Workshop. Understanding ARIA 1.1 and the ARIA Authoring Practices Guide

Jon Gunderson, Ph.D.

University of Illinois at Urbana/Champaign
Disability Resources and Education Services
1207 S. Oak Street
Champaign, IL 61820
jongund@illinois.edu

Abstract

Understanding the how the roles, properties and states defined in the W3C Accessible Rich Internet Accessibility (ARIA) 1.1 specifications is used by assistive technologies (e.g. screen readers) is critical for accessibility professionals to evaluate and provide guidance on how to identify and remediate web accessibility issues of online resources. ARIA technology is often not well understood by designers and developers as evidenced by high percentage of the ARIA found on the web not properly implemented and often times diminishing, rather than enhancing the accessibility of a web resource. The purpose of this workshop is to help participants understand how ARIA is designed to support the creation of accessible web resources that are work with a wide range of browsers and assistive technologies.

Workshop Oultine

An outline of the ARIA concepts that will be covered in the workshop:

- Concepts of role, properties and states
- Standardized mapping of ARIA and HTML5 semantics to accessibility APIs
- Required child roles
- Required properties and states
- Accessible name and description calculation
- Keyboard navigation within and between widgets
- Roving tabindex versus aria-activedescendant
- Keyboard focus styling techniques
- Presentation or None Role
- The “application” role for custom widgets

Communicating Accessible Design Patterns

When accessibility issues are identified, or inappropriate use ARIA markup is found the person making the evaluation needs a way to communicate the correct information to the designers and developers. An important resource in communicating proper accessibility techniques to designers and developers is the W3C ARIA Authoring Practices Guide (APG). The APG is reviewed reference for supporting the requirements of the W3C Accessible Rich Internet Application 1.1 specification and the use of HTML5 native semantics to meet W3C WCAG requirements. The APG provides the technical information need by interaction designers, developers and quality assurance personnel to design and test compliance with for WCAG requirements using the ARIA and HTML5 standards. The APG provides detailed information on how to use ARIA properties, states and roles to represent the interactive web resources to users of assistive technologies. The APG includes examples for landmarks and widget roles, and design information to support keyboard navigation between and within widgets, and how to describe relationships between content in web resources using HTML5 tags and ARIA markup. The APG examples have been extensively reviewed and tested with assistive technologies to help people understand and test the accessibility of ARIA enabled widgets.

Workshop Objectives

- Understand the concepts of role, properties and states
- Design patterns for landmark roles
- Design patterns for widget roles
- Keyboard focus styling techniques
- Keyboard interaction models for role semantics
- Live regions
- How screen readers use ARIA markup
- Tools and methods to inspect and validate the use of ARIA

Copyright notice

Competition. Are WordPress themes Accessibility Ready?

Santina Croniser

Regions Bank
santina@croniser.com

Abstract

This paper evaluates the accessibility capabilities of the open source content management service, WordPress. Using WCAG 2.0 Level AA success criteria, a gap analysis was conducted on WordPress themes labeled “accessibility ready”.

Introduction

“78% of CMOs see custom content as the future of marketing,” according to shopify (“Usage Statistics”, 2018). In order to generate this custom content, businesses are eschewing traditional sales, favoring online tutorials, tips, and videos. Unsurprisingly, this new content is typically generated with a content management systems (CMSs).

W3Techs reports that over half of all websites rely on a CMS. WordPress dominates the CMS market, covering 59.4% of all CMSs and 31.9% of all websites By comparison, other CMSs have a much lower adoption rate, with the nearest competitor (Joomla) holding only 6% (W3Techs, n.d.).

While larger companies prefer a custom CMS solution above a pre-built system, WordPress still accounts for 30.3% of the top 1000 most visited sites (such as Time, TechCrunch, and Wired) (Gelbmann, 2015). WordPress’s adoption across both small and corporate businesses sectors will influence accessibility across the web. This paper evaluates the accessibility of a typical WordPress solution using only the free resources available directly through the CMS.

Accessibility Evaluation Criteria

While Web Content Accessibility Guidelines version 2.0 (https://www.w3.org/TR/WCAG20/) (WCAG 2.0) is not an official governing body, many countries, states, and provinces directly reference WCAG 2.0 level AA compliance. Given it’s international acceptance, this paper will use WCAG 2.0 level AA to measure accessibility compliance.

Technology Utilized

WordPress CMS

Given the availability of a pre-existing WordPress installation, the website Guide Dogs Rule (http://www.croniser.com/guide-dogs-rule) was added to a previously installed multi-site instance. This WordPress instance is regularly maintained and is on version 4.9.8.

Theme

While comercial products may be more robust and may offer better accessibility features, WordPress developers often use a free theme as a basis for custom theme development. For this reason, only themes with the "accessibility ready" tag were considered. After reviewing a several options for both accessibility and aesthetic, the site uses the theme "Checathlon".

Plugins

Although plugin use was kept to a minimum, commonly used plugins are added to this website. Other plugins may be present (due to the multisite install), but they do not affect the functionality of the website Guide Dogs Rule. The plugins used to change the functionality of the site include:

- Checathlon Plus
- Contact Form 7
- SVG Support
- TinyMCE Advanced

Baking In Accessibility: Theme Selection

When searching for WordPress themes within the CMS, users are presented with an option to filter for themes that are "accessibility ready". These themes are evaluated and vetted by a team of volunteers working with WordPress. WordPress defines it's own acceptance criteria (https://make.wordpress.org/themes/handbook/review/accessibility/required/) for an "accessibility ready" theme rather than directly referencing WCAG 2.0.

Gap Analisis

Using the theme directory through the CMS, the are 122 themes that have passed WordPress's "accessibility ready" criteria at the time of this analysis. A simplified review of these themes in conjunction with the content, however, tells a different story.

The templates selected are inline with the ICT Symposium suggested criteria of three separate page layouts (fullpage, two columns with content on the right, two columns with content on the left). Of the options remained, only four themes were chosen at random for the scope of this paper.

Theme #1: Dog Channel

The first theme evaluated is the dog-themed "Dog Channel" (https://wordpress.org/themes/dog-channel/). While the theme offers several accessibility wins, several WCAG compliance issues render this theme inaccessible.

- 1.4.3 (Contrast (Minimum)): Several colors do not meet the minimum for color contrast compliance:
    - Default Link Color
      Foreground color: `#608000`
      Background color (sampled from image): `#E3D833`
      Color contrast ratio: 3.08:1.
    - Active Link Color
      Foreground: `#b50010`
      Background (sampled from image): `#e6a10b`
      Color contrast ratio: 3.19:1
- 2.4.3 (Focus Order): Clicking the "Menu" button triggers a full-screen modal. Using Mac's native screen reader (VoiceOver) with the keyboard to navigate the page, I can leave the modal without closing the modal.

The severity of these accessibility compliance issues (WCAG 2.0 level A) means that the theme was not a viable option.

Theme #2: Say Business

The second theme evaluated is the "Say Business" theme. Given the minimalistic nature of this theme, it could be assumed that accessibility compliance would be easier for the developer to manage. Unfortunately, this theme suffers from color contrast issues as well as other compliance issues.

- 1.4.3 (Contrast (Minimum)): Several colors do not meet the minimum for color contrast compliance:
    - Active Link Color
      Foreground: `#007acc`
      Background: `#f6f6f6`
      Color Contrast Ratio: 4.17:1
- 2.1.1 (Keyboard): While using VoiceOver with the keyboard, one cannot access the search button.
- 2.4.7 (Focus Visible): The theme seems to override a browsers default focus indicator as the focus is not visible on themed elements.

While other compliance issues are likely, these three are critical for website accessibility. Given the adverse effect of those using assistive technology, failing WCAG 2.0 (Level A) success criterion 2.1.1 alone is enough to eliminate this theme.

Theme #3: AwesomePress

The third theme evaluated for accessibility is the AwesomePress theme. What this theme lacks in pizzaz, it makes up for in accessibility. Color contrast issues are nonexistent -- this theme offers a high contrast as the default. Running automated checkers such as the browser plugin axe and Chrome's default Lighthouse show that this is a fully compliant theme. After a brief manual evaluation using keyboard only navigation in addition to VoiceOver shows no apparent accessibility issues.

The largest drawback of this theme is simply aesthetic. Even if a business were to forego a custom color scheme, this theme's layout and features are barebones. While AwesomePress is a worthy candidate for an accessibility ready theme, the monotone theme will likely be the dealbreaker for most businesses.

Theme #4: Checathlon

The fourth and final theme evaluated for accessibility is the Checathon theme. Offering simplicity beyond a monotone theme, this theme proved to be both accessible as well as aesthetically pleasing. Similar to the AwesomePress theme, Checathlon showed no apparent accessibility issues after both a manual evaluation as well as an automated evaluation.

Content Migration and Introduced Accessibility Issues

Migrating content into WordPress using the Checathon theme generally prevented accessibility issues. There were a few accessibility compliance errors that the theme did not or could not correct.

Forms

WordPress Themes typically account for a form's the visual elements. Although the theme presents the form with accessibility in mind, the plugin Contact Form 7 relies on the content editor to correctly code form labels. This causes a critical accessibility failure according to WCAG 2.0's success criterion 3.3.2 Labels or Instructions. Once the form labels were corrected, the page met WCAG 2.0 level AA criteria.

iFrames

Although WordPress has the capability to embed YouTube videos by using only the URL, the implementation of a YouTube iFrame is not compliant according to WCAG 2.0's success criterion 4.1.2 Name, Role, Value. The title attribute is missing from the iFrame embed code. This is a relatively easy fix with the embed code generated from YouTube. Adding a title attribute to the generated code will ensure the iFrame is compliant.

Unfortunately, the YouTube generated embed code introduced a new accessibility issue. By default, YouTube attempts to add the channel name as part of the embed code. Unfortunately, the channel name holds no discernable text. This triggers an accessibility violation of success Criterion 2.4.4 (Link Purpose (In Context)). Appending the URL with the variable `?showinfo=0` will correct this issue.

Tables

By default, one cannot add a table using the editing interface. This necessitates a table plugin. TinyMCE Advanced includes a user interface which allows a content editor to easily adjust table headers to meet WCAG conformance.

The WordPress “Accessibility Ready” Tag

Given that two out of four themes had significant accessibility issues, WordPress should remove the “accessibility ready” tag from these themes. Locating documentation about tag removal, however, proved to be a futile search. According to WordPress’s documentation (https://make.wordpress.org/themes/handbook/faq/#how-do-i-report-a-problem-with-a-theme-that-is-live), the onus to correct an issue is placed upon the theme owner. Although there is some precedent of themes being suspended if they do not uphold WordPress’s standards, no such precedent exists for the voluntary accessibility designation. An exhaustive search through the documentation leaves one with the impression that once the accessibility tag is granted, the tag is not removed.

Summary

Although WordPress allows theme filtering using an “accessibility ready” tag, the themes associated with this tag may not meet WCAG 2.0 Level AA conformance. Even when a theme is conformant, the theme will need additional assistance from plugins. As a final measure, content editors should receive guidance on how to utilize the theme’s features in conjunction with specific plugins in order to maintain an accessible website.

References

Gelbmann M. “WordPress powers 25% of all websites” W3Techs. https://w3techs.com/blog/entry/wordpress-powers-25-percent-of-all-websites. November 9 2015. Retrieved October 13, 2018.

W3Techs. “Usage of content management systems for websites”. https://w3techs.com/technologies/overview/content_management/all. Updated n.d. Accessed October 13, 2018.

WordPress.org. "FAQ – Theme Review Team — WordPress - make WordPress.org." https://make.wordpress.org/themes/handbook/faq/. Updated April 25 2017. Accessed 16 Oct. 2018.

Copyright notice

Gray-Box Mobile Accessibility Testing

Alistair Garrison

Level Access
1600 Spring Hill Road, Suite 400
Vienna, VA 22182
United States
Alistair.Garrison@LevelAccess.com

Abstract

The purpose of this paper is to consider a new method of mobile accessibility testing, applying so-called "gray-box" testing, which only uses higher-level, cross-platform, tests; and the widely-used testing frameworks / tools used to run them – and does not require any additional code to be "baked-in" a specially built test application! In addition, this method, being at the higher level, means a single set of higher-level requirements – such as WCAG 2.1 – can be used to assess the accessibility of any mobile UI (be it a native, hybrid or mobile web application).

Definitions

In this paper we'll use the following definitions from the ASTQB "Mobile Tester" Foundation Certificate:

- **Native Mobile Application**: A mobile application (app) that is designed for a specific device family and is coded to access specific functionality of the device normally via tools that have been specifically designed for the device.
- **Hybrid Mobile Application**: A mobile application that requires communication with the web server but also utilizes plug-ins to access device functionality.
- **Mobile Web Application**: A mobile application that is designed for use by a variety of devices with the majority of the code residing on the web server.

Introduction

I hope we can all agree that mobile phones, and their apps, are a critical component in most people's lives – interestingly, so critical that in some instances people will go hungry instead of running out of phone credit[(1)]; possibly equating their mobile with safety and security as in "Maslow's Hierarchy of Needs."

Quite frankly as such a critical piece in people's lives the ability for anyone (regardless of disability; or age) to access, and use, their mobile applications, as and when they need, is critically important.

To date, ensuring the accessibility of a mobile application (in whatever form) has, for some at least, been far from easy. With different Accessibility requirements, mainly set at the lower code level, needed on different platforms; along with different testing methods, skills, knowledge, tools and Assistive Technologies (AT).

Explaining the current issues with mobile accessibility testing

Meet fictional Bill; he's spent the last 5 years testing native mobile applications. He's responsible for User-Interface (UI) testing. His is a complicated role. In the past he has mainly coded "white-box" tests, adding them into larger groups or "test-suites". Each test-suite coded in specific computer languages (swift; java). Once satisfied, he has then had to then ensure that those test-suites are included along with the application's own source code in a specially built "test" variant, for execution.

It's even more complicated when he's been asked to test the accessibility of an application. As he's not an accessibility expert, he has had to rely on third-party code libraries for accessibility checking, and ask that they be included in the source code – then handle all the expected push-back, as you can imagine!

He's has also had the additional head-ache of having to deal with several sets of UI requirements for accessibility – effectively one per platform. Bill has often complained to colleagues "there's only one for web; so why can't there just be one set of requirements for Mobile?"

Over the past couple of years Bill, like many testers, has transitioned to tools which allow mobile phones to be operated, controlled, and tested from the outside – "over-the-wire" so to say. Using these tools Bill has dramatically cut his need to code the low-level "white-box" tests. Instead, he concentrates on writing higher-level tests, which can be used, as Bill describes "amazingly," on all platforms.

Bill longs for the day he can use these similar higher-level tests for accessibility checking too… To Bill, this just seems like the smart way to proceed.

A solution…

The purpose of this paper is to consider a new method of mobile accessibility testing, applying so-called "gray-box" testing, which only uses higher-level, cross-platform, tests; and the widely-used testing frameworks / tools used to run them – and does not require any additional code to be "baked-in" a specially built test application!

In addition, this method, being at the higher level, means a single set of higher-level requirements – such as WCAG 2.1 – can be used to assess the accessibility of any mobile UI (be it a native, hybrid or mobile web application).

The hope is to make Bill, and all other mobile testers, very happy – as testing accessibility should now be much easier, and more in-line with the way they work!

Gray-box testing

So, what is gray-box testing? Let's say you have a box, which is now sitting on the pre-scan conveyor belt of an airport luggage scanner…

The security guard looks at the outside of the box whilst it rolls up to the scanner. What they are doing is *black-box testing*: testing of a device, program or system whose workings are completely unknown.

The box rolls into the scanner, and the guard looks at an x-ray representation of what is in the box. That guard is now doing *gray-box testing* - testing of a device, program or system whose workings are **now** partially known.

When that box is flagged for opening, and the guard opens the box – then they are doing *white-box testing* - testing of a device, program or system whose workings are **completely** known.

Although, just as an FYI, in the context of software testing – white-box testing generally comes before gray-box and black-box testing, as it is done on the source code, rather than via the UI of a compiled application.

Why use gray-box testing over white-box testing…

As we indicated in Bill's story earlier, in a mobile testing context, white-box solutions generally consist of a set of separate iOS or Android tests which get "baked in" along with the source code during code compilation in order to create a testable version of an application.

Earlier, we also referred to white-box tests as being "lower-level." This was really due to the fact that they are intended for inclusion within the source code itself. At this lower code level, white-box tests can be used to create (instantiate) objects available within the code itself, for testing. As such, white-box tests have access to:

1. each UI component, and its attribute values, in the language of the UI Framework(s) used to create it; and
2. specific code-level knowledge about the broader application.

Access to the low level code is of course good, but, when thinking about accessibility testing in particular, the problem is that a good deal of home-grown accessibility testing knowledge / skill is needed in order to obtain meaningful / actionable results.

Developers particularly need to be cognisant of code-level requirements for ensuring the inclusion of mechanisms that expose additional text descriptions, amongst other things, to AT users. As such, developers need to aware of accessibility best practices on iOS[(2)], and Android[(3)], which cover platform specific accessibility mechanisms. Things like:

On iOS:

- **accessibilityLabel**: A succinct label that identifies the accessibility element, in a localized string.
- **accessibilityHint**: A brief description of the result of performing an action on the accessibility element, in a localized string.
- **accessibilityTrait**: The combination of accessibility traits that best characterize the accessibility element.
- **accessibilityFrame**: The frame of the accessibility element, in screen coordinates.

On Android:

- android:contentDescription XML attribute when labelling static elements;
- the setContentDescription()[4] method for labelling dynamic elements.

Testers, on the other hand, need to understand methods for testing the as-implemented accessibility features on the different platforms. Generally testing them through platform specific testing frameworks like XCUITest[5] on iOS, and Expresso[6] on Android.

And, all of this required knowledge for developers and testers also really dictates at least some appreciation of the ways different ATs on different platforms deal with coded accessibility features.

In teams which do not possess such deep accessibility knowledge, or skills, third-party accessibility testing code libraries are an option.

However, in more recent times we've seen the general market's appetite for such third-party solutions shrink; driven by factors such as security concerns – often associated with the need for these third-party code libraries to be "baked-in" to source code.

In light of this, a need has emerged for low-entry-bar accessibility testing which can be achieved from outside the mobile-application-under-test; and, here, is where gray-box testing comes into the picture; especially as gray-box accessibility testing has been used successfully for some time in a web context.

So, how does gray-box testing work? Firstly in a web context…

To get a high-level view we'll look at how Selenium, as a widely used gray-box testing tool, is used when testing web UI content.

Why? This will lay a good foundation for our later discussions on mobile UI testing.

Described in simple terms, Selenium testing requires three things:

1. **A test program**: where your tests to be run are held, and executed;
2. **A Selenium server**: a server which receives commands from the test program, and sends commands to a webdriver – and visa versa;
3. **A WebDriver**: a browser-specific executable that converts received selenium server commands into specific commands the browser responds to.

Testing accessibility, again, firstly in a web context

Whilst thinking about Selenium, we'll also take a quick look at how we test web UI content for accessibility - as this too will make the concepts of gray-box mobile accessibility testing easier to understand later on.

Described in simple terms, the Selenium commands used in accessibility tests are designed to:

1. move a web page to the correct view (technically DOM state); then
2. inject a JavaScript Accessibility Testing Library; execute the accessibility tests; and finally extract the results back into the test program.

For clarity, here a JavaScript Accessibility Testing Library is a stand-alone JavaScript library which consists of multiple tests, and the necessary APIs for running the tests and getting back the results.

The tests in a JavaScript Accessibility Testing Library primarily analyse the Document Object Model (DOM) tree – effectively the living code representation of all UI components, and their states.

The DOM tree is not the source code, it is instead a representation of the UI, and hence the reason, in this paper, we refer to testing the DOM tree as gray-box testing.

Now, how does gray-box testing work in a mobile context?

Gray-box testing works in a very similar manner in a mobile context – the main different really being that the WebDriver is essentially replaced by a platform-specific app-driver.

In some QA tools the same familiar "Selenium" commands used for web content can also be used e.g. to find, and interact, with UI components in a mobile application. This is made possible by the app-drivers the tools install, which communicate with a Selenium Server.

The app-drivers also make available an XML representation of the native UI components in their current state.

Now, testing accessibility in a mobile context…

The ability for app-drivers to make available an XML representation of the native UI components in their current state is fundamental to our being able to conduct accessibility testing using this process.

The XML representation is not, of course, the actual source code, just as the HTML DOM is not the web page source code. It is simply an image of the UI at that point in time. As such, it's really not too long a stretch of the imagination to think of this XML as an "x-ray representation" of the UI.

From an accessibility testing perspective, as long as this XML representation exposes the states of nodes, in terms of attributes and their values, we can test the accessibility information in this XML representation as an adequate stand-in, or proxy, for the native UI.

Interestingly, the app-drivers often build the XML representation we're looking to test using the phone's native Accessibility API in any case – making it even more useful, even a similar representation of how AT might view UI components.

The XML representation of a UI exposes sufficient detail for a good level of automated accessibility testing to be done, "remotely" – from outside the mobile application-under-test.

For clarity, when mentioning testing in a mobile context – we're talking about three different things:

1. testing native iOS or Android apps;
2. testing native hybrid apps (built using Cordova / PhoneGap);
3. testing web content through a mobile browser.

How does gray-box mobile accessibility testing work in the practice – using a QA tool

There are several QA tools which can be used to obtain this XML "x-ray representation," however, here we'll concentrate on Appium - a very popular free + open-source tool - although there are of course, many other great products.

Appium

Appium is a test automation framework for use with native, hybrid and mobile web applications, which can also be effectively used in conjunction with Continuous Integration (CI) servers, in support of Agile / Devops activities.

It should also be noted that Appium has very close ties with Selenium.

As mentioned before, instead of a WebDriver component, Appium essentially has platform-specific app-drivers to drive the UI interaction on iOS, and Android platforms.

During the test process the platform-relevant app-driver is installed automatically by Appium on a phone emulator or real device, enabling Appium to manipulate an app's UI. As mentioned before, the app-driver can also garner information from the phone[7,8] about the state of the UI, or components within the UI's object tree –through the collection of the XML "x-ray representation."

The UI's object tree is a living hierarchical tree-branch-leaf representation of each of component in the UI. It's equivalent to the Document Object Model (DOM) in a web context.

The XML representation made available through Appium's app-driver executables is generated through a tree-branch-leaf traversal of the native object tree for the application.

When an object is found in the native object tree, it is described – and that description is encapsulated as a properly nested XML-node, in that XML document.

Within Appium[9], the formation and return of the XML is executed via the "Get Page Source" function.

Although there are naturally differences between the XML representation of an iOS native UI, and that of an Android UI, when interpreted correctly, the exact same high-level accessibility tests can be used to test both.

But, that's not all…

The even more exciting news! As you no doubt know, WCAG 2.1 was started with the goal to improve accessibility guidance for three major groups: users with cognitive or learning disabilities, users with low vision, and users with disabilities on mobile devices.

Which raises a question – and possibly solves Bill's earlier complaint…

The question is then – can WCAG 2.1 be used, and tested for, on native mobile application UIs, in addition to its current mobile web application UI use-case.

Yes, it can! Through gray-box testing!

The reason being that WCAG 2.1 is platform agnostic, which deal with expected outputs to be tested for through the UI; rather than on a code level – so works well as a set of requirements you might want to test with higher-level tests.

Excitingly, by carefully translating the XML gained through gray-box testing to HTML5 it is possible to apply WCAG 2.1, and even more, to use the same JavaScript Accessibility Testing Engines used for testing web pages to test native mobile application UIs – bringing with them WCAG 2.1 test coverage, in their latest versions.

As noted before, the only real difference between testing native mobile applications, and mobile web application, when gray-box testing is that for mobile web applications the QA tools use a WebDriver instead of an app-driver.

But, some silver clouds have gray-linings – well, at least initially…

All very exciting, and shiny new… However, sadly, it seems that some silver clouds have a gray lining – well, at least initially…

There are presently a small number of speed-bumps in the path of this potentially "wonderful" testing approach. In a mobile testing context, through investigation these limitations seemingly sit around:

- The information exposed by the native object tree; and
- The collection of the exposed information which goes into building the XML meta-model of a rendered UI.

For full accessibility testing (let's say against WCAG 2.1) testers ideally need to have access to, and be able to test, all of the accessibility support mechanisms and values provided by the developer; however, currently not all of this information is available through the XML "x-ray representation."

The understanding that the market now has the desire for "gray-box" accessibility testing, combined with a gentle raising of the issues through papers such as this and a little lobbying, will I'm certain see these shortfalls quickly removed – and, hopefully collaboration between relevant parties such as Apple, Google, Appium, Facebook (as owner of the webDriverAgent app-driver on iOS platforms).

And, just so you know gray-box mobile accessibility testing is not stalled – whilst we wait for the hoped-for changes, shims for supplementing the XML to bridge the know gaps are of course being worked on – based on image processing methods, platform accessibility services, external applications.

The future is incandescent for gray-box accessibility testing in a mobile context; as it represents a low-bar, easily consumable path for "comparable across mobile product" accessibility testing; and may well also enable organisations to implement a single set of accessibility requirements, like WCAG 2.1, across all their customer facing UIs (web, mobile, etc…).

References and URLs

1) Mobile Testing – An ASTQB-BCS Foundation Guide. P12. ISBN 978-1-78017-404-4
2) https://developer.apple.com/documentation/uikit/accessibility?language=objc
3) https://developer.android.com/guide/topics/ui/accessibility/
4) https://developer.android.com/reference/android/view/View.html#setContentDescription(java.lang.CharSequence)
5) https://developer.apple.com/documentation/xctest?language=objc
6) https://developer.android.com/training/testing/espresso/
7) https://appium.io/docs/en/drivers/ios-xcuitest/
8) https://appium.io/docs/en/drivers/ios-uiautomation/
9) https://appium.readthedocs.io/en/latest/en/commands/session/source/

Copyright notice

Evaluating hardware accessibility

Sam Ogami

Program Manager
HP, Office of Aging & Accessibility
1501 Page Mill Road, Palo Alto, CA 94304, USA
sam.ogami@hp.com

Laura Renfro

HP Evaluation & Test Program Manager (Contract)
Senior Accessibility Lead
Renfro Consulting, Inc.
12720 Whippoorwill Lane, Randolph, KS 66554
laura.renfro@renfroconsulting.com

Abstract

The ICT Hardware ecosystem touches nearly everything we do. The variety of hardware products from mobile, telephony, laptops, wearables, and the internet of things are just some of the trends in technology that have become routine in our everyday life. These technologies offer the promise of making life easier for everyone, but, in fact, can be exclusionary to people with disabilities. Accessibility hardware testing provides a method to evaluate how accessible a product is for users with disabilities. Accessibility testing also helps product designers and developers identify limitations in their products during the development lifecycle. By understanding key terms, scope of use, and how to utilize comprehensive accessibility evaluation tools, designers and developers can optimize their products so that they are usable for an ever-increasing population of disabled users and fulfill the promise of making technology accessible to everyone everywhere. Using the Revised Section 508 hardware standards as a guide, this paper defines hardware accessibility requirements, when they apply, and how to use them to test products.

Introduction

Hardware accessibility testing requires an understanding of the product, the user experience, and accessibility requirements. Meeting hardware accessibility requirements alone are not sufficient indicators of a truly accessible product. Instead, the success of the user must also be considered as part of the evaluation. If a user is unsuccessful at a task or the task is very difficult, then the test would fail even if there is a technically correct way to “pass” the standard. The paper examines a particular product -- a multifunction printer -- to help illustrate our methods for hardware testing that incorporates the user experience.

Key terms

The Revised Section 508 requirements provide definitions that are important to understand when determining the applicability of specific hardware accessibility requirements.

- **Closed functionality:** Defined as "characteristics that limit functionality or prevent a user from attaching or installing assistive technology."
- **Equivalent Facilitation:** An alternative to a specified requirement that provides substantially equivalent or greater accessibility and usability. It has to be just as effective in terms of accessibility, usability, convenience, and reliability for people with disabilities.
- **Hardware.** A tangible device, equipment, or physical component of ICT, such as telephones, computers, multifunction copy machines, and keyboards.
- **Operable Part.** Hardware-based user controls for activating, deactivating, or adjusting ICT. If a part does not activate, deactivate or adjust the device, then it is not in scope of the Revised Section 508 physical requirements contained in the Reach, Height and Depth requirements. As an example, Figure 1 provides a description of operable and non-operable parts in a multi-function printer. Examples of operable parts for a multifunction printer include, but are not limited to, any doors or trays, the automatic document feeder guide, the control panel, any physical controls that are involved with page selection, and any settings on the printer. The scan glass is not an operable part as it does not move. However, the scan glass lid is an operable part.

Defining device functionality & creating use cases

What does the product do? This is an important first step in evaluation of any product for accessibility. With this, there needs to be some consideration of all physical controls and interfaces which are used for common, everyday tasks. Evaluators must understand the common end-user tasks of the device or software being evaluated because these are the highest priority when evaluating for accessibility. Service and maintenance tasks that are not commonly completed by end-users can sometimes be skipped in accessibility evaluations. Input from a product manager and the customers of the product can help define common user tasks. Separating common use tasks from low frequency tasks, such as maintenance tasks, can help give accessibility evaluators a more manageable list of items to test.

Once common tasks are established, detailed use cases can be created that specify each step for the users to complete these tasks. Use cases should be comprehensive starting from the first point of interaction with the device to completion of the intended task.

The common tasks can then be paired with the Revised Section 508 requirements for hardware. The criteria are grouped by like-items both with what the hardware might do and what the expectation is if it has that feature or functionality.

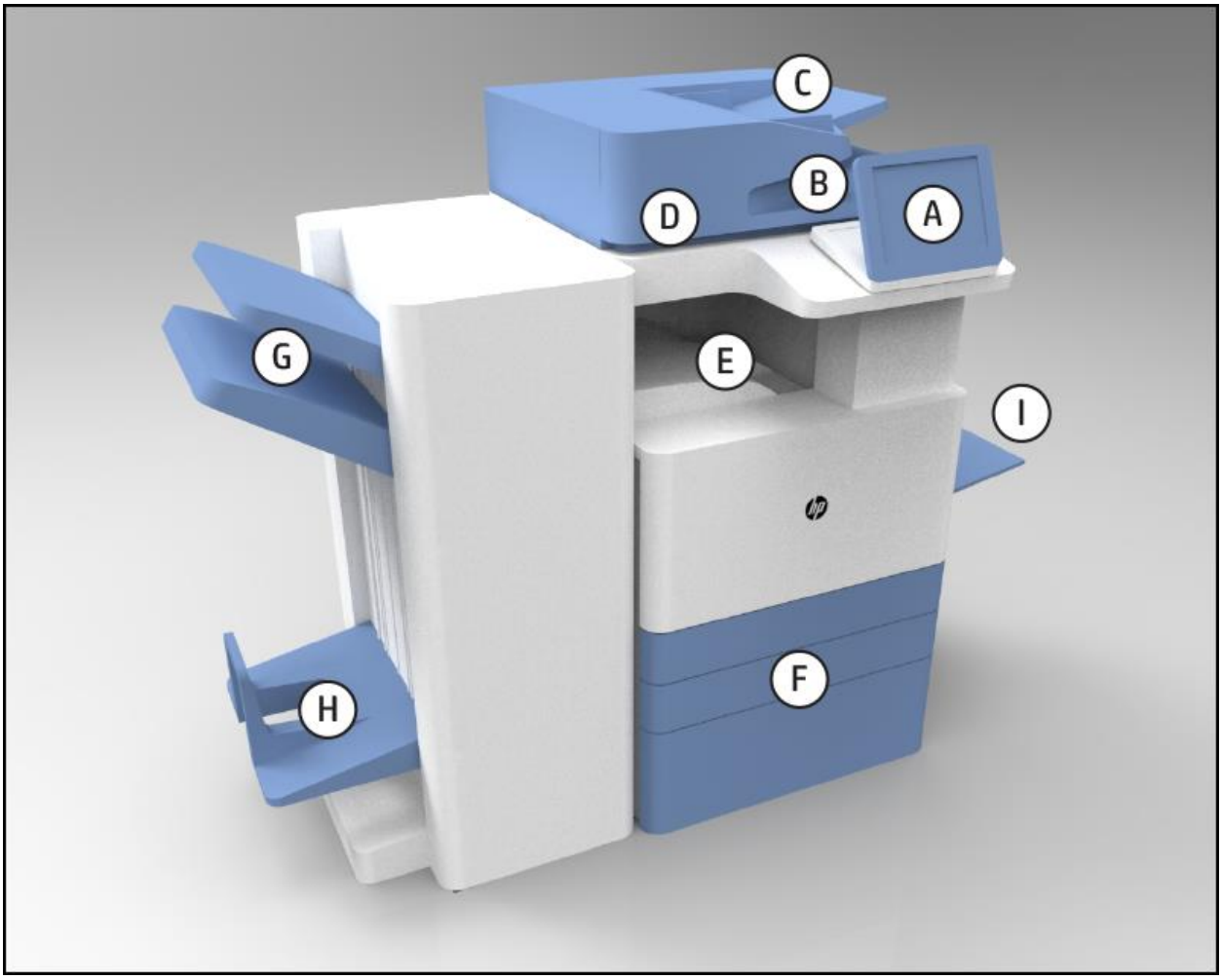


Figure 1 – HP Multi-Function Printer

Printer Component	Label	Operable Part?
A	Control Panel Touch Screen	yes
B	Scanner Output Tray	no
C	Automatic Document Feeder Guides	yes
D	Scanner Door	yes
E	Paper Output bin	no
F	Paper Tray Pedestal	maybe
G	Stacker – Output Tray	no
H	Tray – Output Tray	no
I	Manual Feed Tray	yes

Application: Evaluation of a multifunction printer

Some examples of common tasks for multifunction printers are Print, Copy, Scan, and Fax. When sending a print job to a multifunction printer from a computer, the task list should start from the print screen. Common settings that a user would change (such as color or monochrome, double-sided) need to be tested. Once the data is sent to the printer, access to the device where the print job can be retrieved and where paper needs to be inserted all must be evaluated. Measurements of the operable parts need to be conducted to determine if they meet height, reach and clearance requirements.

Figure 2 illustrates measurements of obstructed side reach requirements.

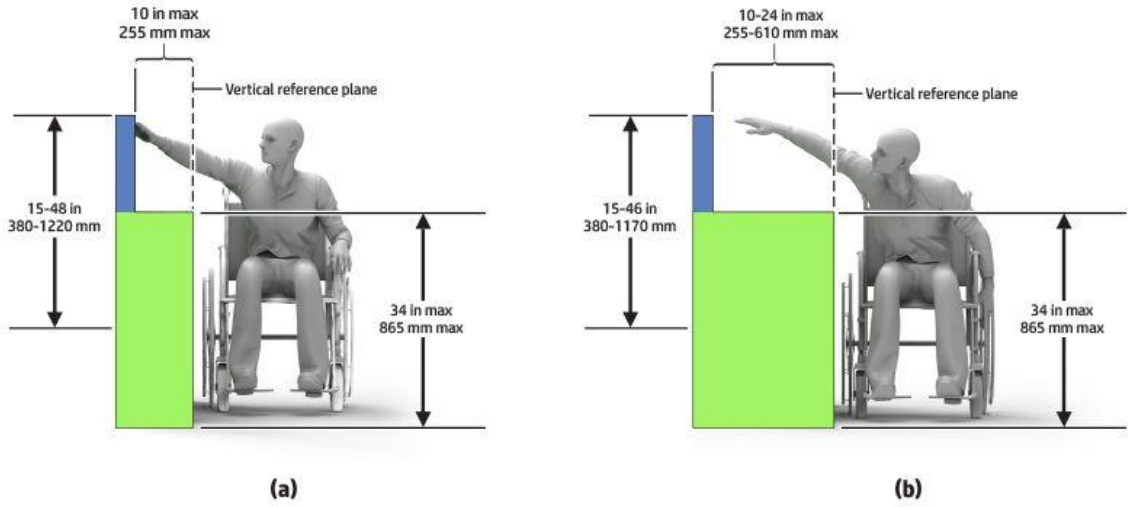


Figure 2 - Obstructed height and reach diagram.

Evaluation of trays for paper and the ink cartridges is necessary if a typical, everyday user would be adding paper and changing an ink cartridge. Test all aspects so that users know the force required to copy, change the quantity and even get the ink ready to add. Note alternative ways to accomplish different tasks. There may be software supporting the hardware that can greatly enhance accessibility and usability.

Multifunction printer desktop applications and drivers are not in scope of this paper, but should be evaluated. In some cases, users can print from an email using an application then the software application does not necessarily need to be the focus of the test, but how the hardware responds when activating the application is definitely the beginning of a common task for that hardware. Does the hardware light up for example when turned on remotely? Does the process work and can the user access the print job?

Multifunction printer mobile applications can be added to the hardware report, not as an actual hardware test, but to supplement the functionality and accessibility of the device which is being hardware tested. If a user can initiate activity and interaction using the mobile device, then the

testing for hardware begins after the execution of a command and testing is initiated at the point the printer is engaged remotely.

Keeping a “Notes” or “Remarks and Explanations” section is essential for good testing. The ancillary records give understanding to the test process and methods. The notes section can also be used to document an equivalent facilitation.

Tools used for evaluation

- A good camera is helpful to record both the test of the use case and as a record keeper to send to the development team in case defects are found. A camera can also be used to test for low contrast so that a digital, computerized test can be overlaid on the hardware. Sometimes a video can be uploaded into the bug tracking software to clarify to the engineering or development team what is happening, why it is failing the test and how to reproduce the test.
- A force meter (Figure 3) is necessary to test latches, buttons and doors for the 5 lbs. of force requirements. Make sure to get one with the ability to recalibrate, push and pull.
- A tape measure or measuring tape (for flexibility) is also needed to determine how the height and reach fall into the range of conformance if the hardware being tested is stationary.
- Good contacts with the engineering team are some of the best tools. For some of the requirements about sound or standardized connections, there needs to be a technical contact that can confirm or supply information.
- The user manual and other technical manuals may also contain invaluable reference material for some of the requirements needing testing.

