The Internet is for Everyone

Slides available at kattow.dev/the-internet-is-for-everyone

This is not an original title.

It originally belonged to Vint Cert's The Internet is for Everyone

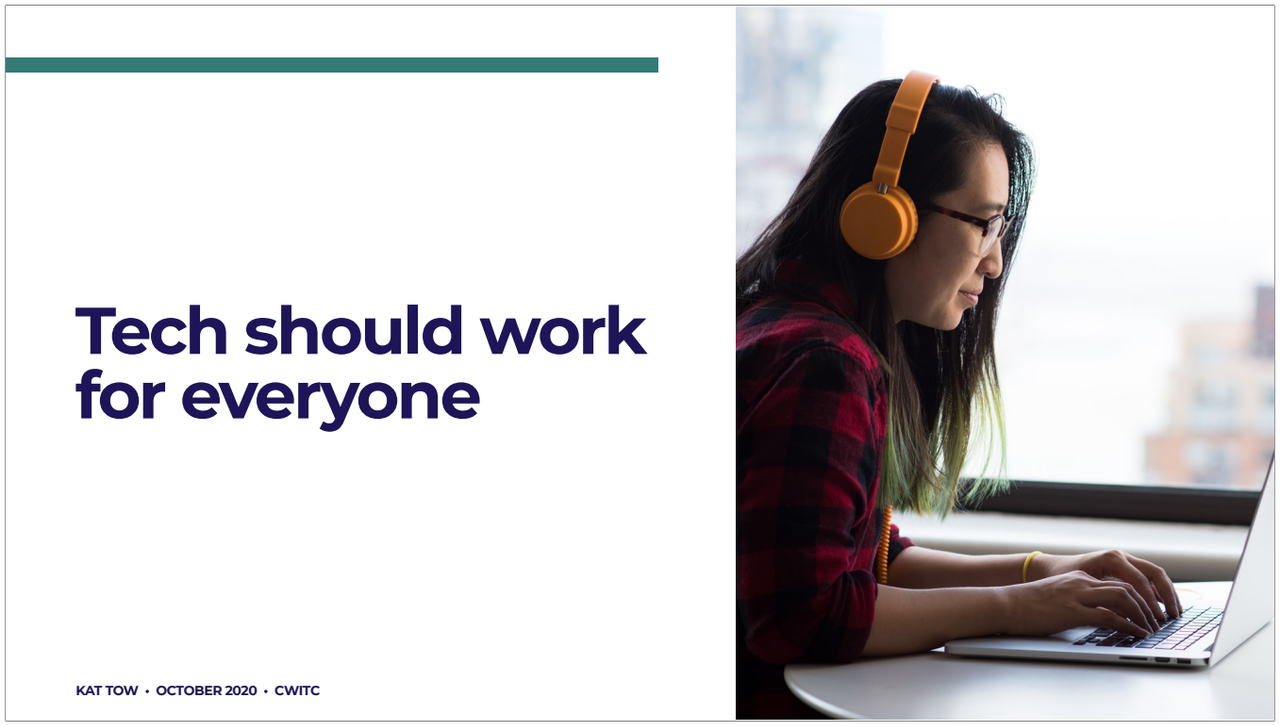
"The Internet is for Everyone” is not an original title. It is the title of a memo written by Vint Cerf at the turn of the century. Written before smartphones, before streaming video and music, before apps. But it is startlingly prescient.

Vint could not predict everything that would happen in the new century, but he knew that the growth of the internet was unstoppable. He also knew that we must be active stewards of the internet. His memo implores us, as technologists and as citizens, to be responsible in its use and development, and to work for digital equity.

