**Development Accessibility Testing 101**

Good morning everybody and thank you all for joining today's call. Let’s jump right into it….

Great designers and developers are inclusive thinking ones, so it is very important that we all test our work and ensure that it is as accessible as possible for everybody.

Current estimates indicate that over 15% of the population have a disability of some kind. That’s roughly 1 billion people or to put it in a better perspective, this number is about the size of China. (And, I might add, in an effort to get our priorities straight, that this is likely a much larger number than IE11’s market share). So we have an ethical and moral obligation to do our small part to help folks out. Especially where it applies to users being able to learn more about a company and seek employment there.

Speaking of employment, many, if not all our clients, tout themselves as equal opportunity employers, so they expect us to reach 100% of their prospective candidates, not 85%, whether we are specifically asked to do so or not. Equal Opportunity Employment begins with TMP.

For developers, this means that every single one of us must smoke test our work. This involves automated and manual testing, so let’s get into it. First up, some automated tools:

**Automated Testing**

**axe**

There are several automated testing tools that I use to test your sites. The most common one I leverage is a browser plugin called aXe (by a great company called Deque). Many of the issues that are cited during testing, stem from this plugin. After this call, I want you all to install it and start using it so that we can start speaking a common language when it comes to accessibility issues.

<https://www.deque.com/axe/>

*Demo*

**axe CLI**

axe comes in a command line flavor, as well. This is particularly useful if you wish to scan multiple URLS and dump the results into a text or JSON file.

<https://github.com/dequelabs/axe-cli>

*Demo*

**Lighthouse**

Google Chrome’s Devtools include built in auditing tools. Besides being able to test for performance, SEO and best practices, you can also test for accessibility issues.

Google’s accessibility auditing actually leverages axe core, which powers axe, mentioned above, so many of the results will be similar.

*Demo*

**WAVE**

The WAVE toolbar plugin also produces solid results. Accessibility can be very subjective at times, so each tool in our arsenal may interpret WCAG guidelines a little differently. This is not a bad thing though, as I believe it is good to always get a second opinion.

There is another, sadder, reason why we should test with this particular tool and that is because it has become a favorite tool for law firms seeking to scare companies into litigation (and hopefully win a settlement), so if we come up clean in this tool, we lower the risk of that happening. In the past two years, our industry has seen a large uptick in lawsuits. Many of them are legitimate issues concerning digital inclusion and accessibility, but many are not. These are often called “drive-by lawsuits”.

<https://wave.webaim.org/extension/>

*Demo*

**Web Development Toolbar**

The Web Developer Toolbar has many useful accessibility testing features in it that we often leverage. The two issues that I check with it are headings and alternative text value.

<https://chrispederick.com/work/web-developer/>

*Demo*

**Manual Testing**

Automated testing is great, but it often only accounts for about 30% of all known accessibility issues a website may face. Certain accessibility issues can only be tested manually. Fortunately, its not that hard to do. First tool up is...

**Colour Contrast Analyser**

If you would like to test for contrast issue, then the Colour Contrast Tool is very good for this job. It is particularly useful for testing comps before they go to development.

<https://developer.paciellogroup.com/resources/contrastanalyser/>

*Demo (Open PDF, Open CCA)*

**Photoshop**

Photoshop can also be useful in testing comps. You can grayscale the work to see how it looks. If certain areas of the design appear to be illegible, then this is usually a good indicator that there will be contrast issue.

Photoshop also has a little known technique to proof comps for common forms of color blindness, such as Deuteranopia and Protonopia.

**Google DevTools: Accessibility Tab**

Jumping back to Google Devtools for a moment, the “Accessibility” tab (often obscured by window width), can provide some detailed information about the nodes in the DOM.

*Demo (*[*https://www.tmp.com/search-jobs*](https://www.tmp.com/search-jobs)*)*

**Screen Readers**

The accessibility team uses various screen readers to test your work. It is very important to listen to your work. If you want to empathize with somebody who is blind or who has low-vision then you have to use the tools they use to navigate your site.

On the PC, we use JAWS and NVDA.

NVDA is free to install (<https://www.nvaccess.org/download/>)

On the Mac we use VoiceOver, which is built into the OS (Click on Command + F5 to toggle it on and off)

You should listen to your work on multiple browsers, but if time is short, then only run your screen reader over Firefox, as this is the preferred browser of many disabled users.

Pay particular attention to interactive elements that may not be labeled properly, have incorrect roles or states. For example, visual buttons with missing text, linked images with missing alternative text, a link with a role value of “button” (e.g., <a href=”...” role=”button”> … </a>), etc. What you convey visually, you must convey auditorily.

Use as an opportunity to see if your interface would benefit from additional enhancements that can be found in ARIA. For example, adding aria-expanded to a button to convey state,etc.

Be careful not to go overboard with ARIA, as misuse can make your interface less accessible. The first rule of ARIA is not to use it at all, meaning that one should always leverage more inherent semantics found in HTML.

<button> … </button>

<div role=”button” tabindex=”0”>...</div>

---

**Keyboard Testing**

The best tool to use for keyboard testing is...wait for it...your keyboard. I know, a real shocker, right? Tabbing through your interface is not hard. Oftentimes, you can even combine screen reader and keyboard testing efforts, by tabbing through and listening to your interface at the same time.

**Visual Testing**

Visual testing is about knowing what to look for. Pay attention to details. Is contrast okay? Not sure? Whip out the Color Contrast Analyser mentioned above. Do all videos have captions? Are the captions accurate? Would the experience benefit from a transcript? Are we spawning new windows without warning? (You all know how I feel about that. See <https://spellacy.net/the-last-word-on-opening-new-windows/>) Does third-party content or tools on our site Bring up these issues with your DPM or manager, etc. The more we advocate, the more accessibility becomes a priority and a part of our culture. That’s what we all want and its really the only way we can succeed at becoming a more inclusive company.

These are the bare minimum of tools needed to address the most common accessibility issues a site may face. As you become more proficient with using these tools, they will eventually become second nature to you. Fixing issues is another problem altogether though. I understand that development is a spectrum of talents. Some developers will be more adept at fixing accessibility issues, while others may struggle, and that is okay. Nobody expects you to be an expert. As you uncover barriers to access, do the best that you can to address any issues that you uncover and if you hit a real stumper, then feel free to reach to developers with more proficiency. We are all here to help and better one another.