Figure 3 – Force meter

Putting it all together

A basic understanding of functionality is essential for an accurate, thorough test. Assess the intended recipient or requestor of the testing as use cases are created so that the outcome benefit is maximized.

Accessibility evaluation is informative. Accessibility evaluation is done to improve hardware design and function for all users. Find and report defects so that a continual cycle of product fixes can happen, and hardware accessibility can improve.

Leave at least a quarter of bottom of the last page blank. This will be where we add the copyright notice.

Copyright notice

Launching Trusted Tester Version 5.0: Evolution and Improvements in the Trusted Tester Approach to ICT Accessibility

Allen Hoffman

Deputy Executive Director
Office of Accessible Systems and Technology
Department of Homeland Security
allen.hoffman@hq.dhs.gov

Katherine Eng

Senior ICT Accessibility Specialist
Co-chair Federal Accessibility Community of Practice Best practices Committee
U.S. Access Board
eng@access-board.gov

Pierce Crowell *(Contributor)*

Section 508 program Manager
Social Security Administration
pierce.crowell@ssa.gov

John Cotter *(Contributor)*

Accessibility Technical Lead
Social Security Administration
john.cotter@ssa.gov

Abstract

A Working Group from the U.S. Federal Chief Information Officers' Council, Accessibility Community of Practice (ACOP) has completed revision of the Trusted Tester process and the Harmonized Process for Section 508 Testing, from which the Trusted Tester process is derived, to align with the Web Content Accessibility Guidelines (WCAG) 2.0 and the Revised Section 508 Standards. The Working Group overcame challenges associated with collaboration among a cooperative interagency organization as well as the technical expertise necessary to create and revise such processes to produce version 5.0 of the Trusted Tester process. In addition to aligning with WCAG 2.0 and the Revised Section 508 Standards, Trusted Tester V5.0 also improves the overall format, flow, and construction of the process and its test conditions to improve readability, coherence, and the overall effectiveness of the test process. Finally, Trusted Tester has adopted a new testing tool, the Accessible Name and Description Inspector (ANDI),

developed in close coordination with the ACOP Working Group, and designed specifically to facilitate code inspection-based testing. ANDI's improvements in facilitating code-based inspection improve test efficiency while also simplifying test procedures. To support the revised Trusted Tester process, the Working Group has revised the entire Trusted Tester training curriculum while also improving access to and availability of the training for interested parties. The ACOP Working Group anticipates that these improvements to the Baseline, the Trusted Tester process, and associated training will improve individuals' and organizations' abilities to effectively adopt and implement standardized ICT accessibility testing methods.

Trusted Tester Environment and Context

Over the past twenty years, accessibility laws, regulations, guidelines, standards, and associated conformance validation processes for electronic content have evolved to improve accessibility for people with disabilities and keep pace with technology. The U.S. Federal Chief Information Officers' Council Accessibility Community of Practice (ACOP) Harmonized Process for Section 508 Testing (the "Baseline"), and the DHS Trusted Tester process, training, and certification have been updated as well. Testing tools have been updated to be more accessible, support developers more, and operate in a wider set of test environments. Updating the Trusted Tester process required: identifying a small group of dedicated and highly skilled individuals; collaborating with tool developers; reconciling differences in technical specifications, accessibility standards, and assistive technology state-of-the-art; and test case validation. The ACOP Working Group used GitHub to share ongoing working drafts of the Baseline and test process with interested individuals and accept comments over time.

Key success factors for the Working Group included:

- comprehensive test coverage;
- adherence to the Web Content Accessibility Guidelines (WCAG) Success Criteria (SCs) and conformance requirements;
- both tool-agnostic requirements, and a tool-specific test process; and
- unambiguous results determinations.

Some of the Working Group's lessons learned and observations were that: no freely available tools were sufficient to meet the groups' needs at the outset, test case development and organization required a significant level of effort, and the Group simply could have used more contributors.

The outcome of the working groups' effort is a solid foundation for organizations to develop:

- alternative test process documentation based on alternate tool sets;
- supplemental baseline and streamlined test processes for additional test environments and platforms;
- additional test cases to improve test process validation across the board; and
- test description documentation conforming to the World Wide Web Consortium (W3C) Accessibility Conformance Testing (ACT) Task Force test rules format.

Evolution of the Trusted Tester process

The Trusted Tester Program provides a tool-aided, code inspection-based testing process for evaluating the accessibility of Web and non-Web content and applications that is consistent and repeatable. The Trusted Tester Program also includes a training and certification track to ensure that individuals have sufficient knowledge to reliably and accurately perform accessibility testing. There are approximately 1300 certified Trusted Testers worldwide today. The Trusted Tester process has been demonstrated to aid small to very large organizations rapidly implement accessibility testing and expedite remediation.

In January of 2017, the U.S. Access Board released a Final Rule updating the Section 508 standards, directly incorporating WCAG 2.0 by reference. Through the leadership of the Federal CIO Council's Accessibility Community of Practice (ACOP)[(1)], a Working Group composed of representatives of multiple Federal agencies has worked to update the Trusted Tester process to support the Revised Section 508 Standards and WCAG 2.0. Aside from adding and revising tests to bridge the gaps between the original Section 508 standards and the revised standards, the Working Group sought other improvements to the test process, including:

- Increased applicability for the *Harmonized Processes for Section 508 Testing*, or "Baseline" tests, that form the foundation of the Trusted Tester process;
- Improved tools to facilitate manual code inspection;
- Increased transparency in ongoing test process development; and
- Improved clarity of test process linking test instructions and test outcomes

The Department of Homeland Security (DHS), Office of Accessible Systems and Technology (OAST) revised the entire 40-80 hours of free, online curriculum of Trusted Tester-related training courses and the certification exam. These improvements, including streamlining the enrollment process and increasing certification exam integrity, are expected to deliver a more efficient Trusted Tester certification process for students overall.

The United States Federal government has relied upon the Section 508 standards that went into effect in 2001[(2)] partially based on W3C WCAG 1.0[(3)] guidelines to ensure that information and communications technology[(4)] including electronic content[(5)] is accessible for people with disabilities. The Revised Section 508 Standards[(6)] released in 2018 rely on the W3C WCAG 2.0[(7)] Success Criteria and Conformance Requirements. The (ACOP) adopted a Baseline test process[(8)] for evaluating web and software applications for Section 508 compliance in 2013 based on the 2001 Section 508 standards. DHS created the DHS Trusted Tester training and certification from those resources. The ACOP has adopted an updated Harmonized Process for Section 508 Testing[(9)] for electronic content and Trusted Tester process[(10)] based on W3C WCAG 2.0 Success Criteria and Conformance Requirements. DHS has updated the Trusted Tester training[(11)] and certification to Trusted Tester V5 to begin in fall of 2018.

Tools and test environments

The Baseline v2.x included a specific set of testing tools while the revised Baseline (v3.0) eliminates references to specific tools and only includes test criteria, leaving reference to specific tools for individual test processes that derive from the Baseline. The Trusted Tester process V3 and V4 relied on the Web Accessibility Toolbar (WAT) with the Colour Contrast Analyser (CCA),[(12)] Microsoft Inspect, Java Ferret, as well as a list of bookmarklets in lieu of WAT, known as the Web Accessibility Favelets (WAF). The updated Trusted Tester process relies only on the Accessible Name and Description Inspector (ANDI)[(13)] and (CCA). Non-Web software interface elements will be supported in a subsequent parallel release of the Baseline and Trusted Tester processes in 2019. Trusted Tester V3 and V4 supported Microsoft IE11 and Chrome on Windows 7, 8, and 10. Trusted Tester V5 supports IE11, Firefox, Chrome, Safari on Apple MacOS, and optionally Microsoft Edge on Windows 10 only.

The new ANDI tool is the primary tool used in Trusted Tester V5 and represents a significant improvement in the ability to facilitate the Trusted Tester process and accessibility testing in general. The primary function of the tool is to identify the accessible name and description output for accessible objects based on W3C's Accessible Name and Description Computation[(14)] specification. ANDI also provides a number of other functions and features, such as contrast, document structure, reading order, and other types of evaluation, all specifically designed to facilitate manual, code-based inspection of web pages. In addition to combining and presenting (with minor exception) all of its features in a single, user-friendly favelet tool, ANDI provides all of its features and outputs in an entirely accessible and Section 508-conformant user interface.

Training and certification

Initially, the Trusted Tester training was instructor-led only and grew over time to a five day course and a one day skills exam. To scale delivery of training, DHS migrated to an online format with instructor-led support via webinar format. Migration to an online format allowed DHS to move from approximately ten new certified testers per month to nearly sixty to eighty per month without completely overwhelming the instructor support. Trusted Tester V5 training remains online with instructor support but separates Web only and non-Web software aspects into two trainings and certifications. The online Trusted Tester training is also now available for MAC OS users and users of all test environment browsers beyond IE11. The previous certification exam was composed of a set of web pages that testers had to evaluate and answer "compliant", "not compliant", or "does-not-apply". The new exam includes a set of pages which is randomized for each attempt to contain different content for testing. Once students get a specific test condition correct for an attempt, that particular test condition is not required for subsequent test attempts. However, the content is randomized on additional attempts for any test conditions not answered correctly, if the student does not pass the exam on the first attempt. This method reduces overall level of effort to complete the skills exam while also improving exam integrity.

Challenges developing Trusted Tester V5

The following is provided as guidance for others who wish to develop a test process derived from the Baseline or to create a baseline for another platform or set of standards.

While more platform-specific test procedures are needed, creating repeatable and accurate test processes is not an easy undertaking and should not be approached as a checklist exercise by anyone who is serious. First and foremost, identifying individuals who are knowledgeable about the entire range of standards, technical specifications, testing tools, and testing best practices is very difficult, yet critical for success. Understanding testing tool requirements, and finding tools that meet those requirements can be a challenge if only relying on static, already existing tool options. Having a developer working with the team who can modify tools as test requirements become more and more refined over time drastically improves the quality of the work outcome. Creating and utilizing a set of test cases to validate test criteria and test processes/tools requires access to knowledgeable developers for the platform the tests are created for. The number of test cases can become large, and regression testing of the process against test cases manually can become unmanageable. Clearly aligning technical specifications with accessibility standards, and platform support can also become challenging at times. For example, balancing the specification for Accessible Name and Description Computation[(14)] with browser implementations and known behaviors of assistive technology all guided the development of the ANDI, which drastically simplifies how some interactive elements are evaluated. The more test environments supported, the more test cases are required. The more test environments are supported, the less precise acceptance criteria can be, to some extent, to allow for minor environmental variances. How much training content vs. pure test process content to include in test processes is also always a concern. Trusted Tester V5 attempts to balance references to relevant standards and incorporate adequate guidance for the reading audience as well.

Outputs and outcomes

The outputs of the Trusted Tester revision process include:

- Revised, Harmonized Process for Section 508 Testing: Baseline Tests for Software & Web Accessibility.[(15)]
- Revised Trusted Tester Section 508 Conformance Test Process, Version 5.0.[(10)]
- Trusted Tester training has been updated in total. Current certified Trusted Testers must recertify to support Trusted Tester V5 testing and reporting. The Trusted Tester V5 training and certification consist of the following online courses with webinar-based instructor support[(11)]:
    - What Is Section 508 and Why Is It Important To You – overview of the Revised Section 508 Standards and roles of various key stakeholders in delivery of accessible information and communications technology.
    - Section 508 Standards for Web-based Electronic content – detailed introduction to Revised Section 508 standards for web-based electronic content with focus on W3C WCAG 2.0 level AA success criteria.

- Trusted Tester V5 Testing Tools – overview of required testing tools for Trusted Tester V5 for web content.
- Trusted Tester Training V5 for Web Content – detailed training for all Trusted Tester test steps, use of tools, reporting including knowledge and hands-on skills evaluation.
- Trusted Tester V5 for Web Certification Exam practice – hands-on skills evaluation practice for certification exam.
- Trusted Tester V5 for Web Certification Exam – certification exam.

Outcomes of the development effort include:

- A solid foundation for organizations to develop:
  - Alternative test process documentation based on alternate tool sets; supplemental baseline and streamlined test processes for additional test environments and platforms.
  - Additional test cases to improve test process validation across the board.
- Improved description of methods necessary to evaluate conformance against the Revised Section 508 Standards using a standardized, repeatable process.
- Improved availability, adoptability, consistency, and coherence of the Trusted Tester process and its related training and certification exam.

Reference URLs

1) https://www.cio.gov/about/accessibility-cop/
2) https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards
3) https://www.w3.org/TR/WAI-WEBCONTENT/
4) https://www.federalregister.gov/d/2017-00395/p-373
5) https://www.federalregister.gov/d/2017-00395/p-368
6) https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-ict-refresh
7) https://www.w3.org/WAI/WCAG21/quickref/?versions=2.0
8) https://www.dhs.gov/sites/default/files/publications/Baseline_Tests_for_Software_and_Web_Accessibility_v2_0_2.pdf
9) https://section508coordinators.github.io/ICTTestingBaseline/
10) https://section508coordinators.github.io/TrustedTester/
11) http://section508testing.net/
12) https://developer.paciellogroup.com/resources/contrastanalyser/
13) https://www.ssa.gov/accessibility/andi/help/install.html
14) https://www.w3.org/TR/accname-1.1/
15) https://section508coordinators.github.io/ICTTestingBaseline/
16) https://section508coordinators.github.io/TrustedTester/

Copyright notice

Rethinking Evaluation of Contrast

Jared Smith

WebAIM.org
6807 Old Main Hill
Logan, UT
84322-6807
jared@webaim.org

Abstract

This paper analyzes tools, guidelines, and methodologies for measuring color contrast, provides practical recommendations for using colors for optimal accessibility, and presents a proposal for modern research on contrast and contrast testing.

Introduction

Ensuring sufficient contrast is a core aspect of web accessibility and usability, yet contrast difficulties are pervasive on the web. A February 2018 analysis by WebAIM staff of the Alexa top 100 U.S. web site home pages revealed that 91 of the 100 most popular web site home pages had WCAG 2.0 AA contrast failures. The home pages on average had 70 contrast errors! Data show that contrast errors are likely the most common WCAG failure on the web – likely significantly increased with WCAG 2.1's expansion of the contrast requirements to graphical interface elements.

The pervasiveness of contrast issues on the web is hard to explain. Trends in modern web design have been toward lower contrast presentations. Much of this reality may be better attributed to lack of awareness and understanding of the impacts of poor contrast.

WCAG 2.0 and Contrast

WCAG 2.0 defines pass/fail thresholds for luminance contrast – the difference in brightness between text and the background of that text. WCAG defines a mathematical formula for generating a contrast ratio between foreground and background hexadecimal color values. This contrast ratio is then compared to defined pass/fail thresholds for text and "Large Text" (both including images of text). This formula is rather complex and not suitable for manual calculation. Fortunately, there are numerous tools, such as the WebAIM Contrast Checker (https://webaim.org/resources/contrastchecker/) that can quickly provide the relevant contrast ratio and indications whether the color combinations pass the various WCAG thresholds.

WCAG defines certain exemptions from the contrast requirements. These include "incidental text" (such as background text in an image that is not central to the content of that image), inactive or disabled controls, and logotypes.

WCAG 2.1, released June 2018, expands the contrast requirements from text to "User Interface Components" and "Graphical Objects" – both requiring a 3:1 contrast ratio. With the significant increase in the use of graphics, icons, and interactive controls on the web, this expansion is very welcome and is supportive of a better end user experience, especially for users with visual disabilities.

While the WCAG measurements can provide a reasonable measure of luminance contrast (which is generally considered the most important aspect of differentiation of elements with color), they are not perfect and very often do not align to human perception of content. There are many complexities in the presentation of color (such as background gradients and images, partial transparency, etc.) that are not adequately defined by WCAG or are not easily testable by automated tools.

Additionally, the research that informed the WCAG formula and thresholds was primarily conducted in the year 2000 on an Amiga 1000 computer. Displays have also significantly improved in the last 17 years to provide better contrast, color resolution, higher pixel resolution, etc. – all of which increase the perception of text content.

Itten's Color Contrasts – Moving Beyond WCAG and Luminance Contrast

Johannes Itten, a Swiss expressionist painter and theorist, was one of the first people to define and identify strategies for successful color combination differentiation or "contrast". Through his research he devised seven methodologies for describing color's contrasting properties.

1. Light-dark (luminance) contrast
2. Contrast of hue
3. Cold-warm contrast
4. Complementary contrast
5. Simultaneous contrast
6. Contrast of saturation
7. Contrast of proportion/extension/quantity

Each of these describe various ways in which color combinations can form contrasts that impact perception. The WCAG formula only analyzes one of these – light-dark (luminance) contrast (and to a very minor extent Contrast of proportion in the WCAG consideration of "Large Text").

While luminance contrast is by far the most impactful in determining contrast between colors, the other contrast properties can have notable impacts on human perception. Anecdotal research by WebAIM staff has repeatedly found this to be true.

As an example of just one of these, consider the two texts in the image above. When printed in color, the one on the left has a maroon colored text on an orange background and the one on the right has blue text on a grey background. When asked which of these is easiest to discern or has the best contrast, nearly all respondents indicate that the blue on grey is better. However, when analyzed via WCAG's luminance contrast formula, both have an identical 4.5:1 luminance contrast difference. Then why do respondents almost universally find the blue on grey more "contrasty"? Because of contrast of hue (and perhaps to some extent cold-warm contrast) – the maroon and orange are similar hues of red and thus are perceived with lower contrast than the blue on grey which are entirely different hues. WCAG's formula does not account for this factor, or many other factors, of human contrast perception. This misalignment can very often cause users to identify certain color combinations as passing WCAG due to adequate perceived contrast, when they in fact fail, and vice versa.

To further complicate things, WCAG defines pass/fail states for contrast. This type of binary approach to accessibility does not correlate to end user experiences for users with disabilities or others. This approach would suggest that any color combinations that pass are "accessible" whereas those that do not pass are not discernable at all. The human variability in perception of colors is much more dynamic and variable. Certainly many combinations that "fail" would be highly perceivable by some users and many combinations that "pass" would be difficult for others. This human variability and its impact on true end user accessibility is often overlooked due to the pass/fail thresholds defined by accessibility guidelines.

Automated Measurement of WCAG Contrast

While WCAG defines a relatively straightforward (though somewhat mathematically complex) formula for measuring luminance contrast, complexities of the modern web make it so automated testing of contrast failures can be incomplete or inexact. Colors can be defined on the web in numerous ways. Conversion to the required hex values and sRGB color space for WCAG's formula is possible, but adds complexity to automated testing.

While determining the foreground and background color and the contrast ratio thereof for any particular page element is relatively easy in a programmatic way. However, just because a foreground and background color are defined for an element does not mean that text is actually presented in that element. Flagging all elements with insufficient "potential" contrast as WCAG failures could lead to significant coding effort to remove tool "failures" that aren't actually WCAG failures or that have any end user impact.

Testing tools must instead perform calculations to determine if text is present – and what the color and background of that text is – before determining the WCAG contrast conformance of that text. These calculations can be difficult, especially due CSS inheritance – the actual foreground and background colors for elements that contain text may be defined in parent elements to those text elements. For example, a web page may define a page-wide text color and background color that is conformant. Then, a different background color may be defined for a footer within that page. The color for the text in that footer element is defined at the page level whereas the background color is defined for the footer itself. And it's very likely that any text in the footer is within child elements (such as paragraphs or lists) within that footer.

Calculating actual colors for only text elements and filtering test failures to only elements that contain non-conformant combinations requires notable programming logic. Considerations of opacity and complexity of backgrounds (especially if multiple semi-opaque elements overlap) or if text is positioned via CSS to overlap an element with an entirely different background also introduce barriers and increase the possibility of inaccurate or incomplete contrast tests.

WCAG 2.1's requirement for 3:1 contrast for non-text elements, such as interface elements (form controls, etc. – including states, such as mouse hover and keyboard focus) and graphical objects further complicates the accuracy and efficiency of automated testing for these. For security reasons, browsers generally limit DOM analysis of state characteristics of interactive elements (such as the hover state of links). Defining automated tests to accurately and completely measure all of these factors ranges from difficult to impossible.

Automated testing of contrast of graphical objects and text within images is also inherently complex and difficult – typically requiring pixel-level analysis of image files. WCAG does not provide explicit guidance on how to measure contrast for things like aliased text borders, transitions and gradients, multi-color graphical elements, etc.

All of the above considerations for automated testing highlight the need for manual analysis of contrast for many web elements – and such manual testing tends to be rather intensive.

Descriptions of Luminance Contrast

Another consideration of WCAG's approach to contrast measurement is how it is presented. The WCAG formula generates a contrast ratio ranging from 1:1 (no luminance contrast) to 21:1 (white on black, or vice-versa). This ratio is not as user-friendly or understandable as other metrics, such as a percentage (0% to 100%). Future WCAG updates may consider the approachability and understandability of how luminance contrast and WCAG ratios are defined and presented. Such metrics may be implemented into automated contrast tools today for better understandability, though at a risk of deviating from WCAG's long-standing metric for contrast measurements.

Exploring Color Perception

Luminance contrast alone is a demonstrably insufficient measure of human perception of contrasting color combinations, particularly for text content. While luminance is most likely the most important factor in color differentiation, it is not capable alone of reflecting end user contrast perception. An in-depth analysis of the WCAG formula is needed. Certain color combinations are favored or disproportionately affected by this formula in ways that may not align with user perception. In other words, there's much more to contrast and human perception than can adequately be analyzed via WCAG's formula and pass/fail thresholds.

Moving Forward – Research and Revisiting WCAG

WebAIM is designing research to collect data that may inform several aspects of contrast perception. This research would consider factors of contrast beyond luminance contrast to see if/how these factors impact human perception. The primary research questions are:

- Which foreground/background color combinations and patterns most impact perception and readability? This would likely be captured via A/B choice based on optimal perception between pre-defined color combinations (similar to the graphic presented above).
- Are users with dyslexia (which may be impacted by too much contrast) or visual disabilities - low vision, color-blindness, and low contrast acuity - impacted more notably or differently with certain combinations/patterns than others? Certainly any considerations for improvements to WCAG must be focused on accessibility for users with disabilities, beyond just color perception by users generally.

Secondary research questions are:

- If and how does our human research data align with the Web Content Accessibility Guidelines (WCAG) 2.0 luminance contrast formula and its pass/fail thresholds? Are there certain characteristics of colors that deviate or are contrary to WCAG's prescriptions – and if so, can these be identified in useful and measurable ways.
- Are there differences in contrast perception based on culture or other demographics? It's possible that aspects of color contrast perception are learned – such as certain color combinations that are considered pleasant and complementary in East Asian cultures that are considered conflicting in Western cultures.
- Do hardware or software considerations result in different responses? Do mobile phone displays vs. desktop displays make a difference? How does display resolution or color depth impact perception of text of various contrasts?
- Are there properties of better/worse combinations that can be defined and analyzed in a way that is more prescriptive for web authors in generating more usable and accessible contrast on the web? In other words, could the WCAG formula be modified or expanded to better align with true end user perception and accessibility?

This research could provide insight into the alignment of the current WCAG 2 luminance contrast measurements to end user perception. If adequate data is collected, it could additionally help prescribe better measurements for contrast that account for other impactful color factors. A new (and inherently more complex) formula for measuring contrast levels between color combination may be necessary. This formula would most likely be based on a more expansive definition of color spaces than what it utilized by the current WCAG formula – a color space that considers luminance as only one factor of color difference, but that also considers additional factors of contrast, such as similarity of hues, color mood/warmth, complementariness, and saturation.

This research could thus lead to improvements to WCAG in the future, and to better manual and automated testing tools and methodologies that provide more accurate, useful, and meaningful feedback regarding end user usability and accessibility of certain color combinations.

Copyright notice

Testing social media for WCAG2 compliance

Gian Wild

AccessibilityOz
Melbourne, Victoria, Australia
gian@accessibilityoz.com

Abstract

Social media is an incredibly important tool in modern society. For three years, four social media platforms have been tested for accessibility compliance: Facebook, YouTube, Twitter and LinkedIn. In 2018, testing identified three common errors across most of the social media platforms: the inclusion of auto-play, the use of infinite scrolling and incorrect focus order.

Methodology for testing the accessibility compliance of social networks

Every year since 2015, the following four social media systems are tested.

- Facebook
- YouTube
- Twitter
- LinkedIn

The standard user journey tested was Register, Log in, Read item and Submit item. The systems were tested with the keyboard only, increasing text size and via a person with a vision impairment using a screen reader. The testers were expert in the W3C Web Content Accessibility Guidelines and identified only those issues that would stop a group from being able to use the web site.

Please note that this was not a complete accessibility audit – other errors may also occur.

Common accessibility errors

A number of accessibility errors occurred across the four platforms.

Auto-play

Automatically playing a video on load of a page is a serious accessibility issue and in violation of one of the four non-interference clauses in WCAG2 Level A Success Criterion 1.4.2: Audio

Control. This can render the web site inaccessible to a number of groups of users, including screen reader users, whose audio rendition of the page will clash with the audio in the video.

It is important to note that these platforms often have a way to turn off the auto-play feature, however it is often not easy to find or accessible itself. For example, YouTube has also added a feature where the next video automatically plays after the previous video is finished. There is a feature to turn this off, however due to the inclusion of infinite scrolling, this feature may not be available on all videos (see Figure 1).

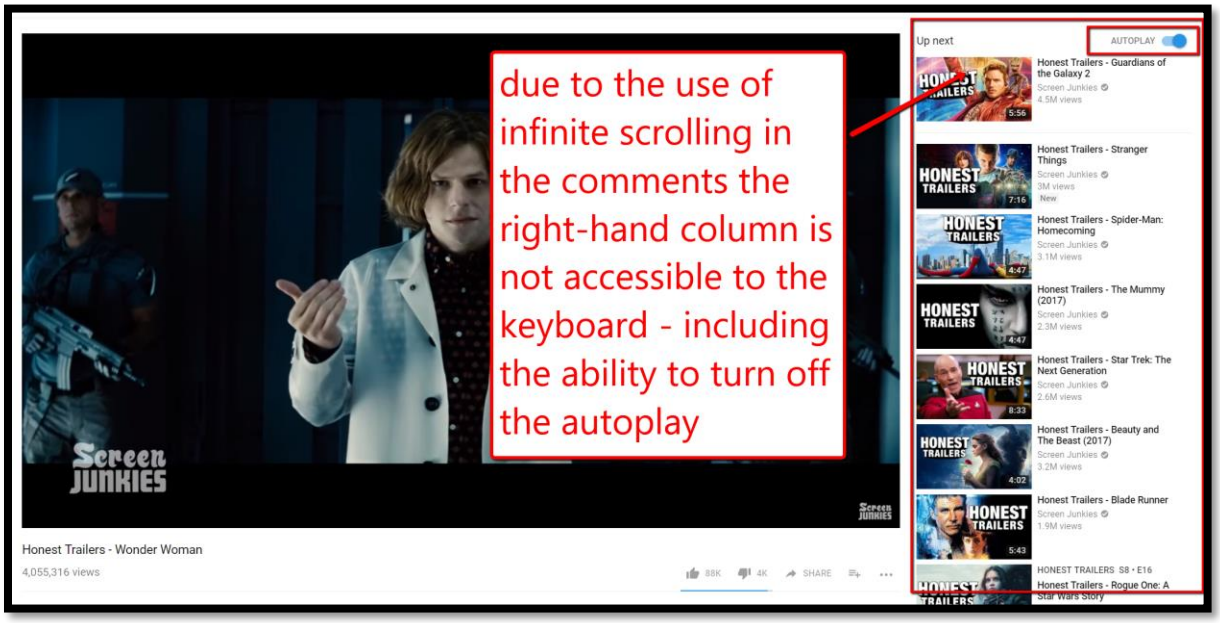


Figure 1 – Infinite scrolling in the comments section means that if a video has thousands of comments, the turn off auto-play feature cannot be accessed by keyboard-only users

Videos on all four social media platforms play automatically. In addition, videos within tweets automatically play, and are also not accessible to the keyboard. This means anyone relying on a keyboard, or an assistive technology such as a screen reader, will be unlikely to be able to stop the video playing.

Focus order

The keyboard focus order of columns in Facebook and YouTube do not match the visual order. This can cause significant confusion for people using the keyboard, as they are unsure as to where the keyboard focus is. This can also be a problem for people with cognitive disabilities using screen readers, as the screen reader will read the columns out-of-order.

The visual order of YouTube does not match the keyboard focus order. The header is the first to receive focus, followed by the central column.

In Facebook, the header receives focus first, followed by the left-hand column. Then the focus jumps to the right-hand column and only then does it move to the central, news feed column (see Figure 2).

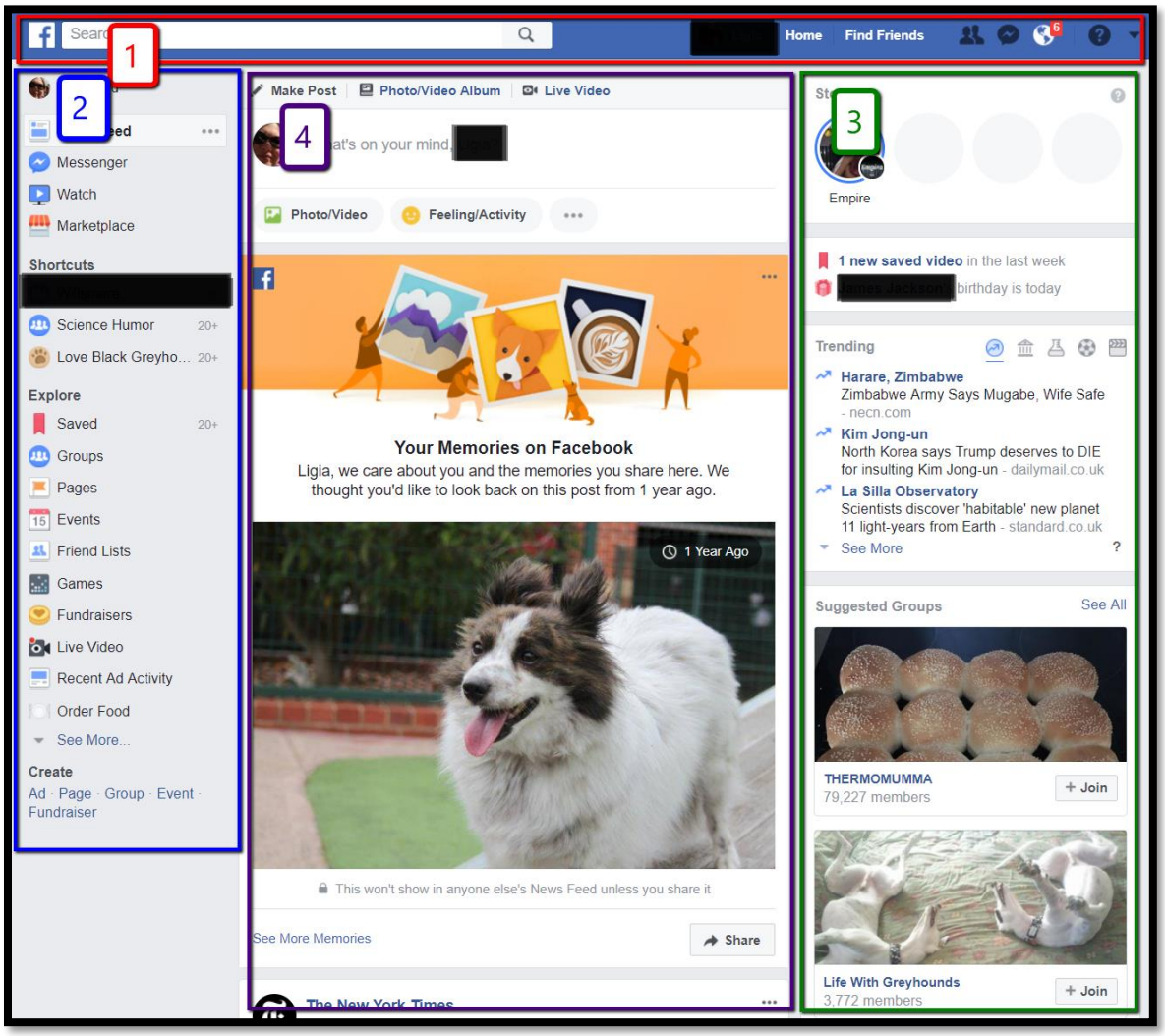


Figure 2 – The order the columns receive keyboard focus is incorrect

Facebook has attempted to mitigate this by instituting skip links, which allow a user to skip to particular sections of the page (see Figure 3).

Figure 3 – Facebook skip links

However this appears at the top of the page only on keyboard focus, so is not available to all users. The instructions are also unclear, referring to a capital I, l or slash mark (see Figure 4).

Figure 4 – It is not clear how to open this top menu

Infinite scrolling

Infinite scrolling: where additional content is added automatically once the user reaches the bottom of the content can also render a social media platform inaccessible. As content is automatically added to one of the columns as the user moves down the page, keyboard-only users can never move past the column that contains infinite scrolling. Due to the use of infinite scrolling in Facebook, the user becomes trapped in the central column and cannot access the Friends list, which appears last in the column order.

In YouTube, due to the use of infinite scrolling in the central column, the left-hand column (which is next to receive focus) cannot be accessed via a keyboard (see Figure 5).

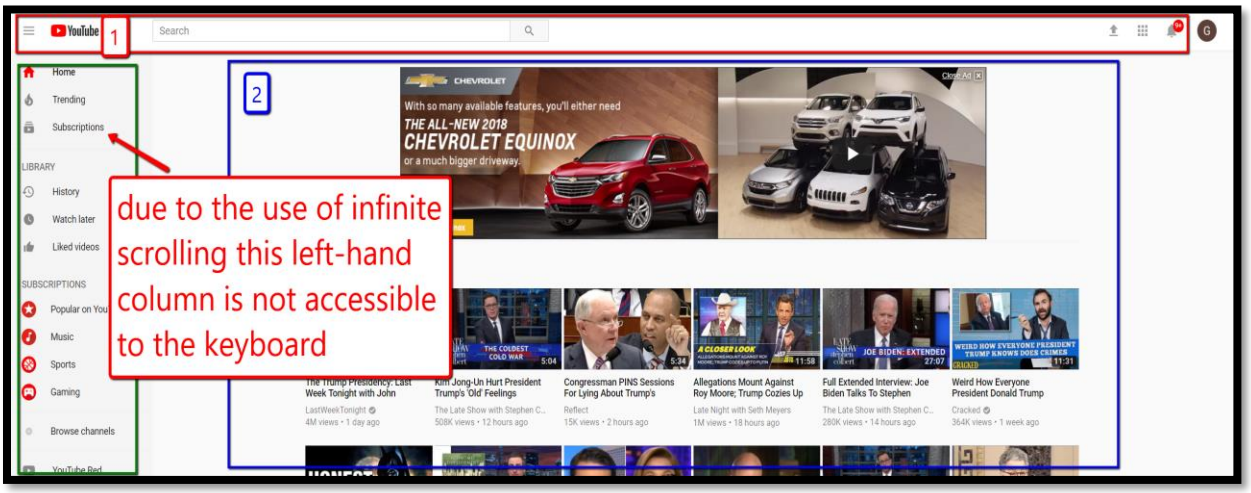


Figure 5 – Infinite scrolling in the central column means that the left-hand column cannot be accessed by keyboard-only users

Although the focus order of columns in Twitter and LinkedIn are correct, the central column utilizes infinite scrolling, and as a result, it is not possible for a keyboard-only user to access the right-hand column in Twitter (see Figure 6). It is possible to access the right-most column in LinkedIn, but only through the use of skip links.

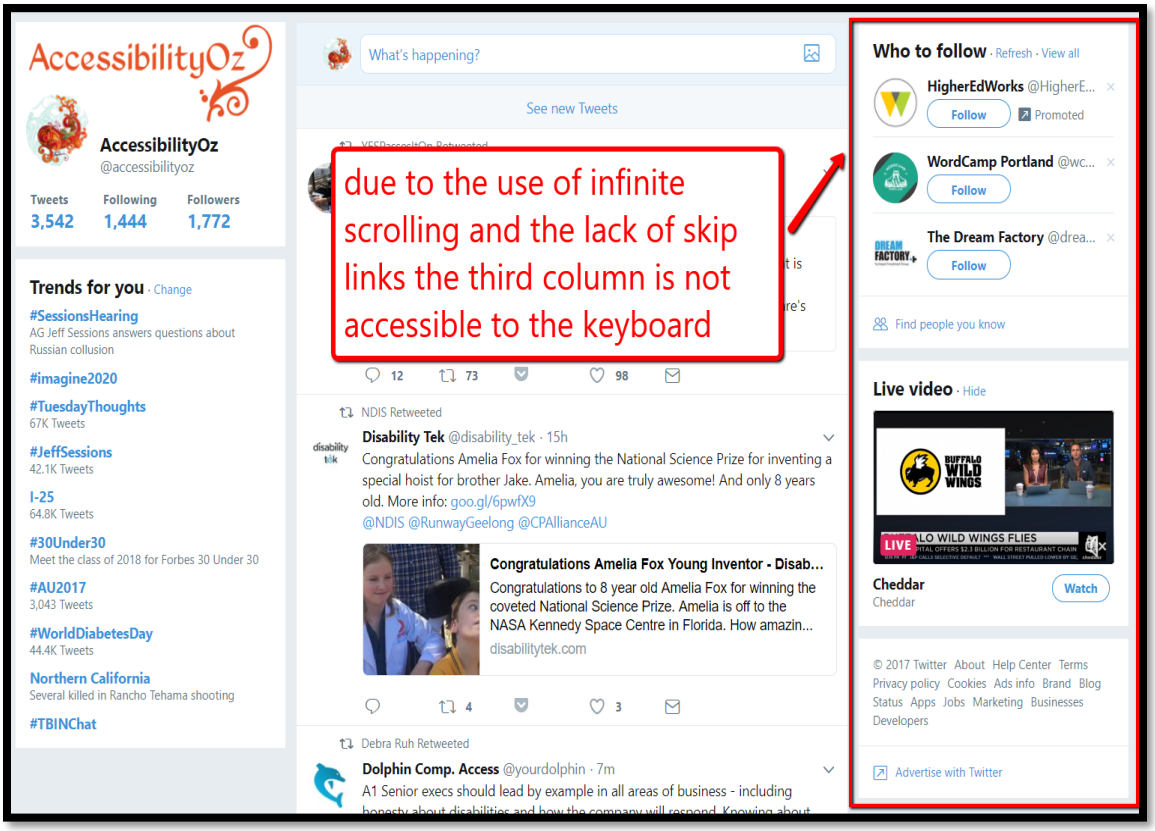


Figure 6 – Due to the inclusion of infinite scrolling, the right-hand column is not accessible to keyboard-only users

Platform-specific accessibility errors

Facebook

It should be noted that the accessibility compliance of Facebook has improved. They have removed the CAPTCHA on signup and have attempted to create automatic text descriptions.

Reverse keyboard trap

When using Facebook with Chrome, Facebook asks if it can send desktop notifications. If the user selects the “Block” option a pop-up appears that cannot be dismissed with the keyboard (see Figure 7). This is a reverse keyboard trap – where an item overlays other content and cannot be dismissed.

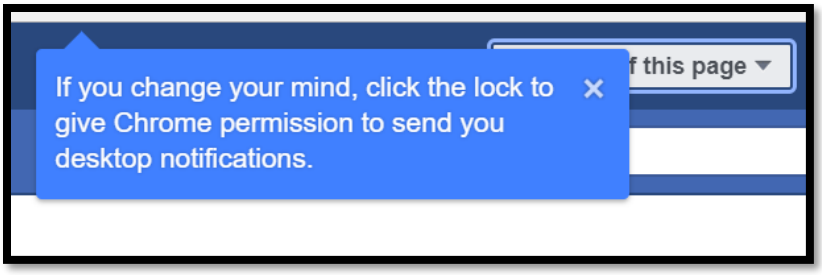


Figure 7 – The pop-up that allows users to turn on desktop notifications is a reverse keyboard trap

This also occurs when using the search (see Figure 8).

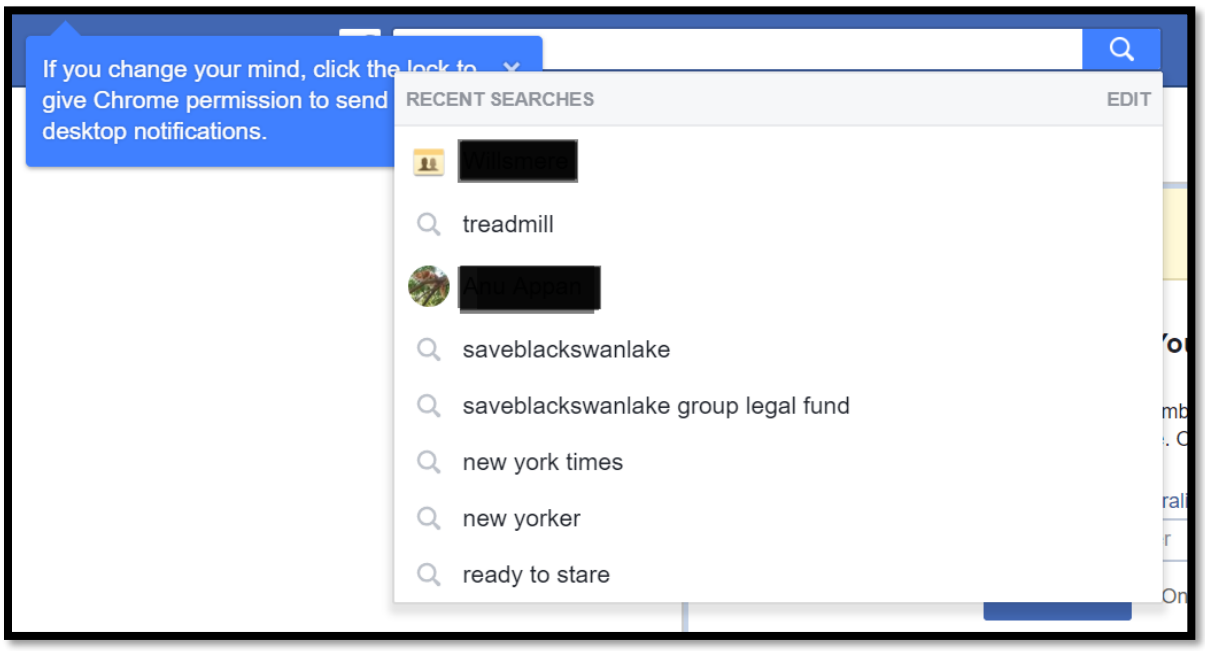


Figure 8 – The drop-down that appears when using the search is also a reverse keyboard trap

YouTube

In past years it has not been possible to dismiss the ads on videos with the keyboard, but this has been successfully addressed. Another issue that was identified in previous testing was the disappearance of the upload icon in the navigation at increased text sizes. This has also been fixed.

Twitter

In previous tests, Twitter always did very well. Unfortunately, with the addition of new functionality, such as videos in tweets, Twitter now has a number of accessibility issues.

Lack of coded field labels

Of most concern is the lack of coded field labels in the login page (see Figure 9). Coded field labels are absolutely essential for screen reader users to use a form.

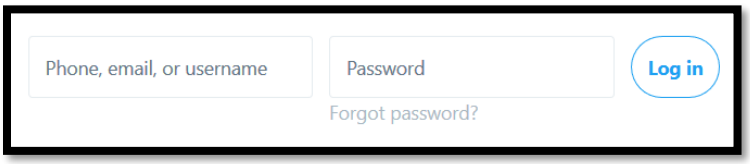


Figure 9 – Login fields for Twitter

The code for the username field is:

```
<input class="text-input email-input js-signin-email"
name="session[username_or_email]" autocomplete="username"
placeholder="Phone, email, or username" type="text">
```

LinkedIn

In previous years LinkedIn scored poorly in accessibility compliance, however there has been a marked improvement in the past six to twelve months. LinkedIn has included skip links for several years, but they are even more comprehensive in 2018.

Button alternatives

In previous years it was not possible to post a status update using the LinkedIn mobile app using VoiceOver on iOS. This was recently fixed. However, one issue remains: the Settings button has an alternative of “button I p 20 4 d p button” (see Figure 10).

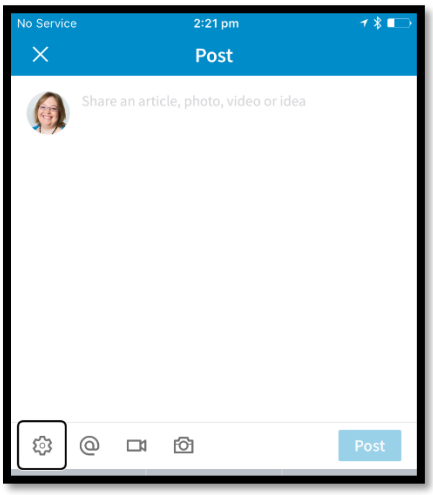


Figure 10 – Setting button does not have an adequate alternative

Conclusion

The use of auto-play, incorrect focus order (on Facebook and YouTube) and infinite scrolling are problems that seem to occur across platforms and means that none of the platforms are fully accessible to a keyboard-only user. These three issues are easily rectified, or alternatives can be provided. For example, providing an easy-to-find function to turn off the auto-play would mitigate videos automatically playing, and some social media platforms provide this. The use of comprehensive, accessible skip links may also mitigate the need to remove infinite scrolling.

Copyright notice

Insights with PowerMapper and R: An exploratory data analysis of U.S. Government website accessibility scans

Sam Kanta, PhD

The Sustainable Change, LLC
1218 N. Falling Water Way
Eagle, ID 83616-5472, USA
sam.kanta@thesustainablechange.com

Abstract

Although State Agencies employ their accessibility standards for electronic and information technologies, States who receive Federal funding are also required to meet WCAG 2.0 AA conformance as per the Section 508 (2017) legislation. Little research exists at scale, as to the degree to which accessibility in States' websites conforms to Federal standards. Automated accessibility scans may form part of the evaluation process in determining the conformance of a website or other ICT to WCAG Guidelines or U.S. legislation. This paper presents the findings of automated web accessibility testing for all 50 U.S. State Government websites. An exploratory data analysis of scan reports indicated extremes in the frequency of non-conformance issues occurring in all State websites, which were capable of remediation without the need for manual testing to confirm their presence.

Introduction

An analysis of U.S. State Government ICT Accessibility policies revealed 27 States provided documentation stipulating the adoption of WCAG 2.0 Guidelines, and the harmonized Section 508 (2017) of U.S. Federal ADA Legislation (Kanta, 2017). Of the remaining State Governments, 15 published documents adhering to the previous version of Section 508 (2000). At the time of the study, 2 States published an independent standard on the basis that meeting WCAG Guidelines to level A was not feasible, and the remaining 3 States lacked specific documentation related to ICT Accessibility practices or standards. From the industry research, questions emerged regarding the current levels of accessibility conformance: to what degree did State websites demonstrate conformance, albeit from the perspective of issues detected programmatically.

Automated testing tools allow accessibility practitioners and researchers the ability to identify non-conformance issues related to WCAG Guidelines. However, the body of research on automated testing within the United States has yet to adequately address the degree to which State Government websites are accessible, especially so soon after the legislative requirement for Federal Agencies, and State Agencies who receive federal funding, to conform to Section 508,

with WCAG 2.0 Guidelines incorporated, as at January 18, 2018 (United States Access Board, 2017). An example of such a tool is PowerMapper, a SaaS suite of scanning functionalities with the objective of detecting issues within web-based code (PowerMapper Software, 2013). The particular feature expressly for detecting potential errors or non-conforming code practices related to accessibility is known as Sort Site - included in the Web Accessibility Evaluation Tools List available via https://www.w3.org/WAI/ER/tools/. The Sort Site User Interface, when used within the cloud version, is indiscernible to the other testing functionality and features, hence the use of the SaaS Product bundle's name for this paper.

Research Methods

A list provided by the U.S. Federal Government, via USA.gov, confirmed State Government URLs. Also, a Google Search occurred for each State website, using the search string "State of (State Name) website" as a way to cross-validate the Federal list. Scans of each State government website occurred over four months (February - May 2018) where data collection focused on the issue reports generated by PowerMapper and results available at the end of each site scan in comma-delimited file format (.CSV).

An *issue*, as defined in PowerMapper, is the presence of a failure to satisfy an accessibility condition (or rule). As of May 2018, 118 rules were applied, addressing 39 checkpoints for WCAG 2.0 accessibility where each rule maps to specific Success Criteria (for specific details, please refer to https://www.powermapper.com/products/sortsite/checks/accessibility-checks/). The Matterhorn 1.02 Protocol was also selected for website scans, enabling the reports to delineate between PDF/UA and content created through scripts and mark-up languages.

Data Analysis

Exploratory Data Analysis (EDA) is a general set of statistical techniques combined with data visualization aimed to understand the characteristics of (often) large data sets (Hartwig and Dearing, 1979). The nature and volume of accessibility reports generated for 50 States made EDA the most appropriate approach to summarize and represent the general trends and comparisons between general concepts, such as WCAG Principles, Guidelines, and Severity Levels and specific performance of individual State Government websites.

The notion of tidy data, attributed to Hadley (2014), is a key part of the analysis process. In order for effective EDA, data is best analyzed when the storage structure is uniform, wherein a matrix each row is considered an observation, and each column a variable, that can be quantitatively captured. The accessibility scan report generated by PowerMapper presented a unique WCAG and/or Matterhorn Protocol potential detected per page. Therefore, an observation row could be considered as the instance of a possible non-conformance issue of a given type and severity.

Before calculating summary statistics, additional variables were created from existing variables from the original report files. The statistical package R Studio, commonly referred to in the literature as "R", was used for the EDA, including the process of tidying the data, and creating multiple data frames (i.e. matrices) for analysis and visualization. With R it was possible to read 50 reports from .csv files, convert to .xlsx, create categorical variables based on URL references to WCAG Guidelines, Techniques, and Failures. The R code was scalable such that analyzing

the data was possible with one report, increasing to all fifty. Irrespective of the data set aggregation; visual plots were capable of being generated at any stage of the process of adding original CSV files.

Findings

The total number of web pages scanned was 24, 373; a geometric mean of 487 pages scanned per website. The mean calculation accounted for the lack of uniformity across websites, explained, for example by the variation across scanned websites, namely, sitemap XML structure, general navigation, and number of pages detectable by PowerMapper.

The issue score was determined by the software calculating the percentage of pages detected in relation to total pages scanned. The reported issue score mean for all State websites was 51%, that is just over half of webpages for a website had one or more accessibility issues detected. PowerMapper aggregated 49,866 issues across 50 State websites grouped into 182 types according to WCAG 2.0 Techniques (*Sufficient*, *Advisory*, and *Failures*).

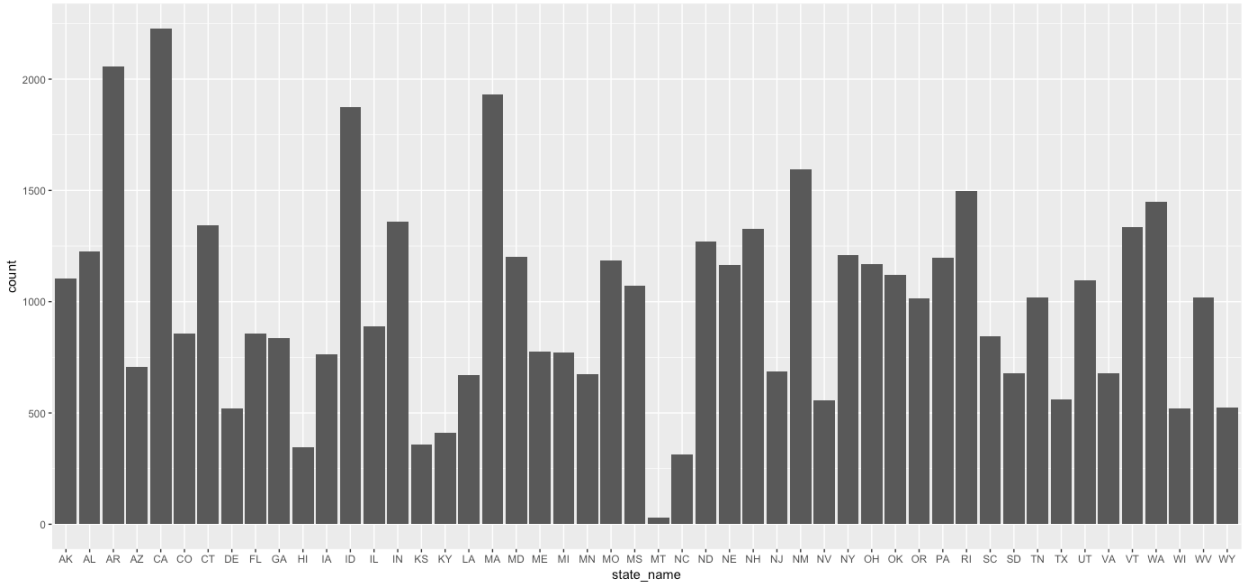


Figure 1 – Total count of Accessibility Issues detected, grouped by State

The summary of total accessibility issues identified across all 50 State Government websites was plotted in R, as shown in Figure 1. The visual representation of tallies reveals a dramatic variation, exemplified when State counts are presented in alphabetical order by State Code abbreviation.

Principles and Guidelines

Reported issues were grouped according to WCAG Principles and provided a summary view of trends. The most predominant Principle in scans were *Robust* (40%), followed by *Operable* (30%), and *Perceivable* (25%). The remaining 5% were categorized as *Understandable*. With the use of R, further analysis of principles was possible at the Guideline level for each issue.

The most recurrent guideline present in all scanned pages were Compatible (n = 18,402) and Navigable (n = 13, 756). The next reported Guidelines: Text Alternatives (n = 5, 210), Adaptable (n = 4, 985), and Distinguishable (n = 3,661). The Keyboard Accessible issue count was in the 10th percentile (n = 1, 126), with the tail of Predictable (n = 116) and Enough Time (n = 110). A single issue was detected for Input Assistance (n = 1). The great variation between reported guidelines detected via PowerMapper was again, through visual plotting, quite evident as shown in Figure 2.

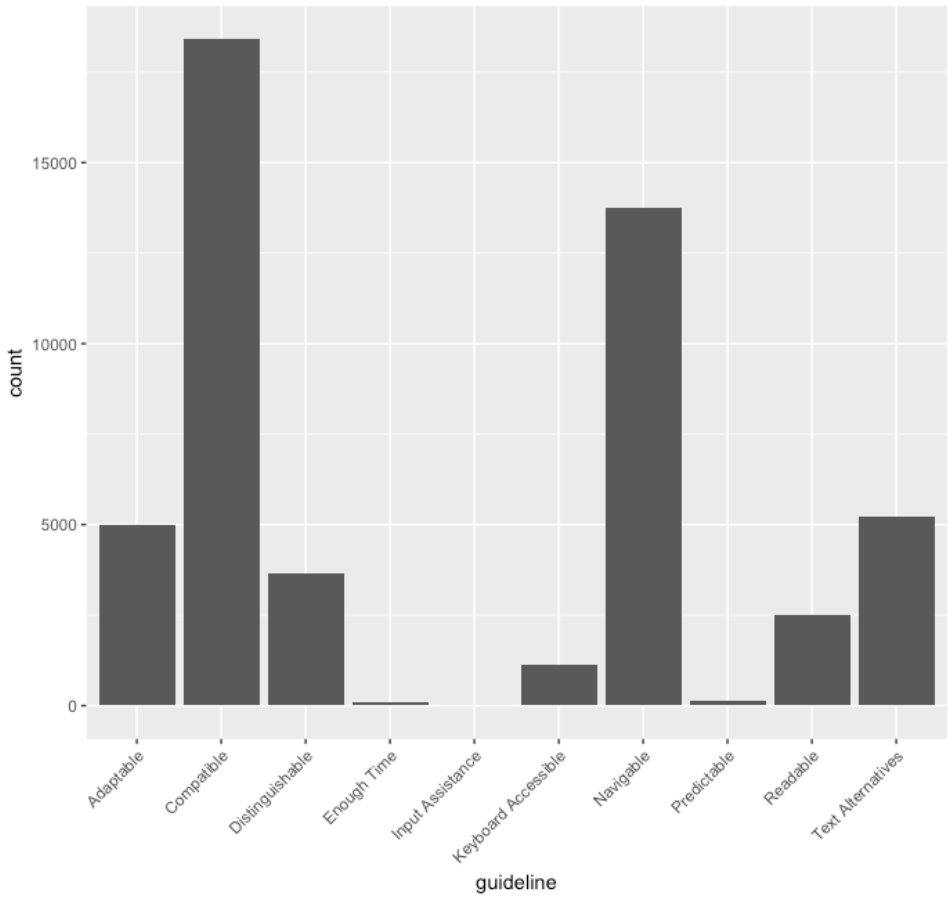


Figure 2 – Accessibility Issues detected, grouped by WCAG 2.0 Guideline

Sufficient and Advisory Techniques

Despite the limited ruleset for accessibility testing, there were There were less than 1% of scan results associated with WCAG 2.0 advisory techniques. The finding is attributed to the human element required in the evaluation process and not sole reliance on automated testing to assess conformance. Several general and HTML techniques were identified in relatively small quantities in comparison to the aggregate total of detected issues and were too few to present in table summary or visual plot.

Failures

The WCAG 2.0 documentation identifies Failures as a subcategory of Techniques. PowerMapper detected errors with specific reference to failures. The subjective nature of Techniques, the least subjective, programmatically speaking, detectable without manual inspection. The most prevalent failure reported was **F70: Failure of Success Criterion 4.1.1 due to incorrect use of start and end tags or attribute markup.** The ten most frequent failures are in the following Table.

Table 1

Ten most frequent WCAG 2.0 Failures reported by PowerMapper (All Websites)

WCAG 2.0 Failure ID	Success Criteria not met	Frequency	% of Reported Issues
F70	4.1.1	7, 783	15.60
F68	4.1.2	5, 763	11.55
F89	2.4.4, 2.4.9, 4.1.2	4, 708	9.43
F77	4.1.1	3, 359	6.73
F65	1.1.1	3, 281	6.57
F91	1.3.1	2, 729	5.47
F25	2.4.2	1, 469	2.94
F49	1.3.2	975	1.95
F30	1.1.1, 1.2.1	804	1.61
F54	2.1.1	760	1.52

One finding that appeared not to follow the general trend of State Issues reported were the frequencies of a given issue for a unique web page. Although States such as CA reported the most detected issues, there was a contrast in the findings at the unique page level. Table 2 presents the ten unique page instances where a specific accessibility guideline issue was detected.

Table 2

Most frequent issues reported for an individual HTML page (identified by State)

State	Reported Issue	WCAG Severity	Frequency
IN	1.4.1	A	500
MS	F65 Matterhorn	A	500
IA	F30 Matterhorn	A	399
ID	4.1.2	A	310
ID	F68	A	310
ND	2.4.6	AA	259
ND	2.4.6	AA	259
KS	G130	AA	253
NH	1.4.4	AA	236
CT	1.4.4	AA	234

Matterhorn Protocol

The Matterhorn Protocol accounted for 8% of issues (n = 3, 827). Of those, the most detected was **11-001: Natural language for text in page content cannot be determined** (n=2,461). The next frequently occurring failure condition reported was related to the absence of Metadata (n=492). The absence of metadata relating to doc titles, tagging of artifacts (mostly images), was the common characteristic of reported issues related to PDF artifacts present within web pages.

WCAG Severity Levels

Initial trends in the data suggest that Level A and Level AA Success Criteria are not met fully by any State Government Website. As an example, one State website tested resulted in 804 reported failures of which 76% were Level A with a recurring accessibility issue of screen readers unable to identify controls, tabular data, images, rendering the experience inaccessible for those reliant on such assistive technology (AT). Similar findings occurred across a range of States, but as of yet, no significant correlations between State Website reported failures were revealed, based on Techniques and Failures for Web Content Accessibility Guidelines 2.0.

A scatterplot representation of Severity A versus AA detected for each State website illustrated several outliers, notably Montana (MT) at the least quantity of A and AA contrasted by California (CA) and Arkansas (AR). Three other clusters were found within the data: a lower-bound x-to-y ratio (NC, HI, KS, KY, WY, WI, DE, TX, NV, SD, ME, IA, AZ, LA, VA, MN, NJ, MI, FL, CO, SC), a middle bound ratio (GA, IL, TN, WV, MS, AK, UT, OR, MO, PA, MD, IN, AL, CT, NE, OH, OK, NH, ND, NY, WA, VT, RI), and an upper-bound ratio cluster (MA, ID, NM).

Based on the analysis, and through data visualization of the A to AA ratios, the prevalence of accessibility identified issues as Level A was clear, as shown in Figure 3.

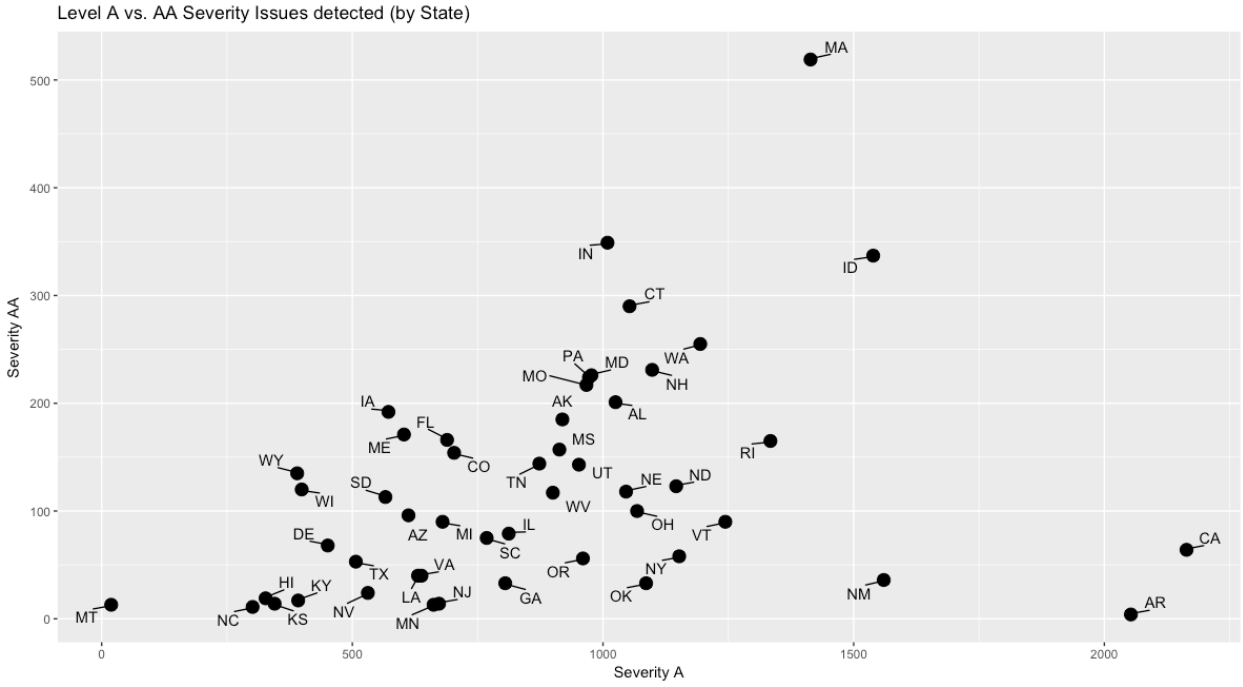


Figure 3 – Scatterplot of Level A versus AA Severity Levels detected for State Government Websites

Discussion

As emphasized throughout accessibility testing research since the mid-2000s, automated testing is not a catch-all methodology, but part of a process that also requires human testing and evaluation (Lazar, Goldstein, and Taylor, 2015). This paper shows the capabilities of Power Mapper in conjunction with statistical analysis undertaken with the software package R and does not make comparisons to other automated testing solutions. The use of R to explore auto-generated reports as those by PowerMapper extends the statistical analysis across multiple websites; aggregate characteristics, and further explore potentially complex correlations, again with the intent of informing future research.

The results show the majority of State Government websites have accessibility issues that do not conform to WCAG 2.0 A or AA Levels, despite the date for conformance to AA having passed (January 18, 2018). Automated test findings by themselves do not make a comprehensive assessment of accessibility conformance yet identify areas for immediate improvement. The findings of this study present opportunities to develop further support and training regarding common issues and remediation strategies. For State Agencies, the insights may also serve to inform strategic roadmaps in attaining and sustaining conformance to Federal requirements for websites to be useful, usable, and accessible.

Limitations

By design, PowerMapper makes a determination based on rules evident in a static page. Any comprehensive evaluation of an accessible website requires a combination of technologies, user test scenarios, and users/testers who reflect the diversity of abilities found in the general population. From a practical perspective, the number of pages scanned by the testing software could not exceed 500 pages; comprising of a primary web domain, and sub-domains if applicable. The selection of pages is determined by the sitemap index, if present, and so the collection of pages to be scanned is out of the control of the researcher. The page and domain limit may potentially skew the comparative analysis of accessibility issues between State websites. Other domains may contain content referenced by a State Agency via hyperlinks.

Overall, PowerMapper provides the ability to assess the degree to which the four web accessibility principles – perceivable, operable, understandable, and robust. On closer inspection, the software evaluates few WCAG Guidelines for certain Principles. For example, 8 out of 9 Guidelines for Time Based Media (1.2) and five of six for Input Assistance (3.3) lack rule sets mapped to WCAG 2.0 Success Criteria.

Future Research

The EDA approach within the R programming framework can be applied to future automated accessibility evaluation studies when comparing variable changes over time. The reproducibility of the study outlined in this paper gives a foundation upon which longitudinal studies can occur. That is, the same R code can be applied to future scans of the same website URLs, and the changes tracked through time-series analysis to assist in understanding what changes occur, if any, about accommodating accessibility within ICTs. The combined capabilities of PowerMapper and R continue to evolve, as too our understanding of automated testing and the need for human perspectives to inform decisions as to what is accessible for all users of public ICTs, such as government websites.

References

Hartwig, F., & Dearing, B. E. (1979). Exploratory data analysis (Vol. 16, Quantitative Applications in the Social Sciences). Beverly Hills (California): Sage.

Kanta, S. (2017). *Accessible Nation: Findings on State Government Section 508 Conformance in the USA.* Internal RESPEC Incorporated report: unpublished.

PDF Association (2014). *Matterhorn Protocol: PDF/UA Conformance Model (Version 1.02).* Retrieved from https://www.pdfa.org/publication/the-matterhorn-protocol-1-02/

PowerMapper Software (2013). PowerMapper OnDemand Suite[Computer Software]. PowerMapper Software, Edinburgh, United Kingdom. Retrieved from https://www.powermapper.com/products/sortsite/ads/acc-accessibility-testing/

RStudio Team (2016). RStudio: Integrated Development Environment for R [Computer Software, version 1.0.143]. RStudio, Inc., Boston, MA. Retrieved from http://www.rstudio.com/

United States Access Board (2017). About the Update of the Section 508 Standards and Section 255 Guidelines for Information and Communication Technology. Retrieved from https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-ict-refresh/overview-of-the-final-rule

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59*(10), 1-23. doi:10.18637/jss.v059.i10

Copyright notice

Validating a Sampling Process for Automated Accessibility Testing of Websites in a National Network

Cyndi Rowland, Ph.D.

WebAIM; National Center on Disability and Access to Education
Utah State University
6801 Old Main Hill
Logan Utah, 84322-6801, USA
Cyndi.Rowland@usu.edu

George Joeckel, M.S.

WebAIM
Utah State University
6801 Old Main Hill
Logan Utah, 84322-6801, USA
George.Joeckel@usu.edu

Abstract

Data should inform decisions when web accessibility evaluators choose to use sampling methods. To this end we performed pilot research using only automated testing to answer the following: “To what extent does a smaller sample identify the same tested accessibility issues present across the larger one?”, and, “How many pages selected at random must be tested to detect 95% of the errors present?” We evaluated 60 websites using WAVE and Dinolytics. We performed Pearson’s *r* correlations between machine-detectable errors in 14-page samples and full sites crawled 6 levels deep. The associations were strong. We also used bootstrapping methods to determine the number of pages needed in a sample to obtain the same result as full sites, with a 95% confidence level. Sample sizes produced by this method did not follow detectable patterns across page or percentage covered by site size. These results may spur important conversations in the field and the need for additional research to tease out moderating and mediating variables.

Introduction

There are a variety of reasons to conduct website accessibility evaluations. Examples include: gathering baselines or benchmarks, setting focus to prioritize improvements, or even confirming or certifying conformance to a set standard. While the rationale for an evaluation will likely drive the methods that are employed, in most cases, automated testing will be utilized to some degree.

Those seeking automated accessibility testing are interested in the most accurate and efficient ways to gather this information. They are keenly aware that this method does not identify all accessibility issues. Yet, it can quickly enumerate those that are programmatically-detectable, reducing the human effort needed to evaluate overall conformance. The question then becomes the scale and size of this machine testing.

Determining the appropriate scope for any machine testing is a fundamental task. As the costs to conduct automated scans of pages continues to decrease, performing a scan of every page on a site may or may not be prohibitive financially. However, it is equally important to recognize that for some evaluation contexts, an analysis of the entire population of URLs may not be feasible.

While scanning every page provides information on all programmatically-detectable errors in every location, this can be expensive and sometimes time-consuming. Scanning a sample reduces time and cost, at the potential cost of accuracy. Having confidence in the outcomes from sampling requires that one assumes any sample is representative of the broader population of pages from the entire website.

Proper sampling is something discussed within the field of accessibility. Yet recommendations have not been adopted that influence what is done in practice. Systematic and random sampling are statistical foundations commonly used. But what does this look like on the web?

For example, the W3C's Website Accessibility Conformance Evaluation methodology (WCAG-EM,1.0; http://www.w3.org/TR/2014/NOTE-WCAG-EM-20140710/) proposes ad hoc and random sampling to obtain a core set of pages that represent the content and functionality of the site. They recommend that the size of the sample is based on 10% of what is termed the "Structured Sample" (https://www.w3.org/TR/WCAG-EM/#step3). The structured sample includes common web pages, essential functionality, types of web pages and web technologies, and other relevant pages. It is acknowledged that a single page may represent several items desired in the structured sample. These collections are then randomly sampled to the 10% level; this is not 10% of the site, rather it is 10% of the Core Structured Sample that was gathered. There is no guidance on the number of pages in a sample, however it is acknowledged that larger sites will have a larger sample: "During this step the evaluator selects a sample of web pages and web page states that is representative of the target website to be evaluated. The purpose of this selection is to ensure that the evaluation results reflect the accessibility performance of the website with reasonable confidence" (W3C Working Group Note, 2018).

The Unified Web Evaluation method (UWEM 1.2 Core; http://www.wabcluster.org/uwem1_2/UWEM_1_2_CORE.pdf) recommends random sampling from a Core Resource List (e.g., homepage, contact, help, site map, and examples of web technologies used such as forms, data tables, multimedia, and client-side scripting). Their recommendation for the minimum size of the sample is 30 unique pages if available. They also recommend adding 2 additional unique pages per 1000, up to a maximum of 50 resources in the evaluation sample.

Practitioners are using similar sampling methods by combining a number of pages for the sample based on the size and complexity of the site along with components, features, and technologies used across web pages. For example, both WebAIM and Interactive Accessibility recommend

that if you are going to use samples, that you find pages representative of the templates, functions, content, technologies, and interaction available on the site. Wahlbin (n.d.) indicated that in general terms, small to medium sites can be performed with a sample of around 10-15 pages and larger sites benefit from samples of 20-25. WebAIM samples typically range from 15 to 25 pages depending on how many it takes to get a representative sample.

We wondered how confident one could be in using a randomly-derived small sample as the basis for a web accessibility evaluation; and What is the degree to which the sample represented the entire site? Velleman and Geest (2013) had asked the question, "How many pages is enough?" In their pilot study, they found that a core set which included the home page plus 13 randomly-selected pages provided 93% of the error types found more broadly on the site. While they performed human evaluations against WCAG 1.0, their results spurred a number of questions we believe we could begin to address using automated testing results.

WebAIM recently partnered with a web development company to create an enterprise-level accessibility reporting system based on our proprietary web evaluation application WAVE (http://wave.webaim.org). The product, Dinolytics (https://dinolytics.com/), contains spidering/crawling functions that put us in a position to create, and test large data sets that would be necessary for this type of research. We were now in a position to collect full site information using Dinolytics, and then compare this against the smaller, 14-page samples, as found in the work of Velleman and Geest (2013).

Thus our research questions became:

1. To what extent does the smaller sample identify the same tested accessibility issues present across the larger one?
2. How many pages selected at random must be tested (in addition to the home page) to detect 95% of the errors present?

Method

During the summer of 2018, we gathered two data sets from a national network of websites belonging to University Centers for Excellence in Developmental Disabilities Education, Research, and Service (UCEDD; https://www.aucd.org/directory/directory.cfm?program=UCEDD), which are funded through the Federal Administration on Community Living. At least one UCEDD can be found in every U.S. state and territory, mostly at research universities and medical schools. Currently there are 67. Each website was located from the UCEDD Directory (AUCD, n.d)

We chose these sites specifically because it is reasonable to believe that they should be addressing accessibility, and as such, might provide a more sensitive measure than would result from sites where web accessibility had not been a consideration. This is because when error rates are high, there is a higher probability of finding the same errors from a smaller number of pages, thereby increasing correlation coefficients and confidence levels.

The first data set was the smaller sample. We identified the home page of each UCEDD. Then, we used a free web crawler (https://robhammond.co/tools/seo-crawler), (Hammond, n.d.) to

collect a pool of pages. We used it to identify up to 250 URLs (the maximum allowed for the free service) under each UCEDD's domain. After manually eliminating non-HTML pages, we used a random number generator to pick 13 pages. Finally, we replaced any pages WAVE could not analyze in that set by manually picking another URL from the pool at random.

The second data set was the full site. We used Dinolytics, a tool that combines the WAVE engine with spidering capabilities. We set the maximum number of pages to return at 20,000 under each UCEDD domain; sufficient to gather all valid URLs under each domain, six levels deep. Dynamically-generated web pages greatly increased the number of pages returned by our crawler. In one case, fewer than 2% of the over 18,000 pages gathered were static. Through a manual sorting process, we eliminated all but one of each unique type of dynamic page found in each site.

Next, we used WAVE to detect 17 programmatically-determined errors, mapped to seven WCAG 2.0 Success Criteria across both the small sample and the full site. This mapping is found on Table 1. The presence of a single error in any Success Criteria Error Group was recorded as a "fail" of that group for the page. These page failures by error group are the basis of our data for both data sets.

Our UCEDD data is based on information from 60 of the 67 sites. Seven sites were eliminated because they either had an insufficient number of pages to reach our 14-page sample, or there were technical reasons that prohibited us from crawling the site. The full sites themselves were quite varied, with a range of 24 to 10,948 pages. The mean of full site pages was 644.

Analysis

We analyzed our data in two main thrusts, each to answer a different research question. In the first, we looked at the relationship between machine-discoverable errors (i.e., the WAVE errors mapped to the WCAG success criteria) detected in the small sample with those found in the full site. This correlation coefficient was calculated using Pearson's *r*, a simple correlation of the linear relationship between two variables.

For the second analysis, we sought to determine how many pages it would take to detect machine-discoverable errors present across the full site at a 95% confidence level. To do this we looked at failures of the Success Criteria Error Groups in both samples on each page. To that end, we used the bootstrapping method documented by Efron & Tibshirani (1994). We randomized the order of each website's URLs and recorded the number of pages needed to discover at least one instance of each Success Criteria Error Group found in the analysis of all the website's URLs. We repeated this process for 100 replications to generate a set of page counts. Next, each set was sorted by page count in descending order. To determine a 95% confidence level, we eliminated the top five page counts. Count number 6 represented the minimum number of pages needed in order to be 95% confident that the smaller sample would match the errors in the full site. We then calculated means of these counts to represent what would be needed for different sizes of sites and for the entire sample.

Table 1: WAVE Errors Mapped to WCAG Success Criteria

WAVE error	WCAG Success Criteria
Image alternative text is not present.	**1.1.1**
An image without alternative text results in an empty link.	**1.1.1; 2.4.4**
Alternative text is not present for a form image button.	**1.1.1; 2.4.4**
An image that has hot spots does not have an alternative text.	**1.1.1; 2.4.4**
Alternative text is not present for an image map area (hot spot).	**1.1.1**
A button is empty or has no value.	**1.1.1; 2.4.4**
A form control does not have a corresponding label.	**1.1.1; 1.3.1**
A form label is present, but does not contain any content.	**1.1.1; 1.3.1**
A form control has more than one label associated with it.	**1.1.1; 1.3.1**
A heading contains no content.	**2.4.6**
An `aria-labelledby` or `aria-describedby` reference exists, but the target for the reference does not exist.	**1.1.1; 1.3.1**
A `<th>` (table header) contains no text.	**1.1.1; 1.3.1**
Blinking content is present.	**2.2.2**
A `<marquee>` element is present.	**2.2.2**
The page title is missing or not descriptive.	**2.4.2**
A link contains no text.	**2.4.4**
The language of the document is not identified.	**3.1.1**

Results

Question #1: To what extent does the smaller sample identify the same tested accessibility issues present across the larger one?

Table 2 shows strong correlation coefficients between the errors detected in the 14-page sample and the full site. Across all 60 sites in the UCEDD sample, the *r* was .57. Given all of the variables present, and calculating the R-squared, this accounts for 32% of the variance in the dependent variable (i.e., the full site), which is notable. We then became interested in determining if these relationships would be affected by the size of websites. There are certainly reasons to suspect that when you are looking at a smaller set of web pages, this relationship would be higher because your sample accounts for a greater percentage of the overall site. This was exactly what we found. Sites under 500 pages revealed a strong *r* of over .7. In these instances, it controls for over 50% of the variance in the sample (i.e., 50% for the smallest sites and 57% for sites up to 500 pages). Above 501 pages this correlation drops significantly. However, the result might be expected when your sample becomes a small proportion of the total static pages on a site.

Table 2: Pearson Correlation Coefficients

	All sites	**14-200 pp**	**201-500 pp**	**Above 501 pp**
n of sites =	60	28	14	18
***r* =**	.57	.76	.71	.28

Question #2: How many pages selected at random must be tested (in addition to the home page) to detect 95% of the errors present?

Performing the bootstrapping procedure on all UCEDD websites in our sample revealed that if you want 95% confidence that your sample detected all machine-discoverable errors, you need to go well beyond a small 14-page sample set. In fact, the average "small" sample needed across the all sites would need to contain 232 pages. Again, we were interested in looking at the contribution of site size to our confidence interval. Table 3 displays data for sites of different sizes. It shows both the average number of pages you would need to have in your sample to be 95% confident that you can detect all the errors, along with the average proportion that these pages represent in the full site.

Table 3: Pages Required in a Sample to Achieve 95% of Errors in a Full Site

	All sites	14-200 pp	201-500 pp	501-1000 pp	Above 1001 pp
n of sites =	60	28	14	10	8
Average number of pages needed for the sample	232	41	68	413	959
Average percentage of the site in the sample	42%	49%	23%	63%	27%

If there are patterns in these data, they are not clear. While the page numbers needed in a sample grows as the site itself grows, this is not the case when you consider the average proportion of pages the sample represents. For example, looking at categories of sites below 500 pages in this study, we would want to select 41 and 68 pages respectively to have a 95% confidence that we would find all machine-detectable errors in the full site. Our site between 501 to 1,000 pages, would require a sample of 413 pages, which is more than 6 times larger than the preceding site size of 201-500 pages. Also, the average percentage of pages across site sizes do not appear to display a pattern. One might expect this average proportion to be flat, or even grow, or decrease in a fairly linear way with the size of the site. More work needs to occur to tease out what are likely mediating or moderating variables that affect this phenomenon. Of course, it is also possible that this reveals issues in measurement that are simply not understood at this time.

Summary

The authors wanted to add data to ongoing conversations in the accessibility evaluation field. While these data are preliminary, it is our hope that discussions will lead to new hypotheses that can be tested and shared. More research is needed to develop accurate and efficient professional practices to create and use samples that will result in web accessibility evaluations.

Future researchers should both replicate and extend what we have done here to determine if our observations hold, as well as to tease out mediating and moderating variables.

The following are a few examples of what we consider to be needed research:

1. More needs to be done to explain the variability we found in the confidence we have using samples of different page counts—and proportions—with different site sizes. We need to detail variables that are affected by site size so we may understand if sampling methods should vary as a function of the size of the site itself. Perhaps the largest sites are those with the greatest use of certain features (e.g., templates that are easily pulled into random samples)? Or perhaps larger sites are the most varied and complex which complicates assumptions required to use sampling methods?
2. It may be important to look at error types themselves. We know that some success criteria are easier to implement than others. For example, are there certain error types that may not be detected at specific confidence levels? What if we could detect all errors at a 90% confidence level with a sample of 28, but to get to a 95% level of confidence, we needed

to triple the page sample? What if this was because of one feature that occurs infrequently across the site? If so, what features commonly fall into this category, and are there other ways we could detect them so that we could use smaller samples and look for specific feature implementation separately? Furthermore, could those features be automatically detected so that they may be more quickly found and evaluated, even if by human assessment?

3. We need to explore the relationship between the overall accessibility of the site and confidence of our sample size. It is hypothesized that the less a site conforms to accessibility standards, the more confidence we will have that we detect errors from a smaller sample. If true, as accessibility is implemented, fuller site detection strategies may be required and confidence levels may only remain if the sample size increases. This would then merge a maturity model of working on site accessibility with a maturity model for evaluation of the same.
4. If large sample site sizes (e.g., over 80) are also required to contain varied and complex components (e.g., all templates, all web technologies, essential functionality), the field may be interested in a cost study to determine the effort needed to assemble a proper sample? What is the time and cost to perform the work to get to a solid sample that will lead to high levels of confidence? It could be, given the speed of machine evaluation, that a full site could undergo complete machine evaluation before a proper sample is even determined. Of course, this does not address those items that must be evaluated by human assessment.
5. Finally, while we were only looking at machine-discoverable errors in this study, the field will benefit when someone conducts similar research to explore all issues of accessibility conformance, including those where human assessment is required. While this will be complicated by the need to manually evaluate some aspects of every page on a site, it will be necessary to inform evaluation practitioners of proper sampling procedures.

References

Association of University Centers on Disability. (n.d.). Directory. Retrieved September 22, 2018, from https://www.aucd.org/directory/directory.cfm?program=UCEDD

Dinolytics | Enterprise Web Accessibility Evaluation. (n.d.). Retrieved July/August, 2018, from https://dinolytics.com/

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

Hammond, R. (n.d.). SEO Crawler. Retrieved July/August, 2018, from https://robhammond.co/tools/seo-crawler

Velleman, E. & van der Geest, T. (2013). Page sample size in web accessibility testing: how many pages is enough?. *In Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*. ACM, New York, NY, USA, , Article 61 , 2 pages. DOI: http://dx.doi.org/10.1145/2513383.2513408

UWEM, Unified Web Evaluation Methodology Version 1.2. (n.d.) Retrieved July/August from http://www.wabcluster.org/uwem1_2/

WCAG-EM, Website Accessibility Conformance Evaluation Methodology(WCAG-EM) 1.0. Working Group Note 10 July 2014. Retrieved July/August, 2018 from http://www.w3.org/TR/WCAG-EM/

WCAG-EM, Website Accessibility Conformance Evaluation Methodology (WCAG-EM) 1.0. Step #3 (n.d.). Retrieved July/August, 2018, from https://www.w3.org/TR/WCAG-EM/#step3

Wahlbin, K. (n.d.). Interactive Accessibility: Selecting Representative Pages. Retrieved September 20, 2018, from http://www.interactiveaccessibility.com/selecting-representative-pages

WebAIM (Ed.). (n.d.). WAVE: Web Accessibility Evaluation Tool. Retrieved June/July, 2018, from http://wave.webaim.org/

Copyright notice

Using a Component-First Approach in Front End Development and Accessibility Testing

Drew H. Bollinger

RedShelf
500 N. Dearborn St. Ste 1200
Chicago, Il 60650 USA
howie@redshelf.com

Abstract

Compliance with accessibility guidelines like Web Content Accessibility Guidelines (WCAG) or Section 508 is often a confusing and overwhelming goal within the software development lifecycle. However, these guidelines are nothing more than sets of rules with explicit success criteria that define how HTML should be rendered in a web page. This makes accessible guidelines an excellent candidate for automated testing. Using a component-first development approach and existing open source tools and testing practices, software teams can easily integrate WCAG and Section 508 compliance into their existing testing processes.

Front End Web Frameworks

Asking a development team to build good software without relying on a shared set of tools and conventions would be like asking a kitchen staff to create their own cooking equipment, hook up plumbing and gas lines, and create new recipes each time they need to prepare a meal. Fortunately, modern web software can be built on mature frameworks that enable teams to build better software more efficiently.

Frameworks and libraries like Ember.js, React.js, Angular.js, and Vue.js offer development teams a way to build software using a set of existing standards and patterns. Depending on what type of application is being built and what resources are available, each modern framework has its own list of pros and cons, but this paper will reference the Ember.js framework for consistency. With that said, it's worth noting that each framework has its own tools and conventions for many of the topics discussed. Most importantly, each framework has methods of testing the components that make up an application.

Components

A fundamental concept in many modern web frameworks is that of a component. A single web page or a web application is constructed by combining a number of smaller or single purpose components to deliver larger features and functionality. For example, the comments section of a blog page might be made up of a component that renders existing comments, and a component

that allows a user to add a new comment. While the two components are related, they can operate independently and do not rely on one another to operate correctly.

In Ember, a component is comprised of a template file and a code file. The template file is a combination of HTML and Handlebars syntax that is rendered anywhere the component is used. The component file is all of the javascript logic, properties, and data required for the component. A common practice is to separate the data retrieval from the component that displays that data, keeping the purpose of the component constrained to rendering the HTML and handling interactions from the user.

Consider a trivial example in which a product owner might ask for a page that displays a list of online users. A component that fulfills this need might have an attribute named `onlineUsers` that can be iterated in its template, rendering each user's name. The template has access to the properties of the component file, and can invoke them using the handlebars syntax, i.e. `{{ onlineUsers }}`.

Online Users Component File

```
//online-users-list.js

export default Component.extend({

     onlineUsers: [ ] //an empty list of users

})
```

Online Users Template File

```
//online-users-list.hbs

<div>Online users</div>

<ul>

     {{#each onlineUsers as |user|}} <!-- iterate the list of
users -->

     <li>

     <div>{{user.name}}</div>

     </li>

     {{/each}}

</ul>
```

It may be helpful to consider the following key pieces that make up the interface of an ember component:

Parent - The "scope" that renders the component. A parent can render multiple components and perform logic and operations that its children don't need to know about.

Attributes - The information the component needs to receive from its parent or other sources.

When the parent invokes the `online-users-list` component, it needs to provide the `onlineUsers` attribute:

Parent Template File

```
//parent.hbs

{{online-users-list onlineUsers=aListOfUsers}}
```

Reviewing this example, it might already be apparent how automated testing can benefit from components with such narrow focus. In addition to the component file (`online-users-list.js`) and the template file (`online-users-list.hbs`), Ember encourages a third file, `online-users-list-test.js`. As the name implies, it is a test file that tests only the scope of the component to which it relates.

Testing Components

Components are a great way to isolate functionality, allowing multiple team members to develop different parts of a larger interface in tandem. However, an actual feature is only ever as good as its weakest link. To mitigate this concern, front end frameworks provide tools to test components in completely isolated environments, ensuring that each component fulfills its described purpose.

When a component is tested in a modern front end framework, the component is actually rendered in browser, completely isolated from any actual application. The test makes assertions that the component, as a standalone piece of functionality, correctly renders all of the expected output, and responds correctly to specific user interactions. Any future developers who need to use the component in other places throughout the application can be certain that it will work as expected. It is common to automatically run all tests for a given project any time changes are made or new code is added to the project. This practice is called continuous integration, and can alert developers if their changes have caused existing tests to fail.

Revisiting the `online-users` component, a test that follows ember's testing syntax and conventions (which, by default, is built on top of the popular QUnit test framework, but can be configured to use other frameworks) might look like:

Online Users Test File

```
//online-users-list-test.js (simplified for readability)

test('it renders the list of online users', {
```

```
    render({{online-users-list onlineUsers=[Sue, Sam,
Syd]}};

    ok(find('li:contains('Sue'), 'assert Sue is rendered');

    ok(find('li:contains('Sam'), 'assert Sam is rendered');

    ok(find('li:contains('Syd'), 'assert Syd is rendered');

});
```

When the browser loads and runs `online-users-list-test`, it loads the component and template files and renders an instance of the component in the browser using the `onlineUsers` attribute provided from the test's context. It then searches the document for a list element containing each of the names passed into the component, and finally makes an assertion that it successfully found the element. Should a developer make changes that prevent successful assertions, the test would no longer pass.

Components *within* Components

As the complexity of a component grows, it is useful to break it down into smaller pieces. Consider the previous component example (a list of online users) — experienced developers may see an opportunity to break it down into two components:

A child component that represents a single user

A parent component that renders a list the `onlineUsers` list using the child component

The value of such explicit separation is notable. The component for a single user can now be used anywhere a user needs to be rendered; it isn't locked into being rendered in the `online-users` component. The template file for the parent component just needs some minor updates, and a new template can be created for the single user detail:

Online User List Template File

```
//online-users-list.hbs

<div>Online users</div>

<ul>

    {{#each onlineUsers as |user|}}

    <li>

    {{user-detail user=user}} <!-- render component for each
user -->
```

```
    </li>

    {{/each}}

</ul>
```

User Detail Template File

```
//user-detail.hbs

<div>{{user.name}}</div>
```

As complexity is broken down into smaller components, it becomes possible to isolate functionality and responsibility of code throughout the application. Considering the components listed above, the user detail component only needs to know about a single user, and is only responsible for rendering a user's name. It doesn't matter where that user came from or how the parent provided it: it's going to render it. A new test named `user-detail-test.js` can assert that the component is rendered on the page correctly:

User Detail Test File

```
//user-detail—test.js (simplified for readability)

test('it renders the name of the user', {

    render({{user-detail user=Sue}});

    ok(find('div:contains('Sue'), 'assert Sue is rendered');

});
```

Compared to the original test for online-users component, the scope has been dramatically reduced, which has a number of benefits. Errors become simpler to debug, it's easier to make changes, and it's easier to document expected behavior. The value of this approach is especially noticeable when incorporating testing for accessibility and WCAG compliance.

Testing Components For Accessibility

Most of the conventions and methods described in this paper are common practices that many software teams adopt in some form or another. This means that for many teams, integrating accessibility testing into an existing project is simply a matter of leveraging an existing test suite.

Consider the `user-detail` component — perhaps after releasing it, the product owner asks to add a user avatar next to the user's name. Assuming the user data being passed to the component now includes an avatar image, the original template for `user-detail` that contained

```
    <div>{{user.name}}</div>
```

May suddenly look something like contain

```
<img src={{user.avatar}} />

<div>{{user.name}}</div>
```

Somebody with knowledge of WCAG rules will instantly recognize that this template fails to comply with WCAG rule H37, “Using alt attributes on img elements.” Historically, issues like this might not be caught until a dedicated QA phase of development, or until well after the change is released to production and a user reports the issue to customer service. A developer could add this component to a number of pages throughout the entire application, unknowingly adding new accessibility violations with each use.

A natural inclination for developer with WCAG experience might be to add an explicit check in the existing test for the component:

User Detail Test File (Test img Element)

```
//user-detail—test.js (simplified for readability)

test('it renders a user with name and avatar', {

     render({{user-detail user=SueWithNameAndAvatar}});

     ok(find('div:contains('Sue'), 'assert Sue is rendered');

     ok(find('img').getAttribute('alt'), 'assert alt tag is
populated');

});
```

The test now renders an instance of the user detail component with an object that represents the user, which includes a name and an image source. When the browser runs `user-detail-test` with these changes, it makes an additional assertion to verify that the `img` element’s `alt` tag is populated. If it isn’t, like in the updated example above, the test will fail, and the developer can take the appropriate action to address the issue.

Of course, adding explicit checks to every element in every component for every rule would get quite cumbersome and difficult to maintain. To avoid this, various organizations have built tools to help abstract and simplify the process. Tools like axe from Deque (https://www.deque.com/axe/), and the open source project ember-a11y-testing (https://github.com/ember-a11y/ember-a11y-testing) do just that.

axe and ember-a11y-testing

Deque's own website states that

> "Axe is an open source rules library for accessibility testing. It was developed to empower developers to take automated accessibility testing into their own hands and to avoid common pitfalls of other automated accessibility tools."
> — Deque (https://www.deque.com/axe/)

And it is certainly the case; it is a powerful engine that other open source communities have embraced and used to build useful tools for front end frameworks. Ember's axe integration is called ember-a11y-testing, and it gives developers some helpful javascript methods that can be called inside a component test, which then runs the HTML of the rendered component against the axe accessibility engine, returning any errors it discovers. When errors are returned, the test fails, instantly informing the developer about the violation, and even includes helpful links to Deque's documentation with more information about the violation.

After including ember-a11y-testing into a project, the previous test that asserts the presence of an alt tag can be updated to make a single call that uses the axe accessibility engine to evaluate the all of the component's HTML against all the accessibility guidelines it supports. The test now becomes:

User Detail Test File (With ember-a11y-testing)

```
//user-detail—test.js (simplified for readability)

import a11yAudit from 'ember-a11y-testing/test-
support/audit';

test('it renders a user with name and avatar', {

     render({{user-detail user=SueWithNameAndAvatar}});

     ok(find('div:contains('Sue'), 'assert Sue is rendered');

     a11yAudit(this.element); //check for accessibility
violations

});
```

When the browser runs `user-detail-test` using the ember-a11y-testing integration, it passes all the rendered HTML from the component to `a11yAudit` function by referencing `this.element`. If the audit is successful, the test continues to make its final assertion and the test passes. If the test fails, the developer is instantly provided with details about the accessibility violation, in this case, a link to Deque's documentation about the img alt tag rule (https://dequeuniversity.com/rules/axe/2.6/image-alt).

This type of fast feedback is useful for developers who are new to accessibility rules and best practices. Consider the original list component; perhaps a developer with some new knowledge of aria attributes decides it may be useful to associate the `<ul>` element with the list's name using an `aria-labelledby` attribute, but makes a simple mistake:

Online User List Template File With Aria Errors

```
//online-users-list.hbs (truncated for readability)

<div class='online-users-list-label'>Online users</div>

<ul aria-labelledby='online-users-list-label'>

     . . .

</ul>
```

The example erroneously tries to make the aria association using a class name instead of an id attribute. A component test using ember-a11y-testing will catch this mistake and alert the developer:

Online Users List Test File (With ember-a11y-testing)

```
//user-detail—test.js (simplified for readability)

import a11yAudit from 'ember-a11y-testing/test-
support/audit';

test('it renders the list of online users', {

     render({{online-users-list onlineUsers=[Sue, Sam,
Syd]}};

     . . .

     a11yAudit(this.element); //check for accessibility
violations

});
```

As a final example, perhaps some time later another developer decides to replace the `<li>` elements in the `<ul>` with `<div>` elements. With ember-a11y-testing implemented, this accessibility violation would also be caught and the developer could immediately take corrective action and revert the changes. The process of using the axe engine through these open source libraries is both simple and quick, and gives developers confidence that they are writing code that meets accessibility guidelines.

Conclusion

Testing conventions in modern web frameworks demonstrate how easy it can be to prevent introducing new accessibility violations into an existing codebase, and how developers can take advantage of existing processes to become advocates for better and more accessible software. It provides a passive yet effective way to educate developers on accessibility standards and best practices. When entire development teams adopt testing processes that incorporate accessibility audits, it creates an ongoing commitment to building and maintaining accessible software.

References

E. (2018, June 05). Ember-a11y/ember-a11y-testing. Retrieved from https://github.com/ember-a11y/ember-a11y-testing

H37: Using alt attributes on img elements. (n.d.). Retrieved from https://www.w3.org/TR/2016/NOTE-WCAG20-TECHS-20161007/H37

Axe. (Deque). Retrieved from https://www.deque.com/axe/

Copyright notice

2018 Update on W3C/WAI Accessibility Conformance Testing (ACT) for WCAG

Shadi Abou-Zahra

W3C Web Accessibility Initiative (WAI)
2004, Route des Lucioles BP94, 06902 Sophia-Antipolis, France
shadi@w3.org

Abstract

Coinciding with the first ICT Accessibility Testing Symposium in 2016, W3C launched work on WCAG Accessibility Conformance Testing (ACT). This new work was announced during that symposium and received positive feedback. By 2017, it was at the stage of consensus building to align many different approaches and perspectives. We held a panel with active debate among the participants, which highlighted some of the present issues. Meanwhile the group addressed the severe issues, and in the coming weeks, the group expects to enter "Candidate Recommendation" stage of the W3C development process. This means the work is ready for implementation testing and early adoption. At this year's ICT Accessibility Testing Symposium we will demonstrate the first set of internationally harmonized ACT Rules, along with implementations of these rules. We will show how these ACT Rules work in automated tools and manual methodologies, and the benefits they provide throughout the testing process.

Background

Evaluating the conformance of web content – including dynamic web and mobile applications – to the W3C/WAI Web Content Accessibility Guidelines (WCAG)[(1)]can be a non-trivial task. In particular, some WCAG Success Criteria are broad and need to be broken-down systematically for evaluation, or they require qualitative analysis within the specific context of the web content being evaluated. Thus, varying interpretations are manifested in the evaluation tools and testing methodologies, with often conflicting results. That is, the same web content might be deemed to have 'passed' accessibility requirements by one method, yet 'failed' by another. This contributes to confusion within the field. In some cases, it also leads to loss of business opportunities and legal disputes, which is not supportive to the cause of accessibility. It is thus critical to pursue a common understanding of WCAG, as well as to harmonize the different interpretations in the context of accessibility conformance testing and evaluation practices.

For many years, researchers, tool developers, and individual experts have attempted to address this issue. Several initiatives and research projects have been undertaken in Europe, the United States, and elsewhere. These resulted in a number of different testing approaches, each with their own collection of advantages and disadvantages. However, until recently these activities did not seem to get a lot of traction among different stakeholders. Yet with the growing uptake of web accessibility standards, there is an increased need for common testing approaches within the

field. In fact, some organizations requested that the W3C Web Accessibility Initiative (WAI) undertake standardization activity in this area, to provide a vendor-neutral, authoritative, and openly available interpretation of WCAG for conformance testing.

Previous Work

Work on tool-supported web accessibility evaluation started soon after publication of WCAG 1.0 in 1999. Early attempts to this included the meanwhile obsoleted "Techniques for Accessibility Evaluation and Repair Tools"[2], which flowed into the "Techniques for WCAG 2.0"[3]. In 2005 the European Commission funded three research projects to develop the "Unified Web Evaluation Methodology"[4]. This also contributed to the development of "WCAG 2.0 Test Samples"[5], to help benchmark the accuracy of web accessibility evaluation tools.

Numerous developments have also taken place outside of the W3C. This includes the test rules development of the "Open Ajax Alliance"[6], "aXe"[7], and many more, some of which may not be publicly available. Thus, a goal of this new effort is to build on and merge these activities rather than to create another separate approach. The central role of the W3C as the leading body in web accessibility standardization is critical to this purpose.

Current Work

More recently work started in the W3C Community Group "Automated WCAG Monitoring (auto-WCAG)"[8]. Community groups are pre-standardization fora with no formal standing in W3C status. Anyone can join such groups and they serve as important vehicles to collect input from the community and incubate new work. Despite its name, the work of this community group is not limited to automated testing only, but also explores semi-automated and manual testing. The work of this group, including a list of testing procedures, is publicly available[9].

Based on this initial work, representatives of W3C member organizations proposed the creation of a task force as part of the W3C Accessibility Guidelines Working Group (AGWG), which is responsible for the development and maintenance of WCAG. The purpose of this new "WCAG Accessibility Conformance Testing (ACT) Task Force"[10] is to standardize a format for testing procedures, to facilitate the development of a consistent set of testing procedures that aligns with WCAG. The development of the actual testing procedures, called ACT Rules, will continue to take place in the auto-WCAG community group, to facilitate participation and contribution by the broadest audience possible.

Presentation

This presentation will introduce the latest on WCAG Accessibility Conformance Testing (ACT):

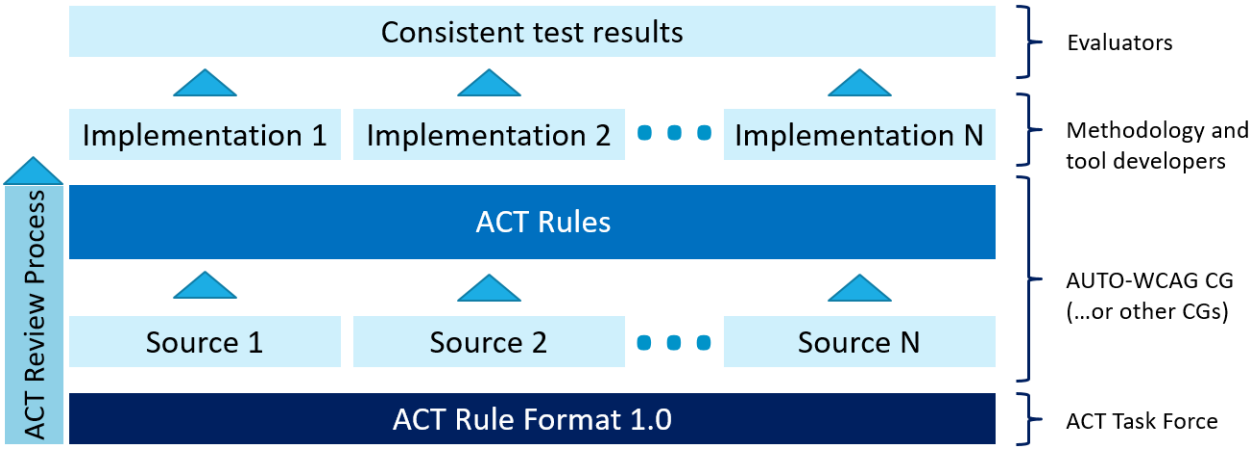


Figure 4: Components of WCAG Accessibility Conformance Testing (ACT)

Specifically, this presentation will:

1. Introduce the ACT Rule Format 1.0 specification from W3C[11]
2. Show ACT Rules developed according to this W3C standard[12]
3. Demonstrate open source implementations of the ACT Rules:
    a. Deque aXe Core[13]
    b. Siteimprove: (to be announced before the event)
    c. Difi Indicators (Norwegian government agency)[14]

Developers of automated testing tools and manual evaluation methodologies will learn how to adopt and use these internationally harmonized ACT Rules, as well as how to contribute to the current set of ACT Rules. Users of automated testing tools and manual methodologies will learn about the benefits of such internationally harmonized ACT Rules, and how to look for them and request them in the tools and methodologies that they are using.

Reference URLs

1. https://www.w3.org/WAI/intro/wcag
2. https://www.w3.org/TR/AERT
3. https://www.w3.org/TR/WCAG20-TECHS/
4. http://www.wabcluster.org/
5. https://www.w3.org/WAI/ER/tests/
6. http://www.openajax.org/
7. https://www.deque.com/products/axe/
8. https://www.w3.org/community/auto-wcag/
9. https://auto-wcag.github.io/auto-wcag/
10. https://www.w3.org/WAI/GL/task-forces/conformance-testing/
11. https://www.w3.org/TR/act-rules-format/
12. https://w3c.github.io/wcag-act-rules/
13. https://www.axe-core.org/
14. https://github.com/TilsynForUniversellUtforming/Indikatorar-for-WCAG-2.0

Copyright notice

Legibility of Videos with ASL signers

Raja S. Kushalnagar

Gallaudet University
800 Florida Ave NE, Washington, DC 20002 USA
raja.kushalnagar@gallaudet.edu

Abstract

The viewing size of a signer correlates with legibility, i.e., the ease with which a viewer can recognize individual signs. The WCAG 2.0 guidelines (G54) mention in the notes that there should be a mechanism to adjust the size to ensure the signer is discernible but does not state minimum discernibility guidelines. The fluent range (the range over which sign viewers can follow the signers at maximum speed) extends from about 7° to 20°, which is far greater than 2° for print. Assuming a standard viewing distance of 16 inches from a 5-inch smartphone display, the corresponding sizes are from 2 to 5 inches, i.e., from 1/3rd to full-screen. This is consistent with vision science findings about human visual processing properties, and how they play a dominant role in constraining the distribution of signer sizes.

Introduction

Signing is of fundamental importance in modern culture and communication among signers, primarily those who are deaf and hard of hearing. There are over 500,000 people who rely on sign language in the United States (Mitchell, Young, Bachleda, & Karchmer, 2006) and over 70 million people worldwide (WFD, 2017). Many deaf people prefer to communicate via sign language over English, as sign language is fully visual compared with English. This has led to the development of Deaf culture, a set of social beliefs, behaviors, art, literary traditions, history, values, and shared institutions of communities that are influenced by deafness and which use sign languages as the main means of communication. The advent of signed videos is comparable to the impact of the printing press in facilitating cultural transmission in cultures world-wide. Deaf people associate in closed, close-knit local communities linked to schools or locations. They are united by their common use of sign language, which functions as a symbol of identity, medium of interaction and basis of cultural knowledge (Baker-Shenk & Cokely, 1996; Hauser, O'Hearn, McKee, Steider, & Thew, 2010).

ASL communication is conveyed not only by hand movements, but also from head movements (e.g. head tilts), facial expressions and use of space around the body (Bellugi & Fischer, 1972). Signs are produced in a very specific body region defined from the waist, with reach of slightly bent elbows to the top of head, and from the location of the sign in space. The viewing size of the signer is a crucial factor determining the signer's legibility, i.e., the ease with which a viewer can recognize individual signs, not including finger-spelling. Sign legibility differs from sign readability, which is the ease with which a viewer can recognize signed narration.

When sign communication is conveyed over video, many characteristics of the sign communication between signer and viewer change due to technical constraints. Signing videos create a new relationship between signers and viewers in content delivery (Keating & Mirus, 2003, 2004), and that signers using cameras alter the spatial dimension of sign language, from three dimensions in face-to-face communication to two dimensions in video. Signers changed their body positions to accommodate the narrow signing window, due to the camera field-of-view constraints, compared with the space available in face-to-face communication. Signers changed to optimize the display of their bodies, face, and hands.

Other factors that impact the ease of understanding a signer include attributes such as the viewing angle between viewer and signer, the signer's signing space, the color of the signer's clothes, etc. Research has shown that sign language interpreter comprehension on television appears to be problematic, in part due to the small size of the interpreter picture and fast signing (Ofcom, 2007; Xiao & Li, 2013).

Size

Sign legibility has implicitly been driven by the properties of human vision, as evidenced by findings that writing symbols have evolved in response to visual pressures rather than writing pressures. Studies (Changizi, Zhang, Ye, & Shimojo, 2006; Morin, 2018) have found that legibility is related to contour configuration and size. Specifically, the distribution of contour topological configurations and sizes that are common in writing systems corresponds to the distribution found in natural images, implying that written forms have been designed to take advantage of visual mechanisms that evolved for perception in the natural world.

Unlike speech which is characterized by rapid and sequential order of words, sign language is characterized by slower production and simultaneous layering of components spread over the signing space of the body (Wilbur, 1999; Wilbur & Allen, 1991). Furthermore, the layering effect implies that the articulation of each piece of information cannot interfere with the perception and production of the others, and that the signing space has to be more clearly visible. In other words, signs have evolved to ensure that their units are clearly distinguishable.

Visual acuity (ability to distinguish visual differences) is sharpest within about 10 degrees range around the center of focus (Mandelbaum & Sloan, 1947). The angular size of visual information, including signers, is measured in minutes of arc or degrees of visual angle from the center of focus. The angular measure depends on both the physical size of the signer and the subject's viewing distance, which gives a more accurate measure of human perception of perceived visual size. The angular size is correlated with the viewing size in the eye, which is known as the retinal image size and is defined as follows:

$$\text{Angular size (in degrees)} = 57.3 \times \text{physical size} / \text{viewing distance}$$

The physical size and viewing distance are measured in the same units, such as millimeters, centimeters, or inches. This equation is an approximation, which holds when the physical size is significantly smaller than the viewing distance.

For instance, suppose the height of a displayed signer in a phone display is 14 mm (physical size), and the viewer keeps the phone at a typical viewing distance of 40 cm (16 inches). The angular height of the signer at the eye is 2°. If the reader reduces the viewing distance from 40 cm to 20 cm, the angular character size doubles to 4°, but the physical character size, of course, does not change. In clinical vision applications, angular character size is often expressed using metrics from visual acuity testing including Snellen notation (Sloan, 1951) or logMAR (Rosser, 2001).

Methodology

We collected and analyzed 240 videos that had one or more signers narrating information via American Sign Language. Of the 240 videos, half (120) were selected from DeafVideo.TV and the other half were from YouTube. YouTube is a general video-streaming service, while DeafVideo.TV is a video-streaming service specifically aimed at deaf signers and the Deaf community.

We selected videos in which the contributor was a fluent deaf signer adult that used ASL and had uploaded at least 50 posts to either video-streaming service. For all videos, we gathered video resolution and signing size, and assumed a standard viewing distance of 16 inches from a 5.5-inch phone display (Bababekova, Rosenfield, Hue, & Huang, 2011).

Data collection

The videos were assessed for quantitative and qualitative factors, addressing the research questions. The following quantitative factors were measured:

1. *Signing size*. The signing size was determined by measuring the size of the signer in relation to the size of the entire video.
2. *Signing rate*. Signing rate was determined by counting signs in of the video and then normalizing the count per second.
3. *Average length* (and standard deviation) of posts in seconds.

Filtering

To determine whether the contributor was a fluent signer, we looked for a statement from the video creator identifying themselves as a fluent deaf signer. This was most often found in the Deafvideo.TV profile or one of the YouTube videos. If text was included in the original video, it was excluded from the study to ensure that all content was ASL only.

On DeafVideo.TV, all videos were in American Sign Language, and the history of all of postings could be easily tracked on this site. Identifying ASL videos on YouTube was more complex as YouTube contains a very large range of audio-visual content with only some ASL content. ASL videos were found on YouTube by using the keyword, “ASL” as there was no American Sign Language or Deaf community on YouTube.

Results

The DeafVideo.TV distributions were clustered with a mean visual angle of 16° in a range from 7° to 16° (1.92 to 4.4 inches on a 5.5-inch tall display). These ranges indicate that the video creators judged the videos to be legible at around one third the screen size. For YouTube videos, the mean visual angle was 18°, and distribution is clustered like DeafVideo.TV but with a somewhat smaller range of 9-19° (2.47– 5.22 inches on a 5.5-inch tall display). These results indicate that the video creators judged the videos to be legible at around half the screen size.

To gauge whether there was a correlation between signer size and video length, a repeated measures ANOVA was carried out for the average length of videos. There was no significant difference between the videos or between YouTube and DeafVideo.TV. For YouTube, the mean length of the videos ranged from 295 seconds (SD = 127) to a high of 342 seconds (SD = 289). For DeafVideo.TV the mean length of the videos ranged from a low of 221 seconds (SD = 208) to a high of 328 seconds (SD = 193). Even though there is no length limit in YouTube videos, there were no instances of videos over 6 minutes long (360 seconds).

Conclusions

One of the criterions required for meeting WCAG 2.1 AAA is to provide sign language interpretation. Due to lack of access to spoken communication in a predominantly spoken world, a significant fraction of deaf individuals are not proficient in their communities' spoken language and read at a 4th grade level or less (Allen, 1986; Traxler, 2000) and may be more proficient in sign language compared with their communities' spoken language. Furthermore, some deaf people prefer to communicate via sign language over their communities' spoken language, as sign language is fully visual compared with English and provides intonation, emotion and other audio information that is reflected in sign language interpretation, but not in captions, sign language interpretation provides richer and more equivalent access to synchronized media.

When sign communication is conveyed over video, many characteristics of the signed information conveyance change due to technical constraints, primarily due to the change from three-dimensions to two-dimensions. WGAG 2.1, success criterion 1.2.6 (Sign Language, pre-recorded) states that sign language interpretation is provided for all pre-recorded audio content in synchronized media. addresses some of these concerns, such as a second video stream of the signer, to be viewed simultaneously. G54 notes that if the video stream is too small, the sign language interpreter will be indiscernible. When creating a video steam that includes a video of a sign language interpreter, make sure there is a mechanism to play the video stream full screen in the accessibility-supported content technology. Otherwise, be sure the interpreter portion of the video is adjustable to the size it would be had the entire video stream been full screen. However, this guideline does not specify the minimum size of the video with the signer, relative to the average viewing distance. The findings from our analysis of 240 videos by native signers on both YouTube and DeafVideo.TV suggests that the minimum viewing size of a signer should range from 1/3rd to full-screen.

References

Allen, T. E. (1986). Patterns of academic achievement among hearing impaired students: 1974 and 1983. *Deaf Children in America*, 161–206.

Bababekova, Y., Rosenfield, M., Hue, J. E., & Huang, R. R. (2011). Font Size and Viewing Distance of Handheld Smart Phones. *Optometry and Vision Science*, *88*(7), 795–797. https://doi.org/10.1097/OPX.0b013e3182198792

Baker-Shenk, C., & Cokely, D. (1996). *American Sign Language, A Teacher's Resource Text on Grammar and Culture*. Gallaudet University Press.

Bellugi, U., & Fischer, S. (1972). A comparison of sign language and spoken language. *Cognition*, *1*(2–3), 173–200. https://doi.org/10.1016/0010-0277(72)90018-2

Changizi, M. A., Zhang, Q., Ye, H., & Shimojo, S. (2006). The Structures of Letters and Symbols throughout Human History Are Selected to Match Those Found in Objects in Natural Scenes. *The American Naturalist*, *167*(5), E117–E139. https://doi.org/10.1086/502806

Hauser, P. C., O'Hearn, A., McKee, M., Steider, A., & Thew, D. (2010). DEAF EPISTEMOLOGY: DEAFHOOD AND DEAFNESS. *American Annals of the Deaf*, *154*(5), 486. Retrieved from http://search.proquest.com/docview/288323505?accountid=108 LA - English

Keating, E., & Mirus, G. (2003). Examining Interactions across Language Modalities: Deaf Children and Hearing Peers at School. *Anthropology <html_ent Glyph="@amp;" Ascii="&"/> Education Quarterly*, *34*(2), 115–135. https://doi.org/10.1525/aeq.2003.34.2.115

Keating, E., & Mirus, G. (2004). American Sign Language in virtual space: Interactions between deaf users of computer-mediated video communication and the impact of technology on language practices. *Language in Society*, *32*(05), 693–714. https://doi.org/10.1017/S0047404503325047

Mandelbaum, J., & Sloan, L. L. (1947). Peripheral visual acuity with special reference to scotopic illumination. *American Journal of Ophthalmology*, *30*(5), 581–588. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/20241713

Mitchell, R. E., Young, T. A., Bachleda, B., & Karchmer, M. A. (2006). How Many People Use ASL in the United States? Why Estimates Need Updating. *Sign Language Studies*, *6*(3), 306–335. https://doi.org/10.1353/sls.2006.0019

Morin, O. (2018). Spontaneous Emergence of Legibility in Writing Systems: The Case of Orientation Anisotropy. *Cognitive Science*, *42*(2), 664–677. https://doi.org/10.1111/cogs.12550

Ofcom. (2007). *Signing on television: New arrangements for low audience channels*.

Rosser, D. A. (2001). The development of a “reduced logMAR” visual acuity chart for use in routine clinical practice. *British Journal of Ophthalmology*, *85*(4), 432–436. https://doi.org/10.1136/bjo.85.4.432

Sloan, L. L. (1951). MEASUREMENT OF VISUAL ACUITY. *A.M.A. Archives of Ophthalmology*, *45*(6), 704. https://doi.org/10.1001/archopht.1951.01700010719013

Traxler, C. B. (2000). The Stanford Achievement Test, 9th Edition: National Norming and Performance Standards for Deaf and Hard-of-Hearing Students. *Journal of Deaf Studies and Deaf Education*, *5*(4), 337–348. https://doi.org/10.1093/deafed/5.4.337

WFD. (2017). Sign Language. Retrieved January 7, 2018, from https://wfdeaf.org/human-rights/crpd/sign-language

Wilbur, R. B. (1999). Stress in ASL: Empirical Evidence and Linguistic Issues. *Language and Speech*, *42*(2–3), 229–250. https://doi.org/10.1177/00238309990420020501

Wilbur, R. B., & Allen, G. D. (1991). Perceptual Evidence Against Internal Structure in American Sign Language Syllables. *Language and Speech*, *34*(1), 27–46. https://doi.org/10.1177/002383099103400102

Xiao, X., & Li, F. (2013). Sign language interpreting on Chinese TV: a survey on user perspectives. *Perspectives*, *21*(1), 100–116. https://doi.org/10.1080/0907676X.2011.632690

Copyright notice

Testing Video Players for accessibility

Gian Wild

AccessibilityOz
Melbourne, Victoria, Australia
gian@accessibilityoz.com

Abstract

We conducted a series of tests of 37 major video players, both free and paid, on the market. Initially we tested on a PC on Google Chrome and excluded video players that contained what we called "show-stoppers": serious accessibility failures. The remaining video players were then tested with people with vision impairments reliant on various screen readers. Any video players that contained show-stoppers were excluded from any additional testing. Thirdly, we tested on two different mobile devices, again excluding video players that contained show-stoppers. Finally, we tested with an iPad and a Bluetooth keyboard. At the end of the testing only two players remained: AblePlayer and OzPlayer.

Why is Video Important?

Video is now ubiquitous. One third of all online activity is watching video (Bowman, 2017), more than two-thirds of all internet traffic is video (Cisco, 2017) and more than half of all video content is accessed via a mobile device (O'Neill, 2016). More than 500 million people watch video on Facebook every day (Jarboe, 2016) – that's more than 100 million hours of video (Constine, 2016)! On YouTube, the numbers are even higher – more than 1 billion hours of video are watched every day (Etherington, 2017).

Video and Accessibility

Web accessibility is about making sure web sites, web applications and mobile apps (including video) are accessible to people with disabilities. The estimate of people with disabilities in the US is approximately 19% of the population – that's 56.7 million people (US Census, 2016).

Web accessibility of web sites is best achieved by following the Web Content Accessibility Guidelines, Version 2.0 created by the World Wide Web Consortium (W3C). The W3C states that "following these guidelines will make content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these" (W3C, 2016). The W3C Web Content Accessibility Guidelines, Version 2.0, contain specific techniques for providing accessibility features within video.

Almost everyone understands the need for accessibility features like transcripts, captions and audio descriptions for people with disabilities: these features also are important to people without

a disability when viewing video. More than 85% of all Facebook video is viewed without sound (Patel, 2016) – and, if a video has captions, a user is almost twice as likely to finish the video than if it did not have captions (Mandel, 2009). We all know Google is blind and deaf and we see that in revenue as well – videos with transcripts earn 16% more revenue than videos without transcripts (Robertson, 2012). And what has been dubbed the "Broadchurch effect" (due to the strong accents in the BBC drama "Broadchurch"), the BBC found that 80% of people who use captions are not using it for accessibility reasons (Ofcom, 2006).

To provide accessible video solutions, web site owners must provide a variety of accessibility features. One of the requirements is that the video player itself must be accessible. Lack of accessibility in video players can be the consequence of a number of things, but usually includes inadequate keyboard access, inoperable captions and non-existence of audio descriptions and transcripts.

Testing Video Player Accessibility

We tested the following video players: AblePlayer, Acorn, Adobe, AFB, Amazon, AMI Player, Brightcove, Facebook, JW Player, Kaltura, MediaElement, MediaSite, Ooyala, OzPlayer, Panopto, PayPal, Plyr, RAMP, Video.js, Vidyard, Vimeo, Viostream, Wistia, Yahoo, YouTube, and YouTube embed. (Disclosure: OzPlayer is an AccessibilityOz product.)

We conducted four rounds of testing:

- Round 1: Desktop testing on Google Chrome, Windows 10;
- Round 2: User testing with a vision-impaired user using a screen reader;
- Round 3: Mobile testing on iPhone and Android devices; and
- Round 4: Mobile testing on an iPad with a Bluetooth keyboard.

We identified certain "show-stoppers" that were failures of the four non-interference clauses in WCAG2: If technologies are used in a way that is not accessibility supported, or if they are used in a non-conforming way, then they do not block the ability of users to access the rest of the page (W3C, 2016).

Round 1 Testing

Initially we conducted testing on Google Chrome version 61.0.3163 under Windows 10. A series of tests were undertaken including whether the video player supported audio descriptions, whether the transcript was accessible to the keyboard and whether the volume of the player could be modified. Video players were deemed to include show-stoppers if any of the following occurred:

- Audio plays automatically (unless the user is made aware this is happening or a pause or stop button is provided) – failure of WCAG2 Level A Success Criterion 1.4.2: Audio Control;

- Video contains a keyboard trap (i.e. users cannot escape from the video using the keyboard) – failure of WCAG2 Level A Success Criterion 2.1.2 No Keyboard Trap; and/or
- Full-screen video contains a reverse keyboard trap (i.e. users cannot escape from full-screen mode using the keyboard) – failure of WCAG2 Level A Success Criterion 2.1.2 No Keyboard Trap.

At the conclusion of Round 1 testing only the following eight video players remained: AblePlayer, JW Player, Kaltura, OzPlayer, Panapto, Plyr and YouTube embed.

Round 2 Testing

Experienced vision-impaired users tested the remaining eight video players on the following combinations of screen reader / operating system and browser:

- JAWS with Windows 10 with: Internet Explorer; FireFox; and Chrome;
- NVDA with Windows 10 with: Internet Explorer; FireFox; Chrome; and Edge;
- VoiceOver with iOS Safari; and
- TalkBack with Android Chrome.

A series of tests were undertaken including whether the player's controls were labelled, button status was announced appropriately and fast-forwarding and rewinding was available to the screen reader user. A video player was deemed to include a show-stopper if the video could not be played– not a failure of WCAG2, but deemed a significant failure. Only one video was excluded at the end of this round of testing: Panopto.

Round 3 Testing

We tested the remaining seven video players Google Pixel 1, Android 8.0, Chrome and iPhone 7+, iOS 10.3.2, Safari. A series of tests were undertaken including whether the mobile video player supported captions and/or audio descriptions, whether the volume of the video player could be modified and whether color contrast was sufficient. A video player was deemed to include a showstopper if any of the following occurred:

- Video could not be played – not a failure of WCAG2, but deemed a significant failure;
- Video could not be paused – failure of WCAG2 Level A Success Criterion 2.2.2 Pause, Stop, Hide; and/or
- Video crashed the browser– not a failure of WCAG2, but deemed a significant failure.

Three video players: Kaltura, MediaSite and YouTube contained embedded show-stoppers. The remaining four video players were: AblePlayer, JW Player, OzPlayer and Plyr.

Round 4 Testing

We tested the remaining four video players on an iPad, iOS 10, Safari, Zagg keyboard. We tested only one task via the keyboard on the iPad: whether the controls could be operated by the keyboard. A video player was deemed to include a showstopper if any of the following occurred:

- Video could not be played– not a failure of WCAG2, but deemed a significant failure;
- Video could not be paused– failure of WCAG2 Level A Success Criterion 2.2.2 Pause, Stop, Hide; and/or
- Video crashed the browser– not a failure of WCAG2, but deemed a significant failure.

Two video players: JW Player and Plyr, contained show-stoppers. The remaining two video players were: AblePlayer; and OzPlayer.

Conclusion

This is the third year that this testing has been conducted. As in previous years, AblePlayer and OzPlayer were the only two video players that do not contain show-stoppers. Although we have seen improvements – such as more video players supporting audio descriptions – we have also seen the advent of new show-stoppers such as reverse keyboard traps. For every show-stopper encountered there will be thousands – perhaps hundreds of thousands – of people who cannot watch the video. As one of our screen reader testers said:

> "Video is a very important source of information and is a remarkable aspect of our culture now. Blind individuals must not be excluded from access to data provided in video content. You can't see, but you can understand." –*Rafal Charlampowicz*

Note

This paper was originally presented at ICCHP 2018, the International Conference on Computers Helping People with Special Needs, July 11-13, Linz, Austria.

References

1. Bowman, M. (2017, February 3). Video Marketing: The future of content marketing. Retrieved from https://www.forbes.com/sites/forbesagencycouncil/2017/02/03/video-marketing-the-future-of-content-marketing/#5417a4be6b53

2. Cisco. (2017, September 15). Cisco Visual Networking Index: Forecast and Methodology, 2016–2021. Retrieved from https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

3. O'Neill, J. (2016). Global Video Index, Q2 2016. Retrieved from http://go.ooyala.com/rs/447-EQK-225/images/Ooyala-Global-Video-Index-Q2-2016.pdf

4. Jarboe, G. (2016, February 12). 500 Million People are Watching Facebook Videos Every Day. Retrieved from http://tubularinsights.com/500-million-watch-facebook-video/

5. Constine, J. (2016, January 27). Facebook Hits 100M Hours Of Video Watched A Day, 1B Users On Groups, 80M On Fb Lite. Retrieved from https://techcrunch.com/2016/01/27/facebook-grows/

6. Etherington, D. (2017, February 28). People now watch 1 billion hours of YouTube per day. Retrieved from https://techcrunch.com/2017/02/28/people-now-watch-1-billion-hours-of-youtube-per-day/

7. US Census Bureau Public Information Office. (2016, May 19). Nearly 1 in 5 People Have a Disability in the U.S., Census Bureau Reports - Miscellaneous - Newsroom - U.S. Census Bureau. Retrieved from https://www.census.gov/newsroom/releases/archives/miscellaneous/cb12-134.html

8. W3C. (2016). Understanding Conformance. Retrieved from https://www.w3.org/TR/UNDERSTANDING-WCAG20/conformance.html

9. Patel, S. (2016, December 22). 85 percent of Facebook video is watched without sound. Retrieved from https://digiday.com/media/silent-world-facebook-video/

10. Mandel, R. (2009, March 23). PLYmedia: Subtitles Increase Online Video Viewing by 40%. Retrieved from http://www.subply.com/en/news/NewsItem_SubPLY_Trial_Results.htm

11. Robertson, M. R. (2012, June 2). 8 Best Practices to Optimize E-Commerce Video Landing Pages for Search. Retrieved from http://tubularinsights.com/8-practices-optimize-video-landing-pages-search/

12. Ofcom. (2006, March 23). Television access services: Review of the Code and guidance. Retrieved from https://www.ofcom.org.uk/__data/assets/pdf_file/0016/42442/access.pdf

13. W3C. (2016). Conformance Requirements. Retrieved from https://www.w3.org/TR/WCAG20/#conformance-reqs

Copyright notice

WCAG: Go Beyond and Be Creative!

Irfan Ali

Principal Research Systems Specialist
Educational Testing Service (ETS)
660 Rosedale Road, MS 03-C, Princeton, NJ-08541, USA
iali@ets.org

Abstract

Everyone is struggling with the best possible way to implement, support and achieve accessibility compliance. Web Content Accessibility Guidelines (WCAG) conformance, whether 2.0 or 2.1, is the global target and benchmark for building a more accessible Web. While WCAG 2.1 has been designed and written to be far more testable than its prior version, there are still situations that require more technical evaluation and focus on usable and accessible implementation. WCAG alone is not sufficient to guarantee website, content, and report accessibility nor is it sufficient to address all issues encountered by users with disabilities.

This paper provides details about some challenges beyond the WCAG guidelines and success criteria. It presents recommendations with the goal of producing a truly usable experience, including some practical examples and technical solutions for all users, not only those who have disabilities, while simultaneously conforming to the WCAG standards.

Introduction

The W3C Web Content Accessibility Guidelines, or WCAG (W3C, 2018) certainly provides useful guidance and associated techniques for building accessible content, but in the case of educational content, there are many stakeholders comprising a multidisciplinary team involved in design, implementation, and testing. It is essential to ensure all members of the multidisciplinary team take responsibility for accessibility and usability and ensure that they take all required steps. Complex content and interactions may not have immediate accessibility implementation solutions nor have clear direction from the success criteria of WCAG. Multidisciplinary teams can leverage their subject matter expertise, experience, and creativity to reach solutions that achieve what we all need - web content or an application that works for a user with disabilities. Ideally, it is best to have accessibility in mind from the beginning of a new web content design strategy. It is essential for all team members to understand that failing to address accessibility upfront can lead to costly remediation and delay deliverables. Discovering usability issues early is also critical and a key element of the approach described in this paper with the goal of explaining the benefit of thinking beyond WCAG guidelines conformance.

Whether you are a content subject matter expert, a specialist in image descriptions, a user interface designer, web developer or a manager, everyone has a responsibility to do their bit to contribute to accessibility. This approach can make sure that products are accessible from the

start, from inception to design to implementation to delivery, and don't need to be retrofitted with accessibility features at a late stage. New development efforts should incorporate accessibility right from the initial design process and utilize iterative prototyping and usability studies of novel interactions, for which we have no existing accessible design patterns. The prototypes themselves are based on research and best practices, but when for example, we are faced with novel interactions, such as simulations, we often don't start with a design, but instead seek to understand how a person with a disability would actually perform the real task. In understanding the real-world task, and how someone with a disability may perform it, we then seek creative solutions to identify if there is a digital equivalent. We do our research with a diverse audience representing the target user population and include focus group studies. During this iterative process, we conduct in-house or remote usability studies with end users with disabilities. Usability studies are also key for informing members of the multidisciplinary team on what works (the wins) and what doesn't (where did we get it wrong). It is critical to have all members of the team invited to observe the usability studies. Thus, in carrying out these studies, it's not just researchers but developers, designers and other stakeholders who participate. It is also important to engage the study participants and listen to their comments, observe where and how they use our services- their location, physical space and preferred assistive technology (AT). We may ask usability study participants to show us websites or related products that work well for them and then ask them to explain the problems in products that do not work very well. These are activities that are fundamental to designing a quality user experience; involving people with disabilities helps ensure an inclusive and high-quality experience. An interesting fact in this exercise is that sometimes we realize that even though all the WCAG standards are being followed, still the content may not be usable by everyone, including some people with disabilities who use assistive technologies.

Some Real World Examples

Applying WCAG intelligently is the key to accessibility, especially in educational content item types. But some content, such as math, region selection, and drag/drop are a few examples where we need to be creative and move beyond strict interpretation of WCAG. For example, we create custom selectable and de-selectable radio buttons for high-stakes testing assessments and examine ways to address drag and drop style interactions so that they work without inducing unnecessary cognitive load for the end user. Sometimes it may be as simple as replacing drag and drop tasks with simpler interaction methods such as multiple-choice radio buttons. Rather than trying to force a complex task to become accessible and usable, which instead results in something that is technically conformant with WCAG, but not usable in the real world, one needs to rethink the task into a more accessible and usable model that achieves the same goal. It is key to remember that it is the end user of your product, who may use an assistive technology, who is being faced with a complex interface, that may work well visually, for example, but induces excess cognitive demands to interact with it non-visually. Simply having all states, roles, and labels on all controls may not yield something that is usable. Remember to focus on the actual usability rather than only focusing on WCAG conformance.

Drag/Drop Example

Creating accessible, and usable, drag and drop interactions has been challenging in the education context. It is essential that the complexity of the interaction not get in the way of the user's ability to understand the content and the task. Drag and drop are visually oriented interactions to associate, match, or order objects, which may be text or graphical element. In the WAI-ARIA standard 1.1, `aria-grabbed` and `aria-dropeffect` have been deprecated (WAI-ARIA, 2018), though not likely to disappear from screen reader support in the near future. While there are examples and guidance for accessible drag and drop (e.g., Hausler, 2017; WHATSOCK, 2018), their usage needs careful consideration from a usability perspective. A key message to developers to enable usable interactions is focus on providing these three types of information any interaction or component of an interaction:

1. Identity: What am I interacting with?
2. Operation: How do I use this object?
3. State: What is the current status of object?

we can highlight some of the key design elements that can improve usability.

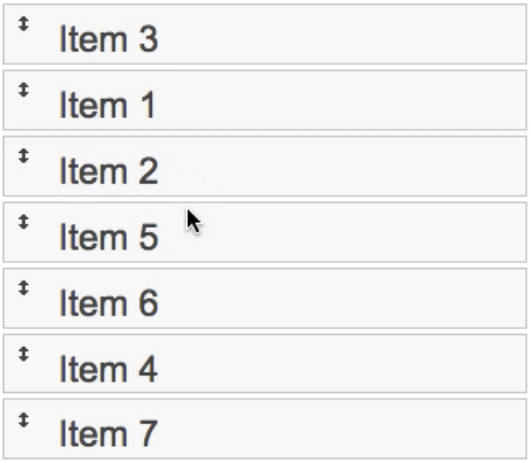


Figure 1 – Drag and drop to reorder a list

To identify the drag and drop, use **aria-describedby** to associate hidden text to each list item that says:

> *"Press space bar to grab this item."*

This will provide identity. When users grab the item, provide operation and state by placing text in live region:

> *"{Item name} grabbed, use the arrow keys to re-order, spacebar to drop. Escape to cancel. Current position in list, 1 of 7".*

You can also provide the final status after an item is dropped:

"{Item name} dropped. Final position in list, 4 of 7".

Screen reader users typically hear text as they navigate a web page or application. There are several ways in which a screen reader navigates the page, but all of the text they encounter is added to the end of the screen reader's queue and is relayed to the user on a first in, first out basis. Live regions are a tool for placing text into the queue without needing the user to navigate to that text on the page. For the purpose of drag and drop, we use an assertive region, so the user hears an update immediately. To make it usable for AT users, place the following span with the accompanied CSS on the page at page load. The content placed in this span will be immediately read out loud to screen reader users. It is visually hidden from the sighted users.

```
<span aria-live="assertive" class="assistive-text"></span>

.assistive-text {Position: absolute; left:-1000px;
width:1px; height:1px; overflow:hidden;}
```

The second part is a piece of code that will both clear the contents of the span and place new text inside. Each time new text is placed inside of the span, the information is immediately read to screen reader users.

```
 updateLive Text (string announcement) {

 //code to update text to our aria-live span

}
```

To reiterate, this method should only be used after native elements are ruled out and ARIA specified components either don't exist or don't work. As you are providing identity, operation, and state on your own, user testing will be necessary to determine the best way to communicate this information to the users.

And one caution: please make sure that a desire to provide a visual interaction such as drag and drop is really the best way to make an interaction usable for everyone. While less visually engaging, we have found many instances of drag and drop interfaces could be easily replaced with traditional multiple-choice selections, making them readily usable by everyone, including keyboard and assistive technology users.

Zone Selection

A common type of interaction in educational content is that of creating a zone within an image and enabling selection of a particular zone. In this example, there is a question, direction, and response, that are associated with a static graphic of a map, which has some city names. To make it accessible and usable, we need to be a little creative while using best practices. We can specify a custom region for each element, using "role" and "aria-label" attributes to provide a means for

the screen reader user to identify the parts of the over question: the question itself, the reference material (the map), and the selection zones and their labels. Region navigation is an alternative to heading level navigation, particularly important when hierarchical relationships aren't valid.

For the question, we can specify a "question" region using "role" and "aria-label" attributes.

Which of the cities shown in figure 2 is the capital on New Jersey?

```
<p role="region" aria-label="Question">Which of the cities
shown below is the capital of New Jersey?</p>
```

Similarly, we can specify a "directions" region using "role" and "aria-label" attributes.

Select a city to Indicate your answer.

```
<p role="region" aria-label="Directions">Select a city to
indicate your answer.</p>
```

Figure 2 – Map of New Jersey with major cities

One final important thing to keep in mind when writing the code for "response" region- don't rely solely upon the WCAG guidelines but be little creative and use ARIA, CSS and JavaScript wisely. We need to create the pointers on the static image, and, while the HTML imagemaps is still supported, it is not the best approach. I recommend creating a radio group and use "Radio" role for each city, with an "aria-checked" attribute to indicate state and make each tab a stop. The

“aria-setsize” and “aria-posinset” attributes are used to announce the number of the radio button within the set.

Also, use blank “alt” attribute and “aria-hidden=’true’” to prevent announcement of the static image by assistive technology. While the image of the map may be useful in answering the question, an extended description of the map would be recommended over the alt text. An extended text description may be in the form of the following and utilize an embedded list of the six cities: “Map of the state of New Jersey. Six cities are shown. Starting from the north to the south, they are 1. Jersey City, 2. Edison …” Typically, we provide the extended text description in a visually hidden element, with a heading of “Extended Figure Description”. Depending on the context, we may wrap the image, and the description within the <figure> element. In cases where position of the cities within the map would be crucial in understanding the question, we may utilize a tactile, embossed supplement containing the map, and reference it in as part of the figure description. The guidance on image description provided by the DIAGRAM Center (DIAGRAM, 2018), can be useful in understanding how best to present graphical information in an accessible and usable form.

```
<div role="region" aria-label="Response">

<img src="nj.png" alt="" aria-hidden="true"/>

<div role="radiogroup">

<div id="c1" tabindex="0" role="radio" aria-checked="false"
aria-setsize="6" aria-posinset="1">Jersey City</div>

<div id="c2" tabindex="0" role="radio" aria-checked="false"
aria-setsize="6" aria-posinset="2">Edison</div>

<div id="c3" tabindex="0" role="radio" aria-checked="false"
aria-setsize="6" aria-posinset="3">Trenton</div>

<div id="c4" tabindex="0" role="radio" aria-checked="false"
aria-setsize="6" aria-posinset="4">Toms River</div>

<div id="c5" tabindex="0" role="radio" aria-checked="false"
aria-setsize="6" aria-posinset="5">Atlantic City</div>

<div id="c6" tabindex="0" role="radio" aria-checked="false"
aria-setsize="6" aria-posinset="6">Cape May</div>

</div></div></div>
```

ARIA live region and Static text in DOM

Assistive technologies will announce *dynamic* changes in the content of a live region. The live region must first be present (and usually empty), so that the browser and assistive technologies are aware of it. Any subsequent changes are then announced. I have found in most occasions that

developers add static text within the live region and add a function to show/hide the text which doesn't work. In following example, the error block is having static text with display property.

```
<div role="alert" id=”error” style=”display:none;”>Please
fix the form errors.</p>
```

Simply including an `aria-live` attribute or a specialized live region `role` in the initial markup as it's loaded will have no effect. Dynamically adding an element with an `aria-live` attribute or specialized `role` to the document also won't result in any announcement by assistive technologies. Always make sure that the live region is present in the document first, and only then dynamically add/change any content.

Adding Instructions before the dynamic functionality

Providing instructions for any dynamic functionality for assistive technology users is a common practice. However, adding this instruction in information architecture could be little tricky to make sure that instructions are placed properly before the functionality and not impacting user experience for general users who are not using assistive technologies. It sounds very obvious, but I have seen it very often that instructions are misplaced. Web forms are perhaps the most common type of functionality that contain instructions. Of course, while this additional instruction might be extremely useful for screen reader users, it may not be visually favored by sighted users. For this reason, we hide the text. There are several ways in which you can hide text, each with different level of support. It is important that a technique be implemented that results in the desired outcome and accessibility

Text Indentation

Providing a `text-indent: -10000px;` moves the content to the left 10000 pixels – thus off the visible screen, but screen reader will still read text with this style. However, if a link or form element is given this style, it may result in a focus indicator that extends from where the element should be located in the page to the place it is actually located which is not pretty. This approach is very viable option if the element does not contain navigable elements.

Zero-pixel sizing techniques

Adding `width:0px, height:0px` will make element with zero height and width and element will be removed from the flow of the page, and most screen readers will ignore this content. It is not recommended to size content to 0 pixels if you want the content to be read by a screen reader. Content styled with `font-size:0px` may work, though the elements would still take horizontal space on the screen. All of these techniques may result in search engine penalties as they may interpreted to be malicious.

CSS clip

A fairly modern technique of using CSS to hide or clip content that does not fit into a 1-pixel visible area will essentially hide the content visibly, but still allow it to be read by modern screen readers.

Positioning content off-screen

Using CSS to move hidden elements to a position off-screen is the most useful and accessible method of hiding content visually.

```
<div class=”hidden”>This content is hidden.</div>

.hidden {position:absolute; left:-1000px; top:auto;
width:1px; height:1px; overflow:hidden;}
```

Conclusion

WCAG is key to implementing accessibility to your projects, but to make it accessible and usable for every user, use the guidelines wisely and creatively along with usability studies to test out novel or complex interactions. Engage with users with disabilities who use assistive technologies to better understand how to improve the user experience. Finally, understand how to leverage key features in standards such as WAI-ARIA to create interactions that provide the information needed by assistive technology users to successfully use your creation. These are important pieces of the puzzle of usable accessibility and allows those at the forefront – creators – to constantly improve and push the boundaries to build better experiences for all users.

References

DIAGRAM (2018). Image Description. DIAGRAM Center. Retrieved from http://diagramcenter.org/making-images-accessible.html

Hausler, J. (2017). 4 Majors Patterns for Accessible Drag and Drop. Retrieved from https://medium.com/salesforce-ux/4-major-patterns-for-accessible-drag-and-drop-1d43f64ebf09

W3C (2018). Web Content Accessibility Guidelines (WCAG) Overview. Retrieved from https://www.w3.org/WAI/standards-guidelines/wcag/

WAI-ARIA (2018). Accessible Rich Internet Applications (WAI-ARIA) 1.1. Retrieved from https://www.w3.org/TR/wai-aria-1.1/

WHATSOCK (2018). The Accessibility Tree. Retrieved from http://whatsock.com/training/#hd12

Invisible Content for screen reader users. Retrieved from

https://webaim.org/techniques/css/invisiblecontent/

Copyright notice

Built for All: A badge for accessibility

Claudio Luis Vera

Fort Lauderdale, FL
claudio@simple-theory.com

Abstract

When selecting software and components, how can we tell what's accessible? As I researched this paper, I found that there few reliable methods short of testing and vetting the software yourself. In most cases, a component was accessible simply because it had been tagged as such. After speaking with accessibility leaders in different open source communities, I also found that their criteria varied wildly from one community to the next, sometimes falling far short of accepted industry standards. One proposed solution would be to create a badge to show that a piece of software has met these standards. While it sounds simple, a badge would require having standardized processes in place for testing and evaluating software, and the community and technical infrastructure to back it up. This paper explores those requirements and proposes an open source project to handle them.

The scope of waste

Without trusted authorities to evaluate software for us, all of us designers, developers, product managers, publishers and business decision makers would need to test every new component we include in our work.

That's even more wasteful than it sounds: aside from testing the component hundreds or thousands of times, we would also need to implement fixes as many times. That redundancy adds up.

Here's an example: A front-end developer creates an open-source photo gallery slider that aims to be accessible. She tests the photo gallery for accessibility on various browsers, devices and breakpoints using available tools – to the best of her ability. She posts the source code for the photo gallery on GitHub and publicizes it within the developer community. Within a short time, the gallery has been downloaded 15,000 times, and has been put into production at least 10,000 times.

Since our developer is working pro bono and donating her work to the community, she hasn't considered hiring expert testers to ensure that her work product is accessible. After all, they're often billed out at $200 per hour.

As the photo gallery gets widely used, some accessibility issues begin to pop up. A few of the photo gallery's users are diligent enough to report their issues to GitHub, but most others make fixes without reporting them – or just ignore the issues altogether. If we assume that it only takes an average of 2 hours to diagnose and fix these issues, it would still take 20,000 person-hours of

developer time to make every live instance of her photo gallery accessible. That's a few orders of magnitude greater than the 300 hours she originally invested in testing and development.

If you now apply this logic to, say, every popular WordPress theme that was not developed with accessibility in mind, then the scope of the problem starts to approach person-*millennia* of technical debt or misused developer time. When you extend that redundancy to the entire software and publishing community, then the amount of waste approaches the GDP of a small country.

A badge for accessibility

One could argue that better choices could have been made if the person selecting the software was aware of the level of its accessibility. If there were a software badge from a trusted authority, the selection process would be trivially easy. Without a software badge, however, the decision maker must rely on anecdotal information from fellow developers or trust a potentially outdated listing in a directory of accessible components.

An accessibility badge would require an entity to support it with the following elements:

- A transparent testing methodology based on open standards
- An established workflow for handling testing
- A community of trusted testers with adequate training
- Infrastructure for automated testing
- Infrastructure for hosting a multitude of project sites with issue queues and comments
- Sufficient resources to cover the infrastructure
- A compelling business plan
- Strong brand awareness with endorsement from established players

Of all the various types of entities (NGO, government agency, for-profit corporation, non-profit, etc.) an open-source project would offer the shortest ramp-up time with the smallest funding requirements.

Evolving collaboration platforms

The past few years have been a period of rapid evolution in how we work together and how we share our work. Collaboration tools and workflows have evolved at a breakneck pace and have been adopted by new disciplines. For example, technologies like git and semantic versioning have grown their user base beyond niche developer communities to include user experience (UX) professionals and visual designers. These advances offer the accessibility community new opportunities that it has not yet capitalized.

Here's an overview of how this evolution has taken place:

A starting point: shared code repositories

For anyone developing software or front-end code, the natural choice for efficiency is to build libraries of reusable components. These libraries begin at the grassroots team level, and then may be rolled up into repositories that are shared internally within business units. For someone implementing accessibility standards in a large organization, code repositories are an excellent place to "bake in" long-term compliance.

Style guides

A similar evolution has been taking place within the design community: years ago, UX designers began developing style guides (https://warpspire.com/kss/), in which they specified design patterns for re-usable front-end components. Style guides began as static documents, being curated by a small number of individuals with the intent of developing a permanent set of branded design guidelines.

Design systems

As the reach of a style guide grows to include an entire enterprise, it requires ongoing contributions from a broader constituency that includes developers, product owners and other stakeholders. To meet this challenge, many style guides have evolved into living, breathing design systems with version control and extensive documentation.

Beginning with software giants like Google and Microsoft, some enterprises began to view their code and design libraries as valuable assets to share publicly with the community (e.g. Material Design (https://material.io/) and Fluent Design (https://www.microsoft.com/design/fluent/)). Other brands have followed suit, with Airbnb offering an extensive visual design spec in its Design Language System (https://airbnb.design/the-way-we-build/) and Salesforce publishing a design system https://www.lightningdesignsystem.com/) for its Lightning Platform.

Design systems differ from style guides in that there is no pretense of permanence. Instead, their standards and documentation are bound into planned release cycles much like software.

Efforts by the accessibility community

Whitelists and project directories

- The simplest way of letting developers know which tools are best for accessibility is to compile a whitelist of accessible software built by others. Here are a few examples:
- The Web Accessibility Initiative (WAI)'s list of Accessible UI Components (https://www.w3.org/blog/wai-components-gallery/submission-guidelines/) (34 items).
- The Global Public Inclusive Infrastructure (GPII) Developer Space (https://ds.gpii.net/) (800+ items)
- The A11Y Style Guide (http://a11y-style-guide.com/style-guide/)

In some cases, items are submitted by the vendors and developers themselves, while others are carefully curated by the directory's owners. However, none of these examples have a posted methodology for validating the submissions for "accessibility, security or privacy considerations (https://www.w3.org/blog/wai-components-gallery/submission-guidelines/#text-4)".

Accessible frameworks

There are a few instances in which developers have set out to build standalone frameworks of fully accessible components, such as Inclusive Components (https://inclusive-components.design/) designed, built, and maintained by Heydon Pickering. While technically promising, they are a challenge for a handful of maintainers to sustain in the long run.

Groups in established communities

For years, there have been accessibility groups which have had varying degrees of success within well-established open source projects like WordPress, Drupal, and React. In at least one case, they've succeeded in securing a commitment from the community's leadership to being accessible in their core product (https://www.drupal.org/about/features/accessibility). Normally though, contributed themes, plug-ins, and other components don't meet WCAG 2.x or other accepted standards in any of these open source projects.

What is Built for All?

Built for All is a proposal for an accessibility badge, and the infrastructure and community required to sustain it.

The Built for All software badge

A Built for All badge would display that a component or software product has met a particular standard for accessibility, allowing developers to make selections based on peer review and uniform testing criteria. A Built for All badge is not a warranty, and it is not intended to be a substitute for testing or provide indemnity for the end products built with any of the components.

Software authors can include the badges in their documentation and publicity materials, thus promoting their commitment to accessibility and the Built for All community.

Versioning and scope

Any whitelist, directory, or badge for accessibility must address the issue of how to certify software that is continually evolving and changing – or risk its content becoming obsolete over time. Not only should a badge specify that a component is accessible, but it also will need to specify the precise version and scope of that component's codebase.

In semantic versioning, code is assigned a set of three version numbers using split into major releases, minor releases, and patch releases.

Using this schema, patch releases refer to bug fixes where the code remains backwards compatible and the functionality remains the same. Minor releases add functionality, but the code still remains backwards compatible.

In order to be effective, a design system or accessibility badge would need to work hand-in-hand with the products' versioning method. Here's how it would work within Built for All:

- Built for All would assign a badge to a specific version of a component and remain *fully* valid for patch releases within a *minor* release.
- A Built for All badge would be *partially* valid for minor releases within a *major* release.
- A Built for All badge would no longer be valid after a major release and require testing for renewal.

Testing and tiers

In an ideal world, software would be tested for accessibility three different ways: automated testing for code compliance, manual testing by experts, and user testing by persons with disabilities. Because manual and user testing are labor intensive, they would require an army of testers which would not be feasible in the early stages of the Built for All community.

In automated testing, the candidate software would be tested against a set of guidelines and rulesets, which will be posted publicly on the Built for All community's website. For smaller candidate products like single components, the tests would be run from a community testing server. For larger projects, automated testing would be downloaded and run locally on the author's virtual server instance using a technology like Vagrant or Docker. The candidate software will be given a score based on the number of rules passed, and the author will be encouraged to correct any detected issues and re-submit their work for another round of automated testing. Once a certain threshold score has been reached, then the work can be submitted for manual testing by human testers.

In the long run, Built for All could eventually offer badges in different levels, similar to LEED certification:

- **Regular**: passes automated testing only
- **Gold**: passes automated and manual testing
- **Platinum**: passes automated, manual and user testing

Issue queue

Progress through the evaluation process will be posted in an issue queue, a structured forum where authors and other community members can provide comments and patches. Once all accessibility issues have been resolved, then the candidate will be marked as reviewed and tested by the community (RTBC) and issued a Built for All badge with the candidate's version number.

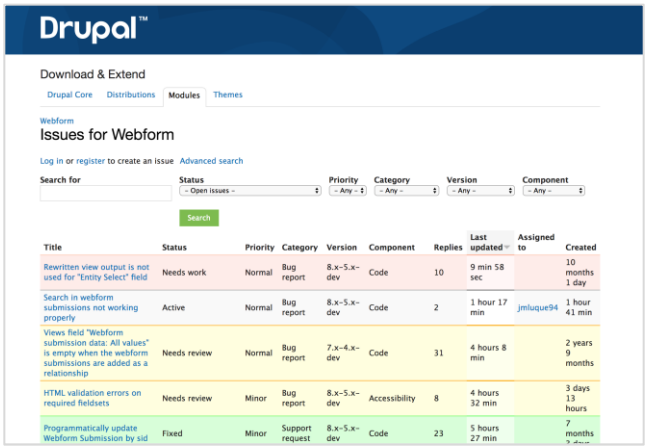


Figure 5: Issue Queue Example from the Drupal community (https://www.drupal.org/project/issues/webform?categories=All)

The Built for All community

As a community, Built for All would require a mix of participants based on their skill and level of involvement. Below is a list of different tiers in the organization, going from lowest to highest:

Members

- Volunteers to report issues
- Volunteers for documentation

Basic membership in the community will be free.

Testers and evaluators

- Junior testers
    - Trained in very specific tasks
    - Can be an onramp to certification
- Trusted testers
    - Experts in assistive technologies like screen readers
    - Experienced in writing audits
- User testing subjects
    - Moderate experience with assistive technology

Overall, testers can be paid evaluators or work pro bono, depending on their expertise, the level of sponsorship, and availability of their particular skillset.

Contributors

- Developers for contributing new testing tools and methods
- Developers for writing patches
- Mentors and trainers for more junior members
- Perform peer review through pull requests of fellow contributors' work

Sponsors

- Individuals / contributing members
- Companies that provide in-kind support such as trusted tester resources for more ambitious projects
- Financial underwriters that contribute to the community's overall operating budget

Governance

- Governing association with rotating leader
- Treasurer
- Infrastructure management
- Core administrative staff

Built for All infrastructure

The Built for All community will need a server infrastructure to handle a variety of different needs:

- **Public servers** for the community's web site, member profiles and issue queues
- **Testing/CI server** for hosting automated testing and downloadable testbots
- **Code servers** for hosting the issue queue and projects with git integration

Incentives

From the start, Built for All will be a meritocracy based on members' level of contribution to the community. Each level in the community would come with an appropriate Built for All member badge and credentials to allow members, testers, contributors, and sponsors to publicize their status.

Members

Members will be offered the ability to be mentored on testing skills in exchange for service to the Built for All community. Also, programs like the IAAP's planned tester certification program could provide further incentives by requiring or encouraging some community service as part of the certification process. Both these incentives would provide the added benefit of growing the number of qualified accessibility testers.

The Built for All community would provide a much-needed onramp for newcomers to accessibility to become trusted testers and contributors.

Sponsorship

A typical software project (like the photo gallery example) would enter the issue queue for review and testing, and its accessibility issues would be tackled by volunteers as they become available. A larger-scale project like an enterprise app could be fast-tracked by sponsoring the Built for All community's operating expenses, which in turn could fund dedicated testers and reviewers for that project.

Conclusion

The simplest way to roll out Built for All would be to launch it as a pilot project within one single open-source community with a culture that is committed to accessibility. Likewise, Built for All could be beta-tested within an enterprise with a well-established design system and a commitment to accessibility and inclusion.

Although a business plan still remains to be written, an accessibility badge would spare the software community a vast amount of waste—perhaps enough to change the economics of digital accessibility.

Copyright notice

Three Developer Behaviors to Eliminate Accessibility Defects

Shane Anderson

Sr. Accessibility Engineer
Optum
13625 Technology Drive, Eden Prairie, MN 55344
Shane_Anderson@Optum.com

Thomas Dinkel

Sr. Accessibility Engineer
Optum
Thomas.Dinkel@Optum.com

Sean Kelly

Sr. Accessibility Engineer
Optum
Sean_Kelly@Optum.com

Abstract

In numerous development and remediation projects at Optum, we find a surprising number of what seem like preventable accessibility defects. Developers can prevent these defects by adopting three behaviors. We introduce accessibility to developers using terms they understand and techniques they already use. By combining HTML code testing, automated accessibility scans, and keyboard testing, developers address a wide array of accessibility issues before they create problems. The end result is that developers work more efficiently because they create fewer defects, and they are better able to learn how accessibility works.

Introduction

Developers for many of the projects we work on lack fundamental skills – not only in accessibility, but also HTML and JavaScript skills. This is in part because the increasing demand for software engineers means the role of web developer is now the entry point to the programming field. As more experienced engineers move on to more complex problems or management positions, junior developers often left to figure things out on their own. Knowing little about the broader needs of the systems they're building or maintaining, they inadvertently introduce accessibility errors. Even mature teams with active usability programs easily forget accessibility with the emphasis on the "typical" user.

Although the inevitable movement of engineers along their career paths is something we do not control, we encounter accessibility issues from many inexperienced developers for three reasons. First, many pay only minimal attention to the quality of their HTML. While developers constantly check their JavaScript for errors, they almost never seek feedback on HTML, partly because it is so forgiving. Second, accessibility is often an afterthought. It's not tested until later in a project, if at all. And last, many developers lack an understanding of modes of interaction. They assume that when something looks right and works with a mouse, that's all that's needed.

By addressing these deficiencies, we give developers the tools to produce better quality, accessible code. Looking for errors in syntax and semantics is a familiar task developers learn in their training for compiled languages. The added benefit for accessibility is that the practice also broadens awareness of the different ways people interact with technology.

HTML is key

Valid HTML is key to our approach. We see it as the root of most accessibility issues. Much of the HTML code we see in the sites we evaluate has little consideration for, and sometimes fully disregards, any meaning it's intended to convey. In short, we encounter a chaotic mess of HTML, and there are several reasons for this.

HTML has a complicated past. Like an old Swiss army knife, many use HTML components for purposes they were never intended. Over time, HTML morphed from a static document language to a language that defines dynamic user interfaces for applications we call websites. While HTML is rarely used anymore to create documents, static document markup is the basis for its most used components. Like many human languages, HTML follows convention rather than the specification at times. For developers unfamiliar with colloquial uses and idioms of that language, HTML seems unnecessarily confusing.

A confused developer ignores the formalities of HTML and chooses an easier path. One such path is to use a single tag type for all purposes. This leads to what is called "div-itis," in which the `<div>` element is used for most components. This makes sense on the surface because HTML elements are containers for specific types of content, and the `<div>` element is a generic container. Instead of maintaining many types of containers, a confused developer finds it easier to use a generic one and differentiate the containers using CSS classes.

Browser evolution is another factor. In the 1990s browsers competed with each other by adding support for new HTML elements. Many of these were normalized and adopted in the HTML specification. In the 2000s people realized the importance of standardizing HTML and CSS, but at the same time new browsers had to support old browser rendering to compete with Internet Explorer. Browsers therefore became very flexible in rendering any HTML monstrosity thrown at them. Modern browsers attempt to fix the HTML as it is parsed into the document object model (DOM).

Another reason is the misinformed focus on the flexibility of HTML5. Many developers felt HTML5 rewrote the rules for HTML. Unfortunately, they interpret this to mean that you can use semantic elements if you want to, but if you don't, that's fine. XHMTL was a stricter specification allowing for easier validation, but it was document centric. HTML5 added much

needed elements for website mark up, but, ironically, because of the false perception, flexibility and customization overshadowed the new, much needed constructs.

Finally, modern website frameworks and user interface (UI) libraries bring a false sense of not needing to worry about HTML. On one hand, HTML has improved since framework and library creators care at least a little about standards. On the other hand, people often believe incorrectly that those improvements are complete. In our work, we often hear that it's only necessary for developers to worry about implementing the business logic. They believe frameworks and libraries are responsible for the HTML output. Implementing accessibility is more difficult for some team, because they didn't budget any time or money for modifying the HTML.

Developer behaviors

HTML testing

HTML5 is a W3C standard recommendation and should be treated as such. HTML5 as conceived is the best solution for consistency across the industry in every way, including for accessibility. Ideally development teams adopt HTML as a standard they code to. Most teams adopt tools and methods introduced by a team member. We can use that behavior to introduce HTML testing and demonstrate the value of HTML as a standard.

Developers already have tools and methods for testing JavaScript. Many modern frameworks have testing frameworks built in. Even teams who don't have integrated testing use tools built into the browser to get immediate feedback. This same immediate feedback doesn't exist for HTML within the browser, but there are extensions, plug-ins, and external tools to do this.

Developers are surprisingly eager to adopt HTML testing, sometimes more so than testing for accessibility directly. Since many of the accessibility errors we find are directly tied to poor HTML practices, testing HTML is a clear path to solving a problem. It directly addresses the familiar syntactical and lexical issues they face with other programming languages.

Once a developer has feedback on their HTML, they use it to fix their code just like they do with JavaScript. That developer then shares the results with other developers on the team. Soon they integrate HTML testing into their testing framework and voilà – HTML is a standard they follow.

HTML testing

The easiest route testing is to point developers to the Nu Validator at the W3C (https://validator.w3.org/nu/). To use it, a developer types in the URL to be scanned. Browser plug-ins exist that test the current page by sending the HTML on command to the Nu Validator.

Automated accessibility scanning

The purpose of concentrating on HTML is to ensure developers understand the medium through which they help or hinder the end user, especially those with disabilities. While HTML validation ensures good HTML syntax, accessibility scanning tools ensure the HTML is used in

a way that supports accessibility. By testing HTML first, accessibility tests are less likely to contain false positives due to HTML errors and rendering issues.

This mirrors what developers do to test code written in other languages. Compilers check syntax and basic language rules. Once developers know the software compiles, they look for semantic errors, a check for logically correct implementation of the language syntax. Developers typically run semantic tests by providing inputs and then looking for expected outputs.

Tools of the trade

For accessibility scanning we like to introduce developers to WAVE by WebAIM and aXe by Deque Systems.

WAVE visually exposes underlying semantic HTML and accessibility features (or the lack thereof). We often introduce it first, because up until HTML testing, the only testing developers usually do is a quick visual inspection of a rendered web page to ensure it “looks” good. WAVE takes this one step further. Developers can compare the visual appearance of a page with the underlying HTML and accessible structures to ensure they complement each other. Graphic elements such as lines, colors, or even bullets on a list visually convey meaning and break up content into understandable and coherent units. If the HTML elements used don’t match the visual meaning, then people with disabilities cannot perceive the content in the way it was intended. WAVE helps expose HTML misuse.

We introduce aXe next. It digs a little deeper to expose accessibility issues such as color contrast, incorrect or missing ARIA, and syntactically correct HTML with semantic issues. The aXe browser plug-in is free and widely available on multiple browsers. The output of an aXe scan displays in the developer tools part of the browser where developers already do a lot of work.

Both WAVE and aXe provide developers a way to complement their HTML tests. As with HTML validation, page scans are more comprehensive when the scanning tool runs with each content interaction, such as a menu before and after expansion.

The online version of WAVE is found at https://wave.webaim.org. Download extensions for Chrome and Firefox at https://wave.webaim.org/extension/.

Download the aXe plug-in at https://www.deque.com/axe/.

Testing keyboard operation

Keyboard functionality is a fundamental feature of HTML that is the basis for compatibility with many assistive technologies. It is a prerequisite for any screen reader use. Many people who can see, but who have motor or other disabilities, use a keyboard alone to navigate.

Keyboard operation and focus issues account for a significant number of individual accessibility issues. They tend to be tied closely to human perception and automated scans generally do not detect them. Issues can be obvious, such as lack of visible keyboard focus or poor contrast of links and controls. More subtle keyboard accessibility issues include the page tab order not matching reading order, lack of notification when contextual focus changes based on user input, and form autocomplete functions that do not respond to keyboard input.

Keyboard testing is commonly the first introduction a developer has to people who use a site differently than they do. It's not that they don't know a keyboard can be used, but the knowledge that some people **only** use a keyboard is often a revelation.

A person must do keyboard testing. There are no automated tools that can detect the need for keyboard focus, the order of focus, if it should work with arrow keys, or if the visual focus indicator is noticeable enough for everyone. Getting into the habit of keyboard testing brings developers closer to the experience of the end user. It empowers developers to make decisions based on their knowledge of the various ways people interact with technology.

Our keyboard testing follows the guidance found at WebAIM at https://webaim.org/techniques/keyboard/.

Conclusion

When a project introduces the requirement for accessibility, the reaction is often negative. Developers are often thrown into training that covers the broad topic of accessibility and they're told their hard work fails to meet expectations. While some of this is unavoidable, the best strategy is to meet developers on their knowledge level with behaviors they understand.

These three developer behaviors can prevent a vast majority of accessibility issues while building a lasting skillset that increases the overall quality of a product. HTML code testing, automated accessibility scans, and keyboard testing give developers the tools to address a wide array of accessibility issues before they create problems, without having to become accessibility experts. This approach reduces the sheer number of accessibility issues to be addressed and makes remediation easier. Developers work more efficiently and therefore create fewer defects, and they learn how accessibility works on the job.

Copyright notice

A11yFirst for CKEditor Changing the Way Authors Think about Editing Content

Nicholas Hoyt

University of Illinois at Urbana/Champaign
Disability Resources and Education Services
1207 S. Oak Street
Champaign, IL 61820
nhoyt@illinois.edu

Jon Gunderson, Ph.D.

University of Illinois at Urbana/Champaign
Disability Resources and Education Services
1207 S. Oak Street
Champaign, IL 61820
jongund@illinois.edu

Abstract

Most web content is created by people using web-based WYSIWYG editors embedded in content management systems like blogs, learning managements systems, and administrative websites. Most of the authors of such content have little understanding of accessibility principles or the importance of accessibility to the organization hosting the website. The goal of this project is to change the author's mental model of the editing task to include accessibility as part of the creation process, which includes helping them to learn how to use structural content like headings and lists, to create appropriate text alternatives for images, and to provide usable text labels for links. Probably the most important change is to make sure authors understand that their organization values accessibility and they play an important role in making sure the website is accessible.

Current Model of Accessibility

The current model for accessible WYSIWYG authoring using editors like TinyMCE and CKEditor is a remediation model. Authors typically create content without any training or thought of accessibility to people with disabilities. When they complete the editing of a web page they need to remember to press the accessibility checker button and go through the document, "fixing" accessibility issues found by the checker. This approach has several problems in supporting the creation of accessible content. First, the user must remember to press the accessibility checker button and review all the accessibility issues. The checker button also

contributes to the stereotype that accessibility is hard and takes extra time, since the author thought they were done, but now finds they are not done and need to review content and make changes. The accessibility checkers are limited to checking items with known accessibility issues, typically ALT text for images and the proper nesting of headings (assuming the author used them at all). The author gets no information on other accessibility issues that cannot be checked automatically, for example the quality of a text description or having a longer description for an image in the content of the page, or whether a section of content needs a heading or sub-heading.

Changing Users' Mental Models to Include Accessibility

Another approach is to change the user interface to build accessibility into the default authoring process by either keeping authors from performing actions that are less accessible (e.g. enabling only allowed headings) or by providing them with just-in-time information when an accessibility problem occurs (e.g. poor link text). But just changing the user interface without changing the user's understanding of the task only frustrates users who are confused over the purpose of the new behaviors and features of the editor. The user first needs a "getting started" resource orienting them to a basic definition of accessibility, the accessibility goals of the organization and their role in supporting accessibility.

Conceptual Model for Accessibility

The conceptual model for the a11yFirst project identifies the objects and actions needed for the task of creating accessible documents. Elements of the conceptual model are represented in the user interface by the interactions it provides to support authors in creating accessible content. Some of the problems we have with current WYSIWYG editors is that they are based on a conceptual model of HTML editing, which assumes that the author understands both the syntax and the semantics of HTML elements and how these semantics support accessibility. Most authors do not understand HTML syntax and even fewer understand the semantics of specific HTML elements and how their semantics affect accessibility. Their lack of understanding leads to the overuse of inline styling techniques, which does not convey the accessibility information that the appropriate HTML elements would convey.

The following are the fundamental concepts the author must understand:

- Accessibility definition: inclusive information access for people of all ability levels
- Accessibility is a top-level priority for the organization
- Author has a personal responsibility within the organization to support accessibility

Fundamental changes to the CKEditor user interface to support accessibility:

- Promote the use of blocks as the main components of a document (heading, list, image, address, blockquote, code snippet)
- Discourage users from reaching for inline style options on first impulse
- For specific components, prompt for accessibility-related information (e.g. link display text or image alt text)
- When possible, limit the available options relating to accessibility (e.g. show only available heading levels)
- Provide easy access to accessibility information from within the editor

Design of a11yFirst Plug-ins and Toolbar

After reviewing the APIs for both TinyMCE and CKEditor WYSISWYG editors, CKEditor was chosen since it seemed to have a more mature and well documented API than TinyMCE and many content managements systems support the use of CKEditor. The project developed the following plugins and a toolbar design that helps authors differentiate between block formats and inline styling options.

- a11yfirsthelp: A new help menu for additional information on the a11yFirst plugins and accessible design information.
- a11yheading: A new plug-in that supports proper nesting of headings, setting other paragraph (e.g. block) level styles and a help option.
- a11yimage: A modified version of the image2 plug-in with options to choose image type and add alternative text information and a help button.
- a11ylink: A modified version of the link plug-in with accessibility feedback and help button.
- a11ystylescombo: Only inline styles that affect a character selection.

Toolbar Layouts

Figure 1 – – Screen shot of the a11yFirst Recommended Toolbar Layout

Figure 1 shows the layout of the a11yFirst toolbar used in the GitHub distribution repository and is designed to help prioritize block level actions over inline actions. One of the goals of the project is to help authors think about blocks over inline styling. The top row of options are block level actions providing authors with the capability to insert headings, paragraph, lists, block quotes, images, tables and code snippets. It also has options for paragraph alignment and a

“Show Blocks” button for people to easily view the blocks being used in the document. The Heading/Paragraph menu is at the top left of the toolbar to help authors understand its importance. The top row includes the a11yHelp menu button as the right most option to make it easier for people to see and help users understand that accessibility is a high priority for the organization. The second row primarily includes inline level actions as well as the Cut, Copy and Paste functions and a button to check accessibility, using the CKEditor Accessibility Checker plug-in.

a11yFirst Help Dialog

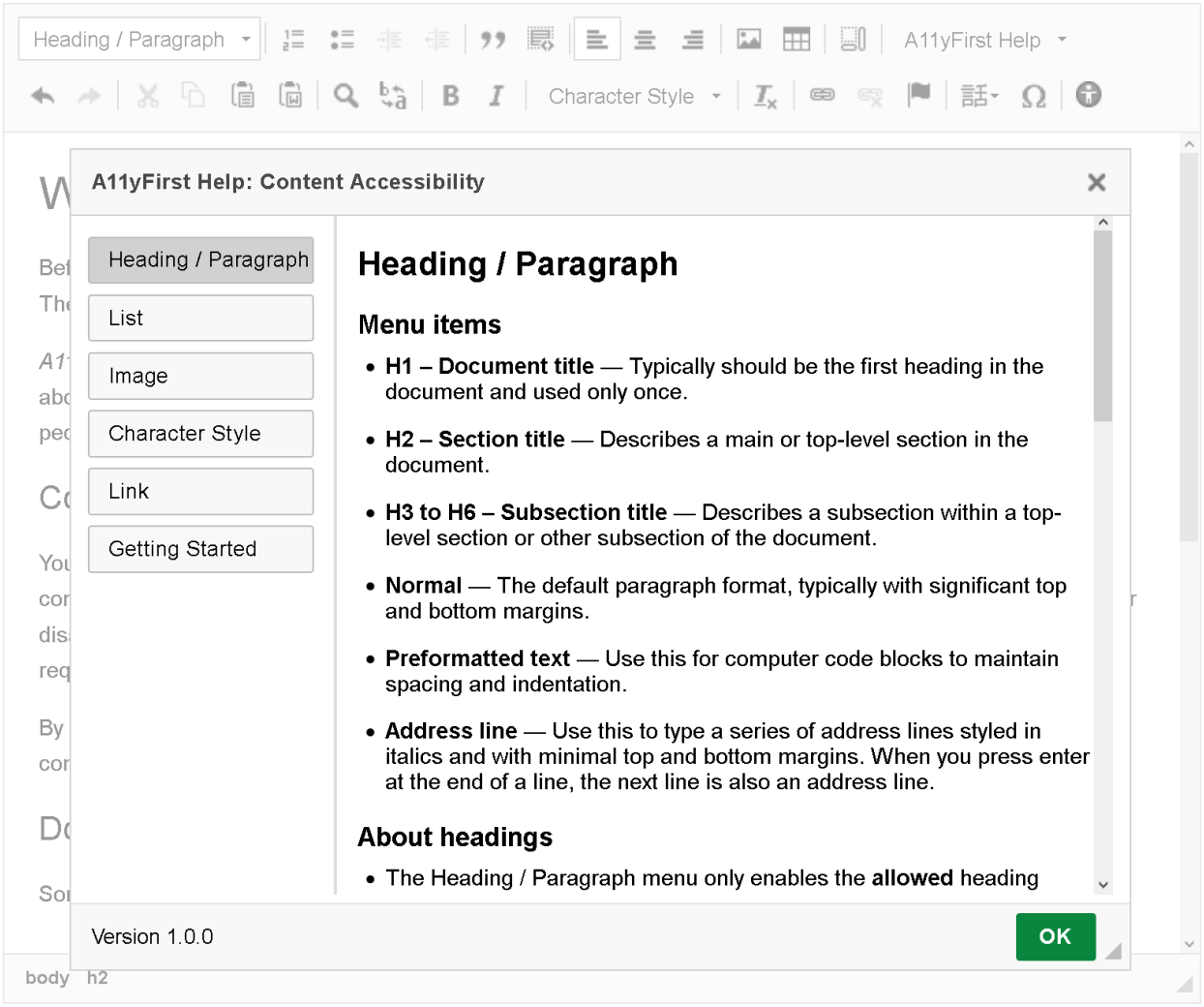


Figure 2 – – Screen shot of the a11yFirst Help Dialog Box

Figure 2 shows the a11yFirst help dialog and is an important reference for authors to learn about the basic requirements for creating accessible documents. The a11yFirst Help button that opens the dialog is in a prominent place (i.e. top row, right-most button) to make it easy for authors to find. The options in the dialog highlight the important issues in creating accessible documents, including the proper use of headings and paragraph formats, the use of list markup, the accessibility issues of describing images and links, and helping authors understand the difference

between inline and block level styling of content. In contrast, accessibility checkers often assume authors know something about accessibility and therefore provide only minimal information on accessibility issues. This is frustrating to authors who are not familiar with accessibility, often leading to them to make poor decisions on correcting accessibility problems and re-enforces their preconceptions of accessibility as being hard or only useful to a small group of people.

Heading/Paragraph and Character Style Menu

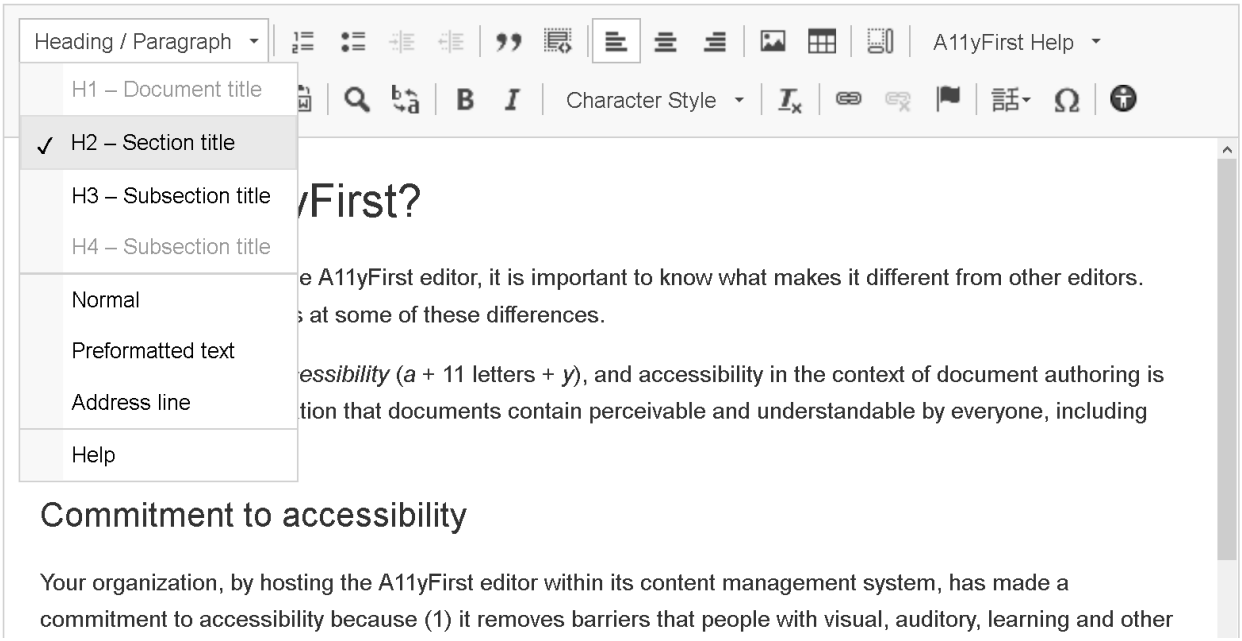


Figure 3 – – Screen shot of the a11yFirst Heading/Paragraph Menu

Figure 3 shows the Heading/Paragraph menu that replaces the default “Format Menu” in CKEditor. The Heading/Paragraph menu went through several revisions based on user testing to identify labeling that helped users understand the use of headings for titling sections and subsections of the document. This resulted in labels that identify both the HTML heading element (e.g. H1-H6) and the purpose of the heading (e.g. Document, Section or Subsection Title) to reinforce the use of each heading level in creating a structured outline of the contents of the document. In addition to the labeling of the heading levels, the menu also disables and enables heading levels based on the cursor position. Heading level are allowed based on the last heading level found before the cursor position and only one H1 element is allowed within the document to serve as the title. This helps authors understand the proper use and nesting of heading levels. The other options in the menu are normal paragraph and specialized paragraph styles of address and preformatted text. There is also an option to go the a11yFirst Help system to get more information on the use of headings and paragraph styles.

In the default CKEditor toolbar the “Styles” menu includes both block-level styles and inline styles. We felt this was confusing to have both block and inline styling in the same menu. We built a “Character Styles” menu that only includes inline styling options and this helps reduce the confusion over the effects of block and inline styling.

Image Properties Dialog

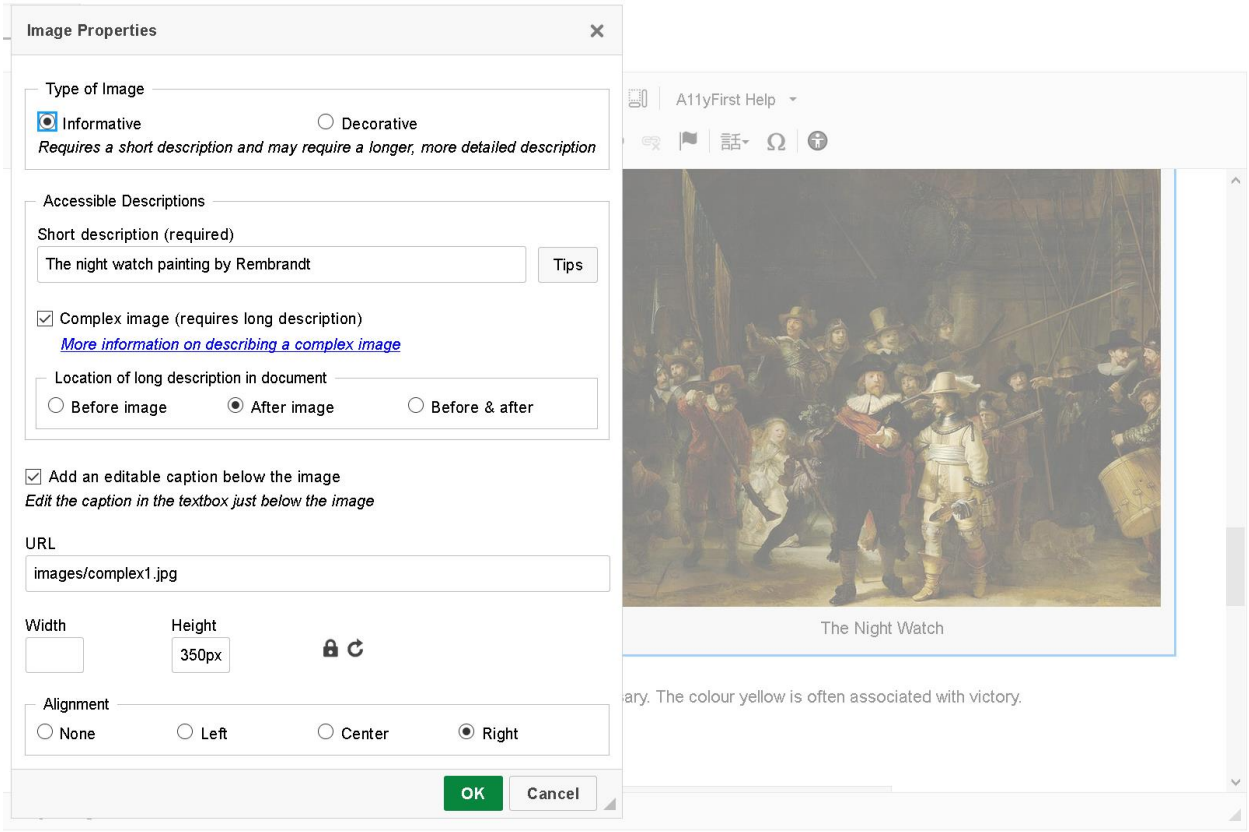


Figure 4 – Screen shot of the a11yFirst Image Dialog

The a11yFirst image plugin is the most complex addition to CKEditor and its goal is to help authors properly identify the purpose of an image in order to provide appropriate short and long text descriptions of the image. The default CKEditor image dialog provides a textbox for "Alternative Text", which is an optional field in the dialog. The a11yFirst image dialog requires the author to identify the image as either "Informative" or "Decorative". When the informative option is selected the author is required to provide a short description. Help text beneath each option, when selected, helps authors understand the difference between the two options. Selecting the decorative option also removes the "Accessible Description" section from the dialog, which emphasizes that this option requires no description. The "Accessible Description" section includes a textbox for providing a required short description and also allows the author to optionally indicate the presence of a longer description of the image in the prose of the document. The inclusion of the longer description in the document reinforces the use of universal design principles by providing multiple ways for a reader to understand the content of the document. The "Tip" button opens the a11yFirst help dialog for the author to learn more about short descriptions of images and a link under the "complex" image checkbox provides information on when a longer description is needed and options for including the long description in the document. The location of the long description is placed in the title attribute of the image and is typically read by screen readers after the short description.

Link Dialog

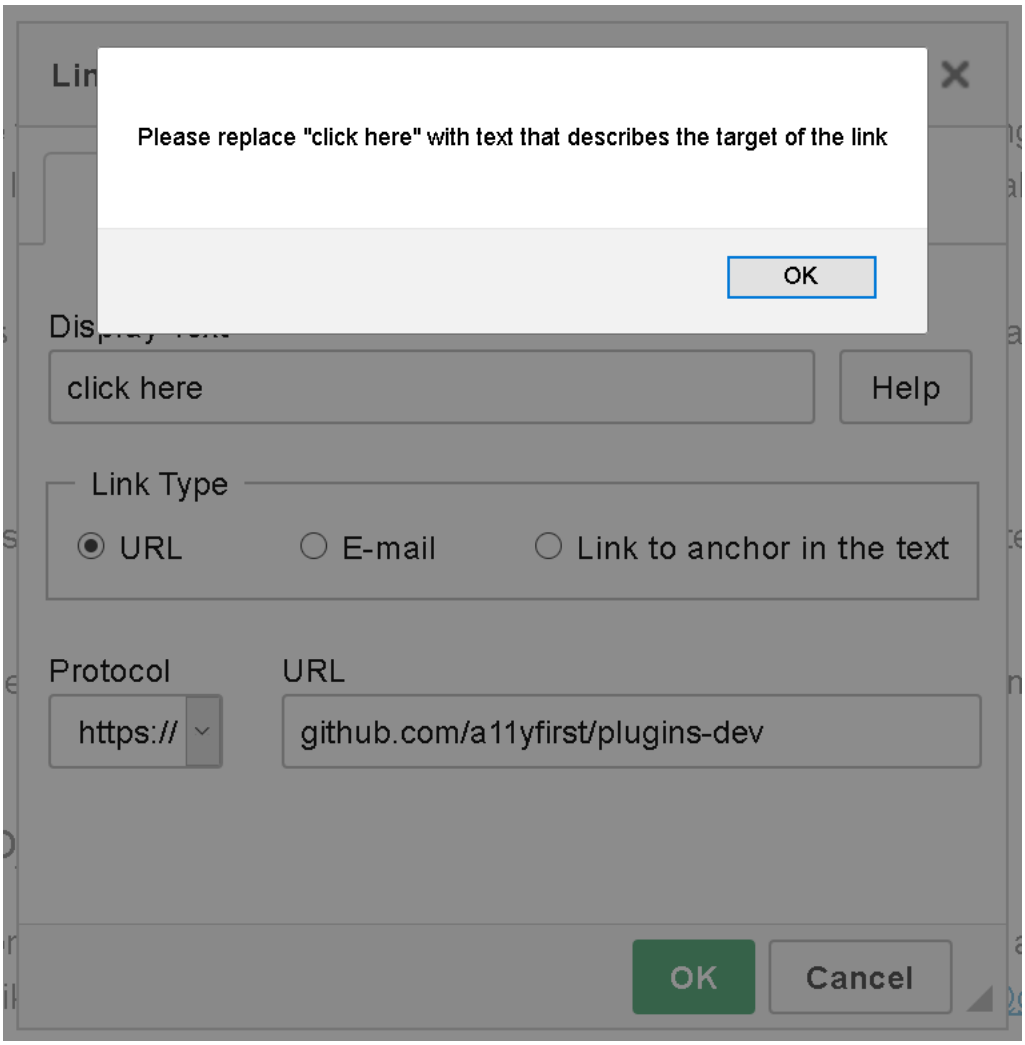


Figure 5 – a11yFirst link dialog with immediate accessibility feedback

Figure 5 shows a screen shot of the link dialog, which does not look any different than the CKEditor standard link dialog. The difference is that additional form validation has been added to the dialog. If the author uses phrases like “Click here” or “More” as the display text, an alert dialog box is used to prompt the user to provide link text that more accurately describes the target of the link. In the default link dialog for CKEditor, if the display text field is left blank, it will insert the URL as the display text. In the a11yFirst validation, the author is prompted with a confirm dialog asking them to confirm that they actually want to use the URL as the display text and informs them of potential accessibility implications of using the URL. A “Help” button next to the display text filed opens the a11yFirst help dialog to provide additional information on writing accessible display text for links.

Usability Testing

Usability testing was conducted on the a11yFirst plug-ins in the University of Illinois Library usability lab throughout the development of the plug-ins. The following are some of the most important findings:

- Just replacing the default Format and Inline Style CKEditor plug-ins with A11yFirst Heading, Block Format and Inline Style plug-ins led to user frustration.
- Authors need support in understanding whether an image is informative or decorative, and if informative, whether it is considered complex, thus requiring a longer description.
- Authors need a “Getting Started” resource to orient them to the importance of accessibility and their role within the organization.
- Just-in-time feedback on accessibility was useful to authors in learning about and creating accessible content.
- Authors accepted changes to the user interface of the editor once they were oriented to the need for accessibility

Links to Demonstration and Code

The following is a link to a demonstration version of CKEditor using the a11yFirst plug-ins:

> https://a11yfirst.library.illinois.edu/

This link is to the a11yFirst GitHub organization that hosts all a11yFirst repositories, including a pre-compiled distribution:

> https://github.com/a11yfirst

Copyright notice

Priorities for the field: Management and Implementation of Testing within Accessibility Programs

Chris M. Law

Accessibility Track
McLean, VA, USA
chrismlaw@accessibilitytrack.com

Pina D'Intino

Aequum Global Access
Pickering, ON, Canada
pina@aequumaccess.com

Julie Romanowski

State Farm
Mahomet, IL, USA
jr.romanowski@gmail.com

Rob Sinclair

Independent
resinclair@msn.com

Abstract

A survey was conducted by the International Association of Accessibility Professionals. In this paper, we convey survey results pertinent to the management and implementation of testing within organizations. There were 205 responses to the survey, 136 of which came from large corporations (over 500 employees), government departments, and educational institutions. Focusing on the subset of 136 responses, we examine accessibility programs from the perspective of target users, motivating factors for programs, training, and leadership and management components. For all 205 respondents, we examine the accessibility professionals' role with respect to accessibility program maturity. We summarize with three areas of discussion on priorities for the field: (1) What to test, and how to train?; (2) Who tests? Accountability and shared responsibilities; and (3) What should the accessibility field focus on? Legal compliance versus Inclusion. We suggest that in developing new resources for supporting accessibility programs in large organizations, the accessibility field should focus on what we found to be the primary business motivator, namely legal compliance / risk management.

Background

In April, 2018, members of the Organizational Development Committee (ODC) of the International Association of Accessibility Professionals (IAAP) surveyed accessibility professionals regarding their organizational accessibility programs. The ODC develops resources and programs to help organizations, corporations, and government entities grow their accessibility programs and strategies to increase overall access and opportunities for persons with disabilities.

The authors of this paper are members of the ODC who worked with other ODC members to formulate the survey (see also acknowledgments, below). The goals of the survey were to (1) identify areas where professionals in the accessibility field were experiencing successes and/or any shortcomings with available resources; (2) gather demographic and other data relating to respondents' real-world practices and experiences; and (3) help define and refine plans for future resource/content creation by the ODC. In this paper, we will present and discuss the implications of a subset of survey results that are of interest to those involved with the management and implementation of testing within larger organizations.

There were 205 responses, and 42% of these were non-members of IAAP. This paper focuses on the survey results—and our assessment of the implications of those results—that pertain to the management and implementation of testing methods within accessibility programs.

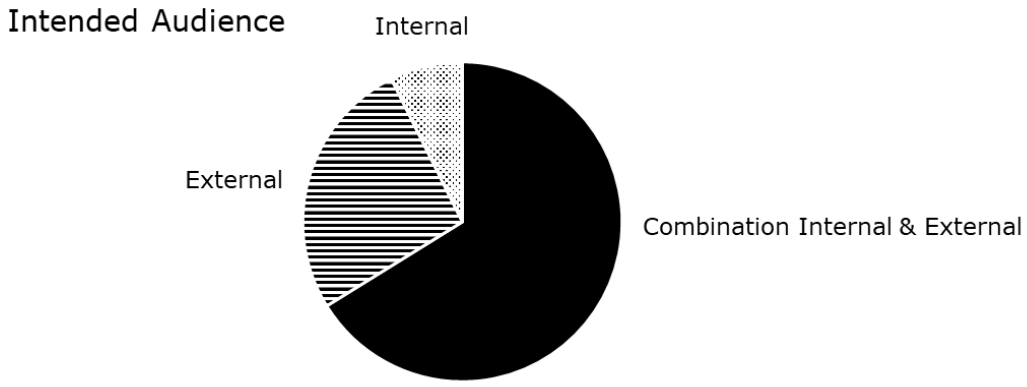
SECTION 1: Large organizations and issues relating to the management of testing

Two thirds (66%) of the 205 respondents worked in either large corporations (over 500 employees), government departments, and educational institutions. The remaining third worked in small businesses of 15 or fewer, mid-size companies, and nonprofit organizations. For the purpose of the analysis in Section 1, we are including results only from the 136 (two-thirds) 'large organization' cohort, as these are organizations where the implementation of some type of accessibility testing program and policy might reasonably be expected, or desired. We might also expect that the accessibility needs of external customers with disabilities, as well as the needs of employees with disabilities, would be of interest to the staff of these organizations.

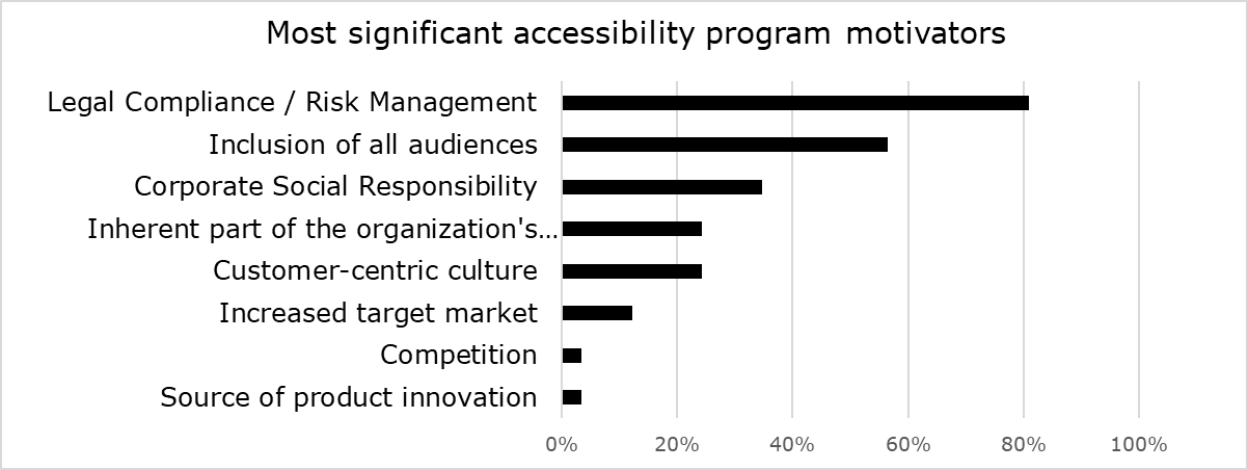
Target users

Two thirds of those working in large organizations said that they focus on both the needs of external customers as well as the needs of internal employees.

For the accessibility field, the desirable response to the question of "Who is your target audience?" is a combination of internal and external audiences, as seen in two thirds of large organization respondents. This represents an encouraging number for the field, but it also highlights that there is still much to do. If one third of respondents are focusing their efforts on only one of these target audiences (internal, 26.5%, or external, 7.4%) and neglecting the other, there is an opportunity for better educational resources to promote 'inclusive' corporate cultures and the need to address both customers and employees.



Legal Compliance is the number one motivator



Note: For the 'top motivators' survey question, n=115. The question prompted for the top three motivators, and respondents who selected >3 motivators were discarded.

At a certain level, business is about increasing possible gains and reducing possible risks. If we consider past messages—originating from the accessibility and inclusive design fields, directed towards industry and large organizations—the former was dominant. There were attempts to cost-justify accessibility by arguing that it is better for business due to more potential customers for your products and services (e.g., Keates, 2007, Hassel, 2015). Reduction of legal risk might be mentioned, but it was not addressed or promoted to the business community as a primary motivator of action. However, we found for the large organizations in the survey that only 12% of respondents said that increased target market was a motivator of their accessibility program, and a mere 3% said that it was a competitive advantage. Instead, from the large-organization perspective, avoidance of legal risk is the highest scoring motivator, cited by 93 of 115 large organization respondents (81%) of the time. While the past foci of the accessibility and inclusion fields has been to promote concepts such as inclusion, corporate social responsibility (CSR), and customer-centric approaches, it would appear that industry is focused on risk avoidance. However, 66 of those 93 respondents (71%) *also* cited CSR *or* inclusion as motivators of their programs. We can speculate that this is representative of what we, the authors, believe is happening, that the world is moving away from the more narrowly focused view of investing

only as much as is needed to mitigate legal risk. We can also speculate that after an organization initially focuses on risk, they become more accepting of CSR and inclusion as a natural result and expansion of this activity within the organization. This would be an interesting follow-up study to pursue to gather further data.

As the number of successful accessibility lawsuits increases, accessibility programs are being utilized to ensure that compliance with applicable laws and requirements are in place. For the implementation of testing programs, this provides a solid justification for their importance (i.e., without testing, how do you know you are legally compliant?).

For those who primarily promote the argument for the 'business case', the message is clear: the business case is not something that spurs action in large organizations. Focus on legal compliance first, if you want to mitigate risk, and then add inclusion and business case arguments after the initial change is made.

Training of employees

We might reasonably expect that in large organizations, ongoing training would be commonplace. How many of those organizations included accessibility training for raising awareness among new and existing staff, and elevating organizational culture to be more inclusive to people with disabilities?

Question: What is your organization's approach to accessibility training?	**Response Rate**
All (Mandatory training required for all employees)	6.6%
Some (Mandatory training required for some employees)	16.9%
Optional (Optional training available internally for all employees)	36.8%
None (No formalized approach to Accessibility training)	36.0%
Other (Currently developing training, a mix of options)	3.7%

These training results are not so encouraging for those who work in the accessibility and inclusion fields. For many years, one of the key messages in the workplace has been inclusion (as well as 'accessibility' and 'universal design') because, depending on how you measure, around 15 to 20% of the population has a disability (Hassell, 2015). This applies to customers as well as potential employees. However, it is also well known that people with disabilities are a significantly marginalized minority when it comes to employment rates, with 36.2% versus 78.9% in the general population (IED, 2018). Further, one recent survey found that in workplaces, although the typical rate of self-disclosure of having a disability (to employers) is 3.2%, the actual rate (including those who choose not to disclose) is 30% (Sherbin & Kennedy, 2017). Given these figures, we might hope that in 2018 inclusion was a natural part of everyone's training in large organizations, but our results show that this is far from the case (6.6%). While there is 'some' or 'optional' training for many, there is still a long way to go before corporate culture in general might be considered anywhere close to having an appropriate level of training.

It starts with policy, but…

Leadership components of accessibility programs

We asked respondents to indicate which leadership components were in place at their organization (note: the order on the survey was quasi-randomized by the list arbitrarily alphabetical, but the items below are in descending order of response rate, given in parentheses):

- Written organization-wide commitment or policy regarding Accessibility and Inclusion (58%)
- Accessibility criteria integrated into contracts and purchase orders for products/services (43%)
- Accessibility checking built into design, development and test tools (43%)
- Funding and dedicated resources for Accessibility throughout organization (31%)
- Actively engaged Executive Sponsor(s) for Accessibility (30%)
- Senior role leading an organization-wide program (e.g., Chief Accessibility Officer) (27%)
- Documented Business Case for Sustained Investment in Accessibility (13%)

Out of the seven leadership response options offered, the most popular item was mentioned by 58% of respondents, and that was having a written organization-wide policy on accessibility and inclusion. While policy is often described as the cornerstone of any accessibility program (The Accessibility Switchboard, 2018a), the ultimate goal is to turn policy into practice by involving the organization's leadership and establishing management components for a reliable and on-going accessibility program . (We address management components below.)

Another common leadership suggestion in guidance for starting or revamping accessibility programs is to begin by establishing support from executive sponsors before proceeding with any attempts at implementation The Accessibility Switchboard, 2018a). Without executive support, management and staff are free to say "no" to accessibility requests (The Accessibility Switchboard, 2018b). We did see encouraging leadership elements mentioned by roughly a quarter to two fifths of respondents (in order from low to high): Senior role leading an organization-wide program (e.g., Chief Accessibility Officer); Actively engaged Executive Sponsor(s) for Accessibility; Funding and dedicated resources for Accessibility throughout organization; Accessibility checking built into design, development and test tools; and Accessibility criteria integrated into contracts and purchase orders for products and services.

It was fortuitous that we included a catch-all response for 'other' on this survey question. When we wrote the survey, none of the ODC team considered that there might be a response that there was *no* leadership activity at a given organization. The results say otherwise, in that 18 of 136 respondents in large organizations provided a definite statement that this was the case where they worked. This represents 18% of cases in which leadership is not engaged at all on accessibility. Even though many components are in place at other organizations, no one leadership component was in place at a rate above 58%. As a field, we need to find a way (urgently) to remedy this situation.

Management Components of Accessibility Programs

Ideally, leadership implements accessibility programs through management and appropriate funding. However, we often find that dedicated individuals (some use the term 'evangelists') try to build a program from the 'ground up', trying to convince their peers to implement and manage an accessibility program (Law, 2010). We did not ask which process was employed to create management components, but we did ask respondents to indicate which types of components are present.

With respect to accessibility testing, four of the components are pertinent:

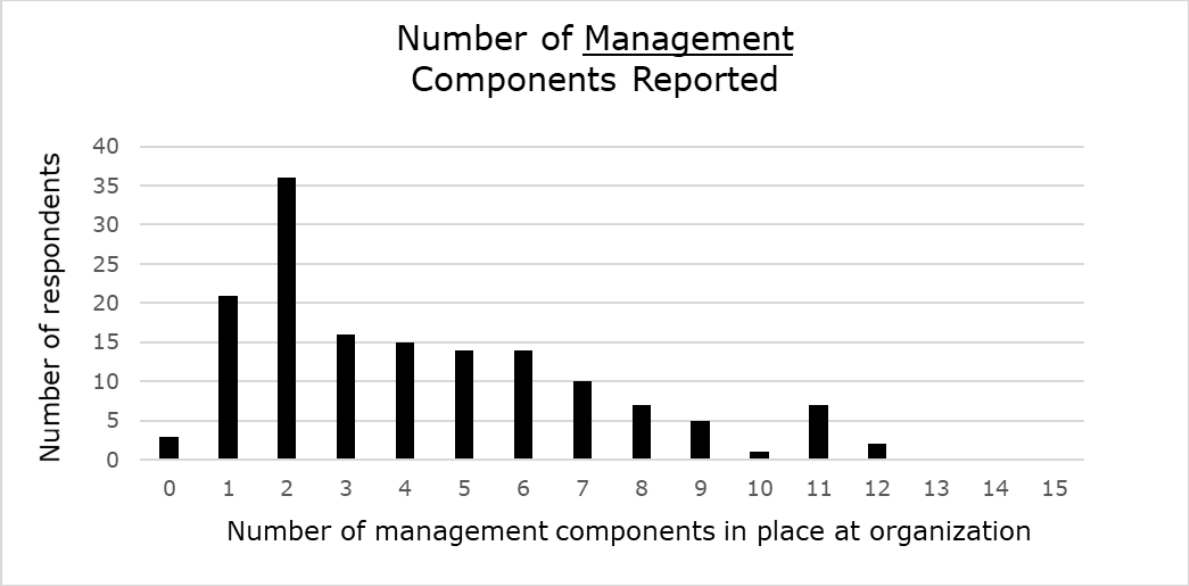
- Engineering practices for design, development and testing of accessible products and services (32%)
- Processes for declaring and reporting compliance or conformance externally (26%)
- Internal Accessibility Scorecard(s) to evaluate success (23%)
- Accessibility Maturity Model used to track progress (15%)

Internal scorecards are commonly populated using data from internal testing. So, scorecards can drive the need/desire for more (and better) accessibility testing. Likewise, the results of accessibility testing can be surfaced to management (and executive leadership) via the creation and distribution of an accessibility scorecard. With relatively low scores for these items (approximately a fifth to one third of respondents), there seems to be a huge opportunity to increase testing practices for accessibility.

When asked which management components are already part of their accessibility program, respondents could choose from the following (note: the order on the survey was quasi-randomized by the list arbitrarily alphabetical, but the items below are in descending order of response rate, given in parentheses):

- Centralized Accessibility Team (50%)
- Design and authoring practices for accessible content and media (50%)
- Internal accessibility portal or repository of resources for employees (47%)
- Use of external vendors and consultants for accessibility expertise (47%)
- Support for employees to pursue accessibility-related certifications and specialties (39%)
- Engagement with the Disability Community (local, national or international groups) (34%)
- Established Employee Resource Group (ERG) related to disability (26%)
- Accessibility leaders integrated into each team across the organization (21%)
- Proactive recruitment of employees with disabilities (15%)
- Periodic benchmarking using the Disability Employment Index (DEI) (10%)
- Employee development programs for Accessibility-related career paths (7%)
- Other (7%)

While we found that each component was mentioned ranging from seven to fifty percent of the time, the more interesting observation from the data was the *number* of components mentioned by each respondent:



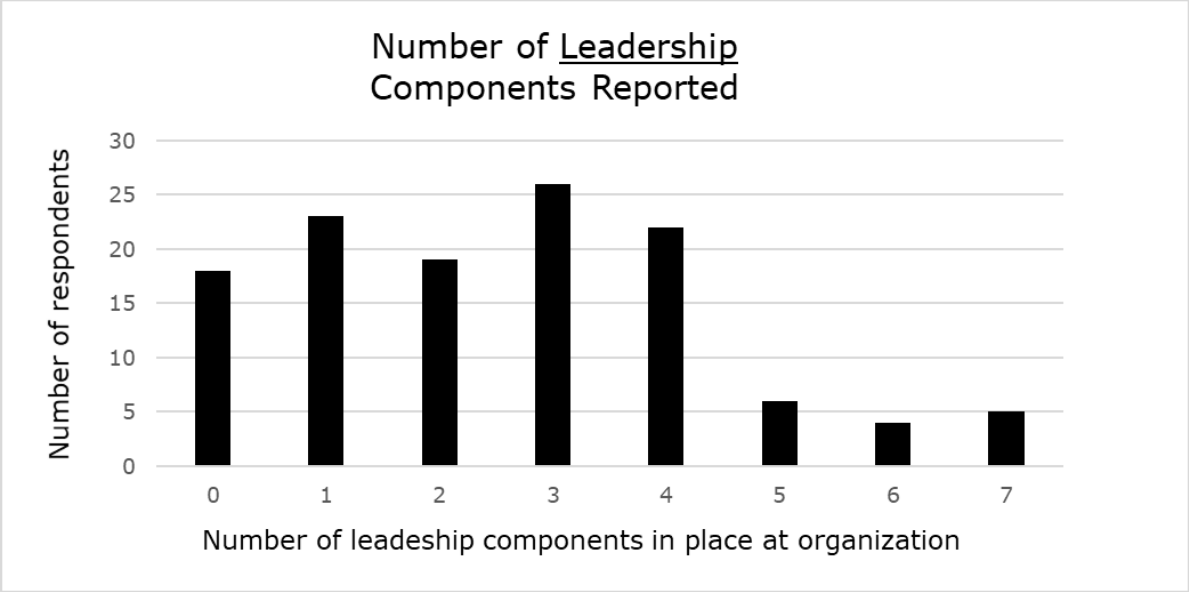
The median number is four out of fifteen possible responses. With half the respondents reporting that only four or less management components are in place, what does this tell us about implementation of accessibility programs as something that should happen organization-wide? This finding could mean that very few companies are building an organization-wide foundation to achieve accessible outcomes (ten or more management components were in place at 9 out of 136 organizations, or 6.6%), and at least half are only relying on a small number of management controls within their organization. However, to find a correlation to support this concept would require further study: there may be organizations that only require a small number of these to deliver great results, and the number of components may or may not turn out to be a determining factor in the success of an organization's accessibility program.

Note: This survey question also included components that would be specific to higher education respondents only. These components have been dropped from this analysis. The large organization respondents included people working in higher education institutions, and their responses about generic management components *are* included in the analysis.

Returning to leadership components, and the practical implications

The above observation on management components led us to return to the question of leadership components, and practical implications that we might find as a result.

Firstly, the median number of leadership components is only two. Again, this brings up a concern that accessibility is very often not being addressed widely, but instead it is being addressed as an isolated aspect:



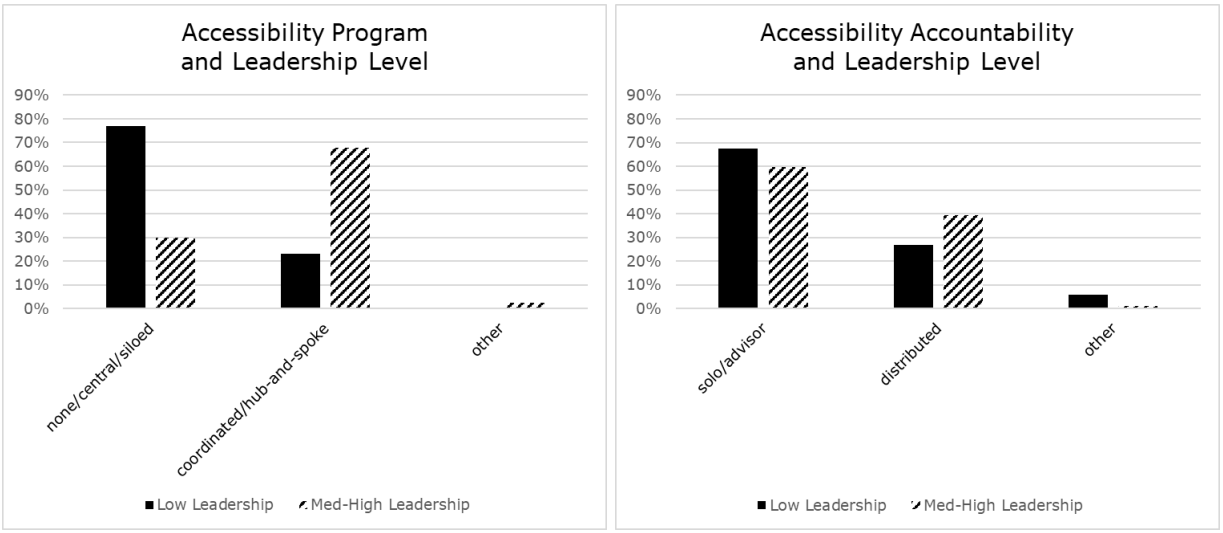
We divided the group of 136 'large organization' respondents based on whether there was 'low leadership', defined as no leadership or only one component in place; and 'medium-high leadership' defined as two or more components in place. We then examined how this plays out in practical terms in large organizations regarding two survey questions on the type of accessibility program, and who is considered 'accountable' for accessibility in the organization.

There are three formats of accessibility program that are generally considered to have low effectiveness for creating meaningful improvements to product and service accessibility, and creating an inclusive corporate culture (Law, 2010):

1. None (No Program - Individuals advocate for accessibility and perform accessibility-related work)
2. Central (Centralized Effort - Accessibility is addressed within one department only (e.g., the external facing web team conducts accessibility development and testing, but no other departments proactively address accessibility))
3. Siloed (Multiple, Siloed Efforts - Accessibility is addressed by multiple departments with little or no interaction with other departments.)

And there are two formats that are considered to be quite effective:

4. Hub and Spoke (Hub and Spoke Coordination - Accessibility work is performed within each department with the support and guidance of a central team to guide organizational goals and provide shared resources)
5. Coordinated (Multiple, Coordinated Efforts - Accessibility work is performed within each department, and these departments collaborate to meet organizational goals)



We grouped the question responses above (1-3, and 4-5) accordingly. We find that in 'low leadership' situations (n=52), the less effective accessibility program formats dominate (77% to 23%, respectively). The converse is true, where in 'medium-high leadership' situations, the numbers are (30% and 68% respectively).

In the same vein, we consider that when leadership commits to accessibility, accountability for accessibility should become:

1. Distributed (Each person or team is responsible for their own accessibility.)

However, accountability for accessibility is often placed on the shoulders of the person or teams who are regarded as *the* source of accessibility expertise within an organization:

2. Solo (A single person or team performs accessibility tasks on behalf of other departments (e.g., conducts their accessibility testing))
3. Advisor (A single person or team provides advice and guidance to support the accessibility work of other departments (e.g., accessible design consultation; an accessibility team occasionally audits for quality assurance))

Again, we grouped the responses (1, and 2-3) respectively. However, what we find here is that it doesn't seem to matter whether you are working under a 'low' or 'medium-high' leadership situation. Accountability for accessibility, even in medium-high leadership situations rests on individuals and teams of working solo or in an advisory capacity. We hypothesize that this is due to the specialized skills and knowledge required to properly design, develop and test accessible solutions. Until there is wide-spread knowledge of how to achieve accessibility (in people holding various job roles), a team of expert advisors and, potentially, implementors is often necessary. (Note: This represents a global skills gap, one of the key challenges being addressed by the IAAP through partner training programs and its own professional certifications and specialty certificate programs.)

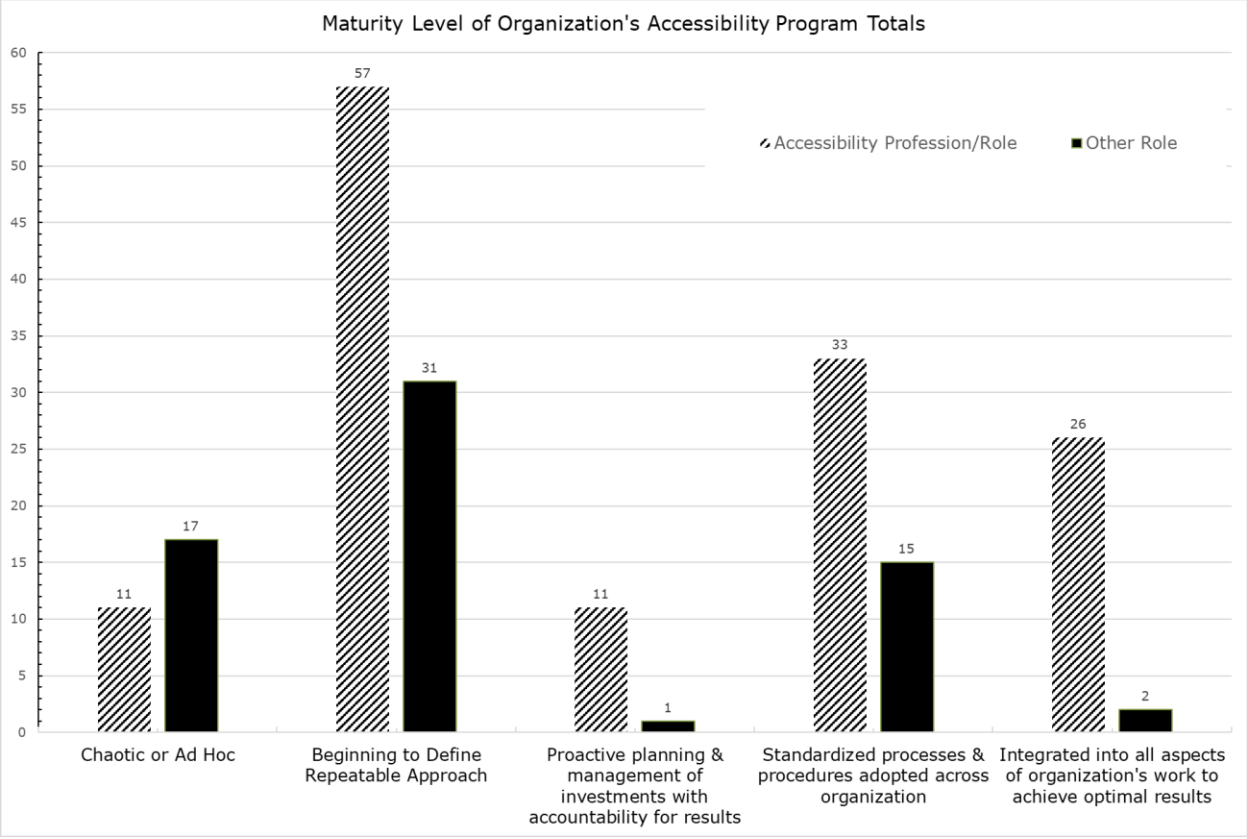
The ideal situation is for everyone in an organization to be held accountable for accessibility and have at least basic skills needed to contribute to that outcome (as they do for quality or security).

(The Accessibility Switchboard, 2018c) This is not the prevailing situation today for accessibility. In two thirds of the cases, accountability for accessibility is not a shared accountability across large organizations.

SECTION 2: All organizations, and accessibility professionals' impact on program maturity

There are a number of published resources for assessing an accessibility program's 'maturity' level' (e.g. NASCIO, 2015). The question underlying this type of assessment is how well the program is placed in order to positively impact accessibility in products and services produced by the organization, customer service, website accessibility, human resources, and so forth. (For this question, we are interested in activities regardless of the size of the organization, and therefore in Section 2 we cover results from all 205 respondents.) We did not specifically request that respondents conduct a formal assessment in order to respond to the survey, but we did ask for their estimation of which of five maturity levels their program fell into:

1. Chaotic or Ad Hoc
2. Beginning to Define Repeatable Approach
3. Proactive planning & management of investments with accountability for results
4. Standardized processes & procedures adopted across organization
5. Integrated into all aspects of organization's work to achieve optimal results



Only one of the 205 respondents marked this as 'unknown' for their organization. The most popular response was Level 2 ('beginning to define'), with 43% reporting this level. If we add to this figure the percentage of Level 1 ('chaotic') responses, 14%, the picture isn't good. We have 57% of respondents saying that their accessibility program maturity level is either 'chaotic' or 'beginning'.

Of interest to the survey team was whether the respondent described their primary job role as an accessibility specialist or accessibility practitioner (or similar), and would this have an impact on the maturity level that was achieved. Only at the lowest level ('chaotic') were the responses higher for the non-accessibility practitioners. As might be expected, at accessibility program higher maturity levels (2 through 5), the accessibility practitioner respondents' organizations were out-performing the organizations of the others.

SUMMARY: Management and Implementation of Testing within Accessibility Programs

Priority 1: What to test, and how to train?

No testing program can overcome a consistent absence of accessible design in the authoring of content nor in the design and development of software and services. The foundation of an effective testing practice is built upon a clear set of design and authoring practices for content and media, and a clear set of repeatable engineering practices for the design and development of accessible products and services. We saw that, more often than not, the design and testing, and related components were *far from prevalent* in organizations. With management components in-place that relate to testing, an organization can generate/possess data to evaluate and justify additional investments (e.g. better templates and tooling to help automate common tasks and reduce the cost of achieving accessibility at scale).

Training was an area in which we also see low uptake by industry. As organizations like IAAP continue to develop certification programs in accessibility, there will be a need to market to, and educate, the staff of large organizations on the historic under-representation of accessibility courses and materials in organization-wide training programs. As we have seen in Section 2, prior training to become an accessibility practitioner is typically associated with higher levels of program maturity, and therefore lower levels of organizational risk in this area.

Priority 2: Who tests? Accountability and shared responsibilities

As practitioners in the accessibility testing field, we need to be engaging and promoting testing throughout all organizational processes in order to make a real change. We recognize that there is a need for accessibility professional certifications because that expertise is needed in any large organization. We also need to recognize that accessibility will not become part of the organization's culture without training, tools and resources that help all staff who have a hand in the design and development and testing of information and communications technology.

The survey reveals that many large organizations have some form of accessibility map/plan, although there are still many who are quite unorganized, in terms of program maturity. We, as a

field, need to build on the starts that have been made and provide supporting resources to sustain accessibility. Shared accountability is one of the areas which left unsolved, will hinder sustainability. Siloed accessibility teams are the majority, we found, and this is a problem that must be urgently tackled by the field.

We can elevate the role that accessibility plays in organizations by making testing an organizational mandate with clear and measurable accountabilities ensuring testing for accessibility is not a one-time event, and making it an expected practice within every team.. A good lead-in would be to treat accessibility like security: no one gets away with releasing any product without considering security first.

Priority 3: What should the accessibility field focus on? Legal compliance versus Inclusion

Those who work in the accessibility field outside of mainstream business have traditionally focused on the societal benefits of accessibility. Pick up any book on universal design and you will read multiple compelling arguments for inclusion. Traditionally, we have not focused on legal reasons as motivators, although this has begun to change (e.g. Lazar, Goldstein & Taylor, 2015; Feingold, 2016). If legal compliance is the number one program motivator, as we have found, then we need to be providing events and resources that recognize this fact, and address how accessibility professionals and the organizations they work for can better manage their accessibility programs for risk reduction first, and elevate their levels of inclusion throughout their organizations thereafter. Like quality, safety and security, if accessibility is not recognized as a business imperative, it will never truly succeed.

Acknowledgments

The authors would like to thank other members of the Organizational Development Committee of the IAAP who participated in the development and delivery of the survey, and in particular Sharon Spencer (IAAP) for overseeing the management and delivery of the survey, and Kevin Hower (IAAP) for providing the initial summary of the results and handling technical aspects of the survey.

References

Accessibility Switchboard (2018a) Starting an organization-wide accessibility program: A guide for Higher Education Institutions. National Federation of the Blind Jernigan Institute. Available: https://www.accessibilityswitchboard.org/

Accessibility Switchboard (2018b) Allocation Q&A: How can I distribute the responsibility and accountability for accessibility?? National Federation of the Blind Jernigan Institute. Available: https://www.accessibilityswitchboard.org/

Accessibility Switchboard (2018b) Change Q&A: How can I overcome resistance to change in an organization-wide accessibility project? National Federation of the Blind Jernigan Institute. Available: https://www.accessibilityswitchboard.org/

Feingold, L. (2016). Structured Negotiation: A winning alternative to lawsuits. American Bar Association.

Hassell, J. (2015). Including your missing 20% by embedding web and mobile accessibility. British Standards Institution.

IED (Institute on Employment and Disability) (2018). 2016 Disability Status Report: United States. Cornell University. Available: http://www.disabilitystatistics.org/

Keates, S. (2007). Design for accessibility: a business guide to countering design exclusion. Mahwah, NJ: LEA.

Law, C.M. (2010). Responding to accessibility issues in business. Ph.D. Thesis. RMIT University. Available: https://researchbank.rmit.edu.au/view/rmit:6156

Lazar, J., Goldstein,, D. & Taylor, A. (2015). Ensuring digital accessibility through process and policy. Morgan Kaufman.

NASCIO (National Association of State Chief Information Officers) (2015). Policy Driven Adoption of Accessibility (PDAA) Vendor Self Assessment Tool. Available: https://www.nascio.org/PDAA

Sherbin, L. & Kennedy, J.T. (2017). Disabilities and Inclusion: US Findings. Center for Talent Innovation. Available: http://www.talentinnovation.org/publication.cfm?publication=1590

Copyright notice

The Forest and the Trees: Scaling for Enterprise-Level Digital Accessibility

Kathryn E. Weber-Hottleman

Information Technology Services, University of Connecticut
25 Gampel Service Road, Storrs, CT, USA 06269-1138
weberhottleman@uconn.edu

Abstract

As technology increasingly infuses daily life, it has become necessary to develop standards for regulating its accessibility. In January 2018, the United States formally appended the Web Content Accessibility Guidelines, version 2.0 (WCAG 2.0) to Section 508 of the Rehabilitation Act of 1973 (Section 508). This addition, combined with Dear Colleague Letters (DCLs) and case precedent, has broached the question of how organizations adhere to this standard. An action plan and, in many cases, an IT accessibility coordinator to implement the strategies in the plan are effective steps towards compliance and adoption. To address information and communication technology (ICT) accessibility issues, organizations can issue internal guidance, determine compliance issues, and train developers and content stewards. This paper is intended to assist in developing action plans to position an organization for ICT accessibility.

Introduction

As technology increasingly infuses daily life, it has become necessary to regulate its accessibility. The purpose of civil rights laws governing accessibility is to ensure that all people, regardless of race, sex, disability, or any other aspect of identity, have equal opportunities to participate in every facet of life. Specific laws, such as the Americans with Disabilities Act as Amended (ADA) and Section 504 of the Rehabilitation Act of 1973 (Section 504), make specific provisions for persons with disabilities. In January 2018, the United States formally adopted the Web Content Accessibility Guidelines, version 2.0 (WCAG 2.0) as its information and communication technology (ICT) accessibility standard and appended these guidelines to Section 508 of the Rehabilitation Act of 1973 (Section 508). This amendment, combined with Dear Colleague Letters (DCLs) and case precedent, has introduced the question of how organizations, whether federal, state, public, or private, can adhere to this standard. A developed action plan and, in many cases, an IT accessibility coordinator to implement the strategies in the plan are effective steps towards compliance and adoption.

The new Section 508 standards apply to federal agencies and offices, not to other organizations. However, the DCLs issued by the Department of Justice (DOJ) and the Office for Civil Rights (OCR) in 2010 (U.S. Department of Justice Civil Rights Division, U.S. Department of Education

Office for Civil Rights, 2010) and 2011 (U.S. Department of Education, Office for Civil Rights, 2011) make it abundantly clear that organizations such as higher education institutions are held to similarly rigorous standards. The DCLs' guidance was borne out in cases against University of California at Berkeley (U.S. Department of Justice Civil Rights Division , 2016), Harvard and MIT ("National Association of the Deaf, et al., Plaintiffs, v. Massachusetts Institute of Technology, Defendants", 2016, and "National Association of the Deaf, et al., Plaintiffs, v. Harvard University, et al., Defendants", 2016), and Miami University of Ohio ("Aleeha Dudley, Plaintiff, and United States of America, Plaintiff-Intervenor, v. Miami University, et al., Defendants.", 2016). Numerous other institutions also received notice of complaints filed with OCR and entered into resolution agreements. Drawing on these precedents, an organization's proactive plan involves steps for evaluating current accessibility levels, remediating issues, and positioning for future ICT accessibility. This paper will address these steps, referencing the Technology Accessibility Playbook published by the Federal CIO Council (2016).

Strategic Action Plan: Policies and Procedures

Play 2: Assess your Section 508 program maturity;

Play 3: Develop a Section 508 Accessibility Roadmap;

Play 4: Establish a Section 508 Policy

An initial step towards addressing ICT accessibility issues is often to issue internal guidance, typically in the form of a policy or policies. At the University of Connecticut (UConn), the Universal Website Accessibility Policy, along with the Providing Information in Alternative Formats Policy and the Policy Against Discrimination, Harassment, and Related Interpersonal Violence, lends weight to practical recommendations. Best practices, tutorials, and similar supports underpin the policy to guide content stewards on enacting it. This guidance is specific and draws on the WCAG 2.0 AA standards (World Wide Web Consortium (W3C), 2008). An excellent example of guidelines supporting a policy is the University of Missouri's (Mizzou) IT accessibility guidelines (University of Missouri, 2016). UConn has also developed marketing best practices and adapted social media best practices for accessibility (Ward & Ashe, 2017), which work in tandem with UConn's brand standards for accessibility, the non-discrimination policy, and the alternative formats policy.

Strategic Action Plan: Testing ICT Accessibility

Play 10: Conduct Section 508 testing

Another step towards ICT accessibility is identifying compliance issues. Often, this begins with assessment of websites and related materials. Assessing newly-created websites is perhaps the easier portion of the task, because safeguards can be added into the development process. For example, UConn has a request form that must be completed for a site to become live. The form asks specifically about accessibility, reminding developers to review the Universal Website Accessibility Policy and to measure the site against its standards. Websites based out of Aurora, UConn's instance of WordPress, also have a feature in which images lacking alternative text are "greyed out," appearing faded to an unusable point and clearly marking them as inaccessible. In

addition, developers have the opportunity to use tools on the IT Accessibility website, a resource with guidance on creating accessible websites and associated content as well as tools for monitoring accessibility throughout development.

The task of testing existing websites can be much more daunting. Existing websites were developed under a variety of standards, and there was and is great freedom to modify templates; coupled with the sheer number of websites, this situation makes accessibility assessment overwhelming and necessitates prioritization. UConn uses four primary criteria for establishing a site's priority for testing: the number of users, the audience direction, the transactional nature, and the frequency with which students with disabilities use the site. If a site has high traffic, such as UConn's Athletics site, it is highly prioritized. A site that draws a high amount of traffic from outside the UConn community, such as UConn's Jorgensen Center for the Performing Arts site, is also considered a top priority. Transactional sites also draw attention, such as UConn's Connecticut Repertory Theatre site, because patrons contract business there. Finally, sites used regularly by students with disabilities, such as the My Access portal for the Center for Students with Disabilities (CSD), are of paramount importance. UConn's starting point for testing existing websites is where these criteria intersect.

Because new sites are always developing and old sites are being refreshed, the accessibility testing schedule refreshes frequently. A new site meeting multiple criteria might trump an older site that meets only one criterion. Also, if a site is scheduled for a major update, there is no sense in reviewing it prior to the update, even if the site is a high priority. An example is the CSD site, which, though high priority, recently underwent a major update. Accessibility was considered throughout its construction, and it was tested for accessibility during its development phase.

Non-website ICT, such as hardware and software, is tested on an ad hoc basis. As a program or product comes under review for purchase or contract renewal, the same criteria determine its priority. For example, a campus-wide product must have accessibility considered during the request for proposals (RFP) process, while a program used by one or two individuals in a single department should *consider* accessibility. This will be discussed further in the Procurement section.

Accessibility Testing Tools

Play 6: Collaborate with the federal accessibility community

There are multiple methods for assessing a website's accessibility. Automated checkers, unearth general issues on a site. Some automated checkers are free, like WAVE by WebAIM, while others require a subscription. Automated checkers only touch the surface of compliance testing, though, and must be supplemented by manual testing. One manual method, the Trusted Tester method, is the culmination of the Department of Homeland Security's Trusted Tester Program. Other methods, taught by programs like WebAIM's accessibility training or the Disability Resources and Educational Services (DRES) IT Accessibility Badging Program out of the University of Illinois at Urbana-Champaign, follow similar procedures. This section will focus on the Trusted Tester method, as UConn's IT Accessibility Coordinator (ITAC) is a Trusted Tester (certification number 301201).

The Trusted Tester method utilizes Internet Explorer because of its compatibility with the Web Accessibility Toolbar, a testing tool developed and maintained by the Paciello Group (The Paciello Group, 2015). It also uses Jim Thatcher's favelets for web accessibility, available singly or as a toolbar for Firefox (Thatcher, 2013). To inspect Java, it recommends Java Ferret, an AccessBridge add-on; this was found to be unwieldy, and UConn recommends using Access Bridge Explorer for Java (GitHub, "Access Bridge Explorer", 2016). To inspect ARIA attributes, the Trusted Tester Program uses AI Inspect, a Windows software developer's kit (SDK) (Microsoft, 2018). Finally, it supplements contrast tools with Contrast Checker and Color Contrast Pal (Acart Communications, 2017 and GitHub, "Color Contrast Pal", 2016).

Additional testing tools include screen readers such as NVDA or JAWS. One cautionary note about utilizing screen readers as testing tools: Some screen readers, such as JAWS, are structured in a way that they rework code into its most accessible format. This means that they can frequently reconfigure digital content that is not accessible by design and read it as though it was accessible. Because of this feature, they are not accurate indicators of a site's accessibility. UConn recommends first testing content according to the method of choice, and then using a simple, free screen reader like NVDA to check inferred reading order and semantics. UConn's rule of thumb is to design sites in such a way that someone using a public computer and using the operating system's screen reader for the first time can easily navigate the site.

Reporting and Remediating Accessibility Issues

Play 11: Track and resolve accessibility issues

UConn uses the following process to identify accessibility issues and communicate them to relevant staff. A website's pages, applications, and content are tested for accessibility. For each page, a report is generated using a template provided by the Trusted Tester Program. This information is compiled into a summary report, with issues sorted based on who owns the content or object in question. UConn designed this report by adapting the Trusted Tester report template for greater readability. In the summary report, information is broken down into three categories: Template, site-specific, and content author.

At UConn, web development and design is typically handled by Information Technology Services (ITS), by designated staff in a given department, or by a third party vendor, while many individuals may contribute content to a site. By categorizing the report results, individuals and departments can determine what remediation can be effected in-house and what remediation must be addressed by web development and design staff. Many UConn sites are deployed using Aurora templates, which means that not all remediation can be done by individuals or departments but is instead referred to ITS. However, templates are often customized, either by the ITS web development and design team or by a department or individual. These site-specific theme issues may need to be directed to ITS, or they may be addressed within a department. Content issues are the responsibility of the content author, that is, anyone who is able to modify content on a given website.

In addition to dividing responsibility, UConn's template also provides space for a potential solution and a rationale. Solutions are broad suggestions in line with WCAG 2.0 AA, and

rationales share how the issue can impact users with disabilities. Both solutions and rationales are drawn from a UConn template.

Once the report is written, it is distributed to the relevant staff. ITS is usually involved for template and/or site-specific issues. The department that owns the site will naturally receive a copy of the report as well, as it will have to remediate content issues and potentially site-specific issues. If the site was developed by a vendor, the department will likely need to contact the vendor for template issues. Occasionally, depending on the department staff's level of technical expertise, the report may be condensed to target "low-hanging fruit." This decision is made based on considerations such as the staff's level of coding experience and vendor involvement. Accompanying the report is an invitation to meet with the ITAC and tailor solutions to the individual department's needs. Because websites have a variety of goals in addition to accessibility, it is necessary to customize the remediation plan to integrate with the website's objectives. These meetings also help clarify areas of responsibility for the remediation process.

Strategic Action Plan: Training Others

Play 12: Educate the workforce

A key part of remediating accessibility issues is training developers and content authors. Accessibility training can build on preexisting training; for example, at UConn, training for Blackboard and Aurora has accessibility training included. It is also possible to adapt and customize existing third party training, with permission. WebAIM, W3C, the Section 508 playbook, and vendors like 3Play Media and Blackboard all offer accessibility training.

Essentially, three basic trainings must be developed, targeting different content stewards: the average content author, Procurement, and IT. Content author training is basic and non-technical, focusing on quick, easy fixes. Techniques from this training can be applied to learning management system (LMS) content or to web content. Procurement training is more in-depth, focusing on questions to ask vendors and contractors. One discussion with procurement is about the timing for introducing accessibility into conversations with the vendor. Finally, IT training is highly technical and needs to be updated regularly to reflect changes in standards and in coding. It is focused on code and template development. Discussions include checking third party widgets and plugins for accessibility and potentially includes a discussion around involving the vendor with remediation strategies, if a third party product is under consideration. For example, UConn has historically submitted tickets to vendors regarding compliance issues.

The IT Accessibility Coordinator

Play 1: Establish a Section 508 Program Manager to lead compliance efforts

It is primarily the ITAC's responsibility to not only draft the plan but also to implement it. The ITAC sets priorities, determines areas of responsibility, and creates a system for monitoring accessibility. This means that the ITAC must be mindful of the sustainability and scalability of the action plan. Another piece of the ITAC's role is to build relationships with stakeholders. The ITAC is responsible for allotting remediation tasks and creating a global sense of teamwork. An

enterprise-level sense of supporting each other and working together is critical to develop an accessibility culture at an organization.

To manage this range of responsibilities, the ITAC has the following qualifications, with the caveat that much is learned on the job, as ICT accessibility is a constantly evolving field. The ITAC must be trained in accessibility testing, preferably through a recognized program like the Trusted Tester Program. He must have a thorough understanding of the ADA, Sections 504 and 508, state accessibility laws, and WCAG 2.0. Experience applying these laws and guidelines, in a disability services office or an office of institutional equity, is also necessary. Furthermore, familiarity with web development and design is helpful for proposing solutions and discussing accessibility standards with IT teams. Finally, the ITAC needs both a strong sense of collaboration and a high level of independence. The ITAC is likely to intersect with many departments but to primarily work on his own. This gives the ITAC the independence to strategically build relationships and form his own "team."

There are many areas where the ITAC could be situated, like the disability services office, IT, or the office of institutional equity. At UConn, the ITAC is housed in ITS. There are several advantages to this. First, the ITAC has a universal role, and this is clearly demonstrated because UConn's ITS serves the entire institution. Being housed in ITS also minimizes preconceived ideas about the ITAC's role, which might not be the case if he was situated in the disability services office or the office of institutional equity. ITS also affords the ITAC many resources, departmentally and in the form of professional development opportunities. Finally, the ITAC can focus solely on ICT accessibility, without the responsibility of a caseload.

There are some challenges. At UConn, the ITAC currently works on his own, as previously noted. There is also a steep learning curve, both for the ITAC about ITS and vice versa. Accessibility was obviously included throughout the design and development processes of all ITS products, including LMS design, but naturally web development and design teams do not spend months on in-depth research regarding web accessibility standards and changing regulations. Conversely, web development and design was part of the ITAC's training but was certainly not a focus. Also, because the ITAC is a new position for the University, flexibility is required from University leadership and from the ITAC. At UConn, frequent input from ITS leadership and from stakeholders around campus has helped shape priorities, the accessibility timeline, and expectations. The ITAC, in return, incorporates priorities and adjusts timing as necessary.

Stakeholders

Play 5: Develop a Section 508 Program Team

Regardless of where the ITAC is housed, there is much to learn as he and the organization work together. ICT accessibility is not solely the responsibility of the ITAC; it requires support from top-level leadership to make it a priority and to infuse it into the organization's culture. Perhaps the best possible situation is when the organization has already identified accessibility as an issue and the ITAC can present possible solutions and training, as occurred at UConn.

The many stakeholders in institutional accessibility can be considered in two groups, the visionaries and the realizers. The visionaries shape the organization's direction and are responsible for top-down decisions. They also govern policies at the organization. Likely, the ITAC's position came about as a result of the visionaries. The realizers are the hands-on problem-solvers. They enact the remediation changes and carry out the policies. They are likely to work on a daily basis with the ITAC and will often be a sounding board for concerns or proposed solutions.

Both groups must work together to achieve the broad goal of an accessibility culture across the entire organization. Without the buy-in of leadership, there is no official sanction for accessibility changes. Without support from realizers, there are no concrete solutions. Working together, they can promote an institutional mindset where accessibility is a standard initial consideration for any ICT that is created or purchased.

Procurement

Play 8: Integrate accessibility needs into market research and acquisition processes;

Play 7: Integrate accessibility needs into requirements and design processes

There are several stakeholders besides Procurement included in the question of procuring accessible ICT. First is ITS, where the ITAC is situated. The compliance office and disability services office may be pertinent, depending on the intended use of the product or service requested. Finally, the department requesting or purchasing the ICT is involved. Working with Procurement guides all requests for ICT to consider accessibility and to consider accessibility in institution-wide purchases.

The ITAC prioritizes requested products and services based on the following criteria. First is a "heat map," considering the product's cost and the number of the product's users. As both cost and number of users increase, the priority increases. By these criteria, enterprise-level products will always be top priority. A product to be used by one individual or department will be lower priority. A second criterion is that student or public use will always be prioritized. If a product is necessary for a student or a member of the broader community to engage with UConn, then it must be an accessible product. Third, products essential for class or work are prioritized over non-essentials, for similar reasons. Finally, transactional products are prioritized over non-transactional products. For example, ticketing systems like those used by IT, Jorgensen Center for the Performing Arts, and Athletics are prioritized over non-transactional sites. This process is very similar to how websites are prioritized for testing.

Discussions with Procurement raise accessibility questions for the requesting department to consider. Incorporating accessibility questions into these initial discussions potentially alleviates the stress of having to remediate products, websites, and other ICT for accessibility. In this regard, it is financially responsible, because it costs time and money to remediate, which can be detrimental to end users. For example, a vendor may not make changes quickly enough to actually help a student, or it may not remediate changes without additional cost. Proactivity also furthers the campus culture of accessibility, where ICT accessibility is not an afterthought but is built in from the beginning of UConn's relationship with a product or vendor.

In order to accomplish this, others besides the ITAC must be able to evaluate a product's accessibility, at least on a high level. Training on how to read documents like a VPAT is available through W3C and WebAIM, as well as through vendors like SiteImprove Academy. In-house training can also be developed, directing departments to ask questions about a VPAT. Training and discussions cause departments to think critically about accessibility and to recognize the needs of users with disabilities.

There are limitations to this process. One limitation is the amount of time required to thoroughly test a product. To test a single product can take several days, if one is able to focus on testing. With other responsibilities, it can take over a week or longer. If an organization cannot provide additional resources, such as student workers, to shoulder some of the testing load, the ITAC will likely become a bottleneck for the procurement process. A potential solution is to test products following purchase. The drawback here is that a product may be found to be inaccessible, despite the good-faith efforts of departments and Procurement to ascertain its accessibility. Without the ITAC's expertise and work focus, it is possible an inaccessible product may still be purchased.

Conclusion

How does this affect an organization's day-to-day operations? Web development and design and LMS development now all consider accessibility. Content authors and stewards have begun to ask themselves about a page's accessibility or about captioning videos. Procurement has begun to involve the ITAC in enterprise-level product discussions. Departments have begun earmarking funds for accessibility-related tools and features. As an institution, UConn is shifting towards a culture where accessibility is not only embraced but worked into the fabric of its operations. Reflecting on the significant changes the ITAC has seen, in only six months, it is noted that the majority of the time, people only lacked tools to engineer accessibility. The mindset was there, the conversations were already happening, but no one knew how to standardize the testing process or establish a repository of solutions. Given these tools, the organization is moving swiftly to a place where all users can experience UConn.

References

Acart Communications. (2017). Contrast Checker. Retrieved August 24, 2018, from https://contrastchecker.com/

Aleeha Dudley, Plaintiff, and United States of America, Plaintiff-Intervenor, v. Miami University, et al., Defendants. (United States District Court, Southern District of Ohio December 14, 2016) (ADA.gov, Dist. file).

Berkowitz, A. (2018, July 14). *Create accessible navigation from scratch with WordPress*. Lecture presented at WPCampus in Washington University, St. Louis, MO. Retrieved August 24, 2018, from https://2018.wpcampus.org/schedule/create-accessible-navigation-with-wordpress/

Federal CIO Council. (2016). *Technology Accessibility Playbook: How to build an effective Section 508 Program* (United States of America, Federal CIO Council).

GitHub. (2016, June 27). Access Bridge Explorer. Retrieved August 24, 2018, from https://github.com/google/access-bridge-explorer

GitHub. (2016, August 3). Color Contrast Pal. Retrieved August 24, 2018, from https://toolness.github.io/color-contrast-pal/

Kincaid, J. M. (2018, June 13). *Driverless Cars & the ADA: Where Are We Without Federal Oversight In The Era Of Trump?* Lecture presented at Postsecondary Disability Training Institute in Sheraton Inner Harbor, Baltimore, MD.

Microsoft. (2018, May 30). Inspect. Retrieved August 24, 2018, from https://docs.microsoft.com/en-us/windows/desktop/WinAuto/inspect-objects

National Association of the Deaf, et al., Plaintiffs, v. Harvard University, et al., Defendants (United States District Court District of Massachusetts February 9, 2016) (Civil Rights Education and Enforcement Center, Dist. file).

National Association of the Deaf, et al., Plaintiffs, v. Massachusetts Institute of Technology, Defendants (United States District Court, District of Massachusetts November 4, 2016) (Civil Rights Education and Enforcement Center, Dist. file).

Thatcher, J. (2013, February 22). Favelets for Checking Web Accessibility. Retrieved August 24, 2018, from http://jimthatcher.com/favelets/

The Paciello Group. (2015, May 20). Web Accessibility Toolbar (WAT). Retrieved August 24, 2018, from https://developer.paciellogroup.com/resources/wat/

University of Missouri. (2016, September 20). Web standards. Retrieved August 24, 2018, from https://accessibility.missouri.edu/standards/web-standards/

U.S. Department of Education, Office for Civil Rights. (2011, April 4). Dear Colleague Letter [Press release]. Retrieved from https://www2.ed.gov/about/offices/list/ocr/letters/colleague-201104_pg3.html

U.S. Department of Justice, Civil Rights Division. (2016, August 30). The United States' Findings and Conclusions Based on its Investigation Under Title II of the Americans with Disabilities Act of the University of California at Berkeley, DJ No. 204-11-309 [Letter to University of California Berkeley].

U.S. Department of Justice, Civil Rights Division, & U.S. Department of Education, Office for Civil Rights. (2010, June 29). *Joint "Dear Colleague" Letter: Electronic Book Readers* [Press release]. Retrieved from https://www2.ed.gov/about/offices/list/ocr/letters/colleague-20100629.html

Ward, S. B., & Ashe, C. E. (2017, May 1). The Student Affairs Accessible Social Media Toolkit. Retrieved from https://drive.google.com/open?id=0B_-ZfCP7Aqp2RFRaRFQ2U1RTYXM

World Wide Web Consortium (W3C). (2008, December 11). Web Content Accessibility Guidelines (WCAG) 2.0. Retrieved August 24, 2018, from https://www.w3.org/TR/2008/REC-WCAG20-20081211/

Copyright notice

Web Accessibility: Where is the Non-Profit Sector?

Brittani S. Washington

Department of Computer and Information Sciences
Towson University, Towson, MD, United States
bwashington@towson.edu

Jinjuan Heidi Feng

Professor, Department of Computer and Information Sciences
Towson University, Towson, MD, United States
jfeng@towson.edu

Abstract

This paper analyzes 48 websites from a group of non-profit organizations located in the United States of America. The websites were evaluated during the summer of 2018. The results show that the non-profit sector have not taken the proper steps to implement the accessibility requirements. The findings provide insight to policy makers and accessibility practitioners on the current status of accessibility compliance in the nonprofit sector, which may help initiate conversation or projects to address the accessibility challenges that are faced.

Introduction

As the world is evolving provisions are being made for every citizen, especially those with disabilities. The world we live in has developed and incorporated the use of assistive technologies; such as screen readers and voice recognition software. If websites aren't designed properly it makes information difficult to access for those with disabilities. It creates additional barriers for them to function in a world where they are supposed to be considered as equals. The United States Department of Justice (DOJ) Civil Rights Division has implemented laws and regulations that govern Title II (state and local government services) and Title III (public accommodations and commercial facilities); however, non-profit organizations are not clearly defined within those laws and regulations unless they are deemed a public accommodation organization.

In terms of web accessibility there are two main resources that are available for web developers that could assist them with making their websites more accessible, Section 508 Standards and the Web Content Accessibility Guidelines. The Section 508 Standards are standards put in place for federal agencies to follow and are not effectively enforced. Since the DOJ wasn't effectively enforcing the Act, many private parties and companies begin to expose the lack of care. The Web Content Accessibility Guidelines (WCAG) help designers make web pages as accessible as

possible to the widest range of users, including users with disabilities (Accessibility of State and Local Government Websites to People with Disabilities, 2008). As it pertains to the current WCAG, compliance in the non-profit sector is very limited. We conducted a compliance test for around 50 selected websites to help understand the status of web accessibility in the non-profit sector.

Research Method

48 non-profit organizations were selected based on 3 selection criteria: (1) Geographical area; (2) Types of organization; and (3) the number of webpages contained in the website. A hierarchical structure was used to categorize how the websites were categorized (Figure 1).

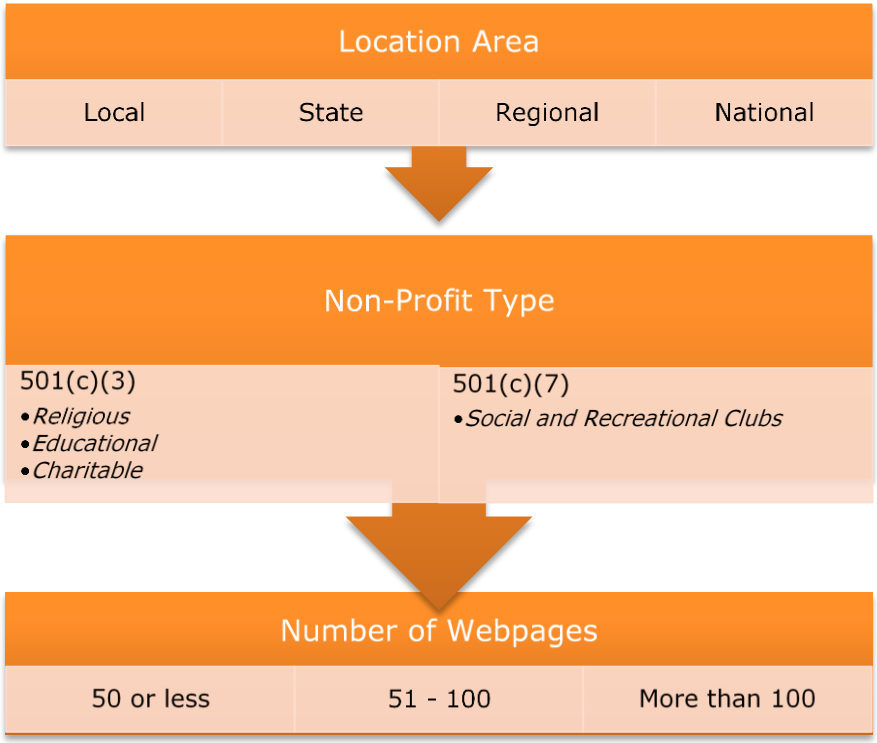


Figure 1 – Hierarchical Structure

First, geographical areas were categorized into 4 levels: local, state, regional, and national. Then two types of nonprofit organizations were identified for use, 501(c)(3) and 501(c)(7). C3 organizations were divided into 3 categories: religious, educational, and charitable. C7 organizations focused on social and recreational clubs. The rationale behind this selection is that these types of organizations have a large influence on the communities. Further, websites with different number of web pages were selected and grouped as follows: 50 or less, 51 – 100, and more than 100.

Evaluation Strategy

Automated Testing

After the websites were selected, they were analyzed using the Functional Accessibility Evaluator (FAE) automated tool that was developed by Lazar, Williams, Gunderson, and Foltz during the research on Monitoring U.S. Federal Website Accessibility (Lazar et. al. 2017; Gunderson et al. 2017). Utilizing the Website Implementation Score, FAE allowed each website to be evaluated based on 132 rules derived from the principles of WCAG 2.0. FAE computes an implementation score for each individual rule (RIS) using the following equation, in which P represents the number of passed items; F represents the number of failed items; and MC represents the number of items requiring manual check (Lazar et. al. 2017):

$$RIS = \frac{P}{P + F + MC} \times 100$$

An implementation score for a website is an average of the implementation scores of the individual rule results; A score 0 being the lowest, meaning the rule wasn't implemented on any of the webpages; A score of 100 being the highest, meaning the rule was successfully implemented on all necessary webpages. The FAE tool used for evaluation can analyze a maximum number of 25 webpages for each website.

The websites were also analyzed using the automated WAVE Web Accessibility Evaluation Tool (WAVE). WAVE wasn't developed to determine if a website's accessibility passes or fails in its entirety. It is used to identify errors or compliance issues found in the Section 508 and WCAG 2.1 guidelines. The errors or issues noted were reported in 6 categories (e.g., errors, alerts, features). We added the number of instances reported in 6 categories to calculate the total number of accessibility issues reported for each website.

Results

FAE Website Implementation Score

A One-Way Analysis of Variance (ANOVA) test with the FAE website implementation score as the dependent variable and the type of geographical area as the independent variable suggests that there is a significant difference between the websites of different geographical features ($F(3, 44) = 3.93$, $p < 0.05$). The post hoc Tukey's test shows that the website implementation scores of the state level organizations are significantly lower than those of the local organizations and national organizations (Figure 2).

We also conducted two One-Way Analysis of Variance (ANOVA) tests with the FAE website implementation score as the dependent variable and the type of organizations and page numbers as independent variable respectively. The tests found no significant impact on the implementation scores by either the type of organizations ($F(3, 44) = 2.74$, n.s.) or the number of webpages ($F(3, 44) = 0.52$, n.s.).

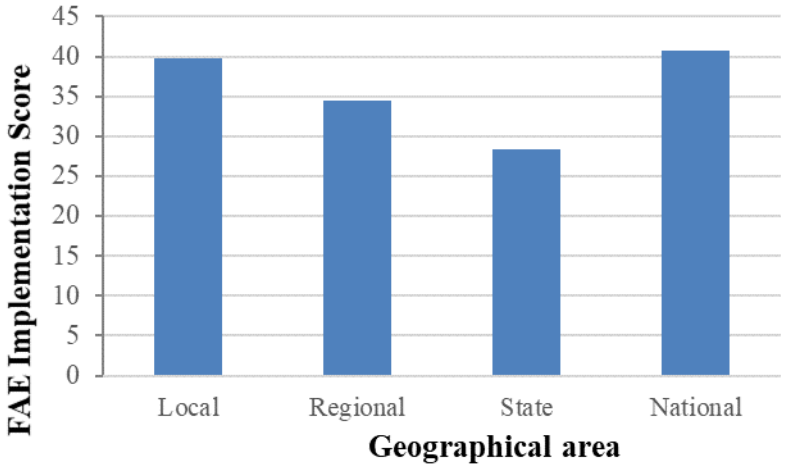


Figure 2 – FAE website implementation score by geographical area

Number of accessibility issues reported by WAVE

We conducted two One-Way Analysis of Variance (ANOVA) tests with the number of accessibility issues reported by WAVE as the dependent variable and the type of geographical areas and organizations as independent variable respectively. The test found no significant impact on the number of accessibility issues by either the type of geographical areas ($F(3, 44) = 1.08$, n.s.).or the type of organizations ($F(3, 44) = 0.41$, n.s.).

A One-Way Analysis of Variance (ANOVA) test with the number of accessibility issues as the dependent variable and the range of page numbers as independent variable suggests that there is a significant difference between the websites with different sizes ($F(3, 44) = 3.59$, $p < 0.05$). Small and medium size websites reported lower number of accessibility issues than websites with more than 100 pages (Figure. 3).

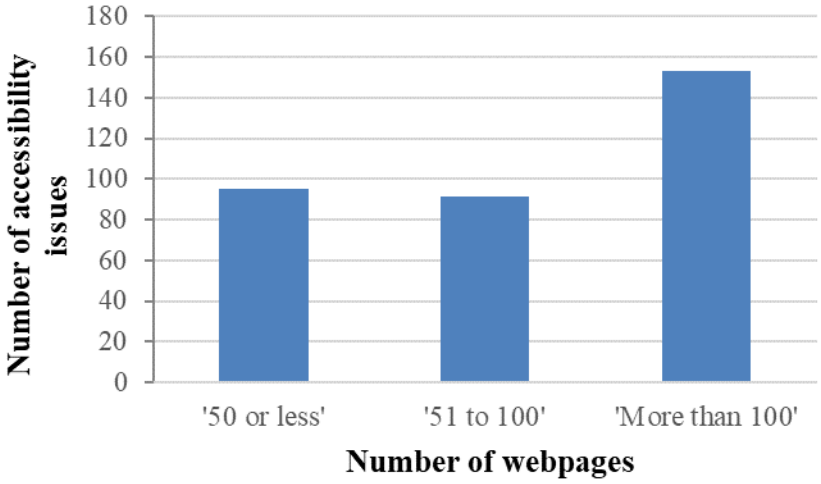


Figure 3 - The impact of website size on the number of accessibility issues reported by WAVE

Individual Rule Results

This section identifies the top rule group violations by determining the frequency of occurrence of a rule being violated across the websites. After all of the top rule group violations were determined, the top 5 most frequent individual rule violations were selected to be analyzed at the individual rule levels across all websites.

Overview

The FAE Tool groups the 132 rules into 12 groups: audio/video, forms, headings, images, keyboard, landmarks, links, site navigation, styles/content, tables, timing, and widgets/scripts. Out of those 12 rule groups, audio/video, keyboard, and timing did not return a value as it relates to violations since those rule groups have to be manually checked. As a result, 9 rule groups are available to be evaluated; however, due to space limit only the top 5 ranked rule groups will be analyzed which equals to 56% of the groups where rule violations are applicable. As illustrated in figure 5, those five groups are (in order from left to right) landmarks, links, styles/content, forms, and headings.

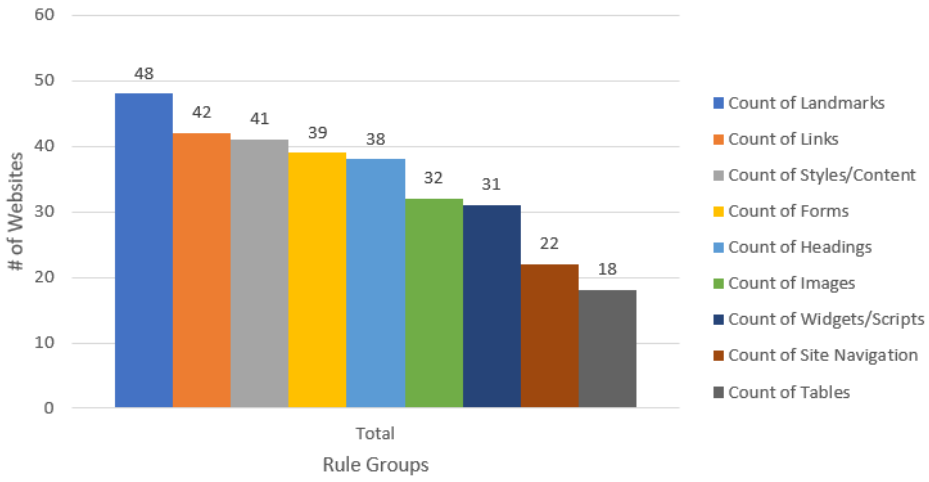


Figure 4 – Number of websites with accessibility issues in a specific rule group

Out of those 5 rule groups the top 5 individual rules that were analyzed are Landmark 2, Link 1, Heading 5, Color 1, and Landmark 1. The frequency of occurrence was calculated by counting if an individual rule violation existed on a website. For example, in the table below the Landmark 2 rule violation affected 44 of the 48 websites analyzed.

VIOLATION	# OF OCCURRENCE
COUNT OF LANDMARK 2	44
COUNT OF LINK 1	42
COUNT OF HEADING 5	38
COUNT OF COLOR 1	36
COUNT OF LANDMARK 1	33
COUNT OF CONTROL 1	33
COUNT OF FRAME 2	22
COUNT OF LANDMARK 17	20

Landmark 2

The Landmark 2 rule is associated with the Landmarks rule group. This rule governs that all content must exist inside of the proper container elements, which are created to provide ways to organize content on webpages. If like content are grouped together it's easier for those with assistive technologies to find the information that they need. Out of the websites analyzed, 44 of them have violations associated with this rule. The relative frequency of the Landmark 2 rule violation occurring equals to approximately 92% of the websites that were analyzed in the data set, which warns that this is a common problem within non-profit organizations.

Link 1

The Link 1 rule is associated with the Links rule group. This rule governs that the accessible name link must describe its target or purpose in order to lower the level of frustration for users of assistive technologies. The use of "Click Here" name links makes it extremely frustrating for those with assistive technologies. If a screen reader reads, "**Click Here** to view job opportunities", the user has to revert back to the beginning of the sentence to attempt to understand where to click. If a screen reader reads, "To learn more about job opportunities with our company, visit **Careers**", and the user is mainly using their screen readers to scan links for employment they will easily find the careers section of the website. Out of the websites analyzed, 42 of them have violations of this rule where the link names do not match the target area. The relative frequency of the Link 1 rule violation occurring equals to approximately 88% of the websites that were analyzed in the data set.

Heading 5

The Heading 5 rule is associated with the Headings rule group. This rule governs that all headings must be properly nested on a page. Establishing headings helps users understand the structure of information on a webpage. It also allows for easy interpretation of the content for screen readers. Out of the websites analyzed, 38 of them have violations associated with this rule. The relative frequency of the Heading 5 rule violation occurring equals to approximately 79% of the websites that were analyzed in the data set.

Color 1

The Color 1 rule is associated with the Styles/Content rule group. This rule governs that the text content must exceed the color contrast ratio. For example, higher color contrast for text makes it a lot easier for those who are visually impaired to read the content. Out of the websites analyzed, 36 of them have violations associated with this rule. The relative frequency of the Color 1 rule violation occurring equals to approximately 75% of the websites that were analyzed in the data set.

Landmark 1

The Landmark 1 rule is associated with the Landmarks rule group. This rule governs that each page must have at least one main landmark used to identify the main content. With this being implemented, it allows users to skip to the wanted content of a website and bypassing the unwanted content. Out of the websites analyzed, 33 of them have violations associated with this rule. The relative frequency of the Landmark 1 rule violation occurring equals to approximately 69% of the websites that were analyzed in the data set.

Discussion

The overall findings from the data collection identified that all websites analyzed have rules that are not implemented, meaning the site's developers might not understand the accessibility requirements of the rules or did not consider accessibility in the design of the website. Also, it shows that all websites analyzed required manual checks, meaning markups have been identified by the FAE tool that needs to be reviewed to determine if accessibility requirements have been met.

The analysis of the FAE implementation scores suggests that accessibility violations prevail in non-profit websites. The geographical areas significantly influence the FAE implementation scores. The state level organizations have significantly lower scores than those of the local and national organizations. This suggests that the local and national level organizations might have better implementation of their content. However, the WAVE analysis found no significant difference in the total number of accessibility issues among the different geographical areas. So the significant difference identified by the FAE analysis needs to be further validated with more websites.

Neither the FAE analysis nor the WAVE analysis found significant difference among the different types of non-profit organizations, suggesting that the accessibility challenges and lack of training exist across different types of organizations.

The WAVE analysis found significant difference in the total number of accessibility problems among websites with different number of pages, with the small and medium size websites reporting lower number of accessibility issues than websites with more than 100 pages. This finding is well expected in that fewer pages usually contain fewer content items and therefore, lower number of accessibility violations.

This preliminary study provides initial insights regarding the current web accessibility status of the non-profit sector and suggests prevailing accessibility problems across all types of non-profit websites. However, due to the limited number of websites analyzed in the study, it does not provide a complete picture of the problems. The difference between the number of webpages analyzed between the FAE tool and the WAVE tool made it difficult to compare the results of the two analysis. Also, the FAE tool does not have the capability to analyze features that require manual checks. It only has the capability to note if they are present. As a result, the next steps would be to develop a manual checklist tool comprised of the rules identified from the FAE tool’s results.

The finding of this study helps bridge the gap between accessibility and technology in the non-profit sector. The study exposes the lack of implementation as it relates to main accessibility features. With a better understanding of the order of the severity of the accessibility deficiencies in the non-profit sector and the proper variables, non-profit organizations can determine their next steps as it relates to initiating a website refresh or a website redesign.

References

Accessibility of State and Local Government Websites to People with Disabilities. (2008, October 9). Retrieved September 15, 2017, from https://www.ada.gov/websites2.htm

Lazar, J., Gunderson, J., and Golderfeld, Z. (2017). Federal Website WCAG 2.0 Compliance and Accessibility Trends. In Proceedings of 2017 ICT Accessibility Testing Symposium: Automated & Manual Testing, WCAG 2.1, and Beyond. 27-34.

Williams, V., Gunderson, J., & Foltz, T. (2017). Investigating the Potential of a Dashboard for Monitoring US Federal Website Accessibility. In Proceedings of the 50th Hawaii International Conference on System Sciences.

Smith, J., & Whiting, J. (2001). WAVE Web Accessibility Evaluation Tool. Retrieved from http://wave.webaim.org/

University of Illinois. (2018). Functional Accessibility Evaluator 2.0. Retrieved from https://fae.disability.illinois.edu/

Copyright notice

Symposium Committee

Chairs

Symposium Chair:	Chris M. Law
Symposium Co-Chair:	Norman Robinson
Marketing & Outreach Chair:	Laura Renfro
Awards Chair:	Andrew Nielson
Program Chair:	Chris M. Law
International Engagement Chair:	Shadi Abou-Zahra
Software Testing Industry Engagement Chair:	Karl Groves
Competition Chair:	Erin Kirchner-Lucas
Presentations Chair:	Peter McNally
Workshops Chair:	Matt Feldman
Mobile Testing Sub-Committee Chair:	Gian Wild

Committee Members

Shadi Abou-Zahra
Strategy and Technology Specialist, W3C Web Accessibility Initiative (WAI)

Jennison Asuncion
Engineering Manager (Accessibility), LinkedIn

Jon Avila
Chief Accessibility Officer, Level Access (formerly SSB Bart Group)

Cristopher Broyles
Chief Accessibility-Solutions Officer, American Foundation for the Blind

Jennifer Chadwick
Lead Accessibility Strategist (North America), Siteimprove

Wendy Chisholm
Principal Accessibility Strategist, Microsoft

Katherine Eng
Senior ICT Accessibility Specialist, U.S. Access Board

Matt Feldman
Director of Services, Paciello Group, A VFO Company

Karl Groves
Founder and President, Tenon.io

Jon Gunderson
Coordinator, Accessible Information Technology Group, University of Illinois at Urbana/Champaign

Sunish Gupta
Lecturer, Northeastern University

Katie Haritos-Shea
Vice President of Accessibility, EverFi

Allen Hoffman
Deputy Executive Director, The Office of Accessible Systems & Technology, Department of Homeland Security

Erin Kirchner-Lucas
Accessibility Coordinator, RedShelf

Mark Lapole
Lead Product Manager, Accessibility, eBay

Chris M. Law
Organizational Accessibility Consultant, Accessibility Track

Christine K. Loew
Director, Accessibility for Digital Assessments, The College Board

Amy Mason
Access Technology Specialist, National Federation of the Blind Jernigan Institute

Earl McJett
Section 508 Coordinator, Federal Deposit Insurance Corporation

Peter McNally
Senior Consultant, User Experience Center, Bentley University

Eduardo Meza-Etienne
President & CEO, Meza Consulting, LLC

Karen Moyes
Section 508 Coordinator, Westat

Alyson Muff
Senior Technical Analyst, ICF International

Andrew Nielson
Senior Program Manager, New Editions Consulting, Inc.

John Rempel
Quality Control & Training Specialist, AMAC at Georgia Tech

Laura Renfro
Senior Accessibility Lead, Renfro Consulting

Norman Robinson
Agency Section 508 Program Manager, Department of State

Madeleine Rothberg
Senior Subject Matter Expert, National Center for Accessible Media / WGBH

Cyndi Rowland
Director, WebAIM, Utah State University

Janet Sedgley
Founder, TesselLearn

Korey Singleton
Assistive Technology Initiative Manager, George Mason University

Dominique Wheeler
ISD/508 Compliance Specialist, Sevatec, Inc.

Gian Wild
Founder, Owner and CEO, AccessibilityOz

Terri Youngblood
President, Accessible Systems, Inc.

Author Index