We've come a long way

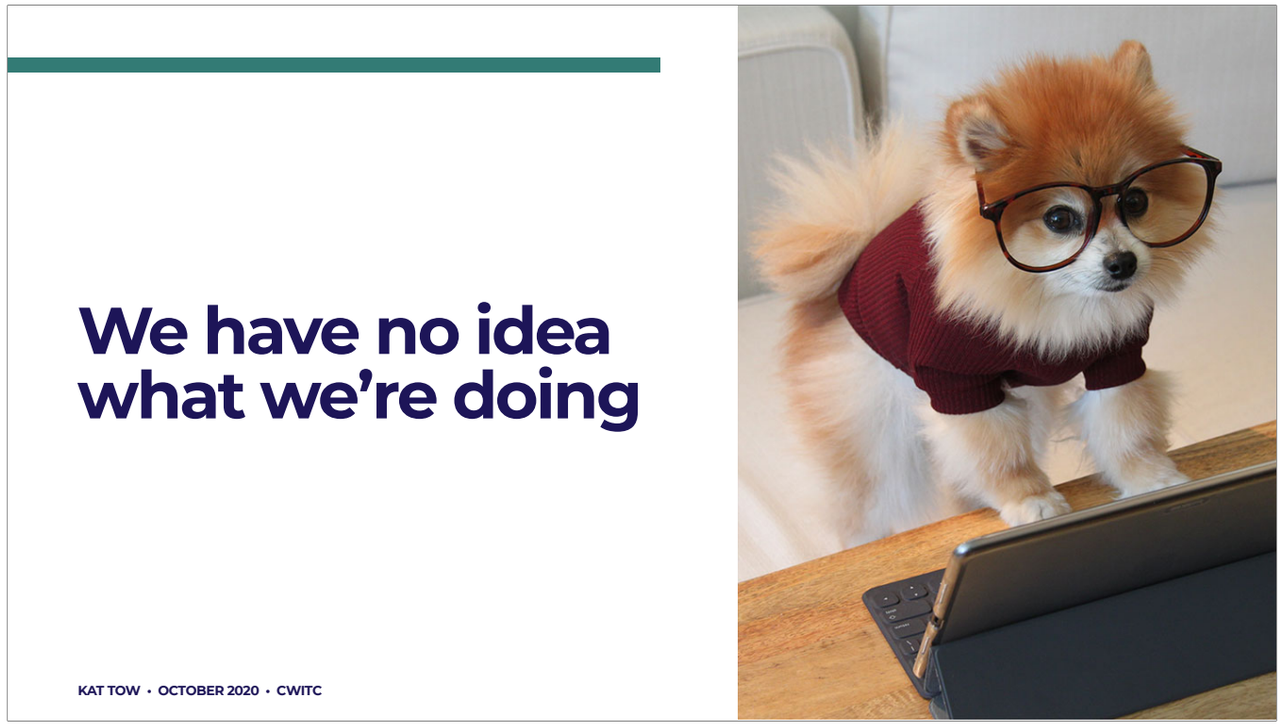
KAT TOW • OCTOBER 2020 • CWITC

The world is radically different today than it was during the Y2K scare. A lot has changed. Technology has changed, a lot. Tech can do so much more now, and for most of us here, technology is an indispensable part of our daily lives. We rely on technology for essential services, for entertainment, and to share our lives with loved ones.



But not all of those technologies that we rely on work for everyone. Some technologies require fine motor skills. Some don't provide meaningful content for screen readers. Some include visual effects that make some people sick. And that makes it difficult, or even impossible, for some people to use some technologies.

It doesn't have to be that way. We know that we *can* create technology that is inclusive and accessible, that works for everyone. But we don't always do that. Why? Most of the time, it's because we don't know what we're doing.



I don't mean that we're flailing around, slinging code in the dark and hoping it compiles (although we've all had those days). I mean that we don't always know *what we don't know* - it's impossible to create an inclusive and accessible product if you don't know how people with disabilities use technology, or if you don't realize that some disabilities and barriers even exist.

If we truly believe that the Internet is for everyone, and if we want the tools and products that we make to fulfill that promise, we need to understand what "everyone" means. "Everyone" doesn't look only like the people in this Zoom call.

So what do we need to know? This talk will first outline what we mean by disability, and who might face problems using technology. We'll also cover why this matters - not just in the interest of equity, but to your business, to your shareholders. We'll then start to talk about how you can start identifying and improving accessibility in your work.

The speaker

Is a web developer

Is not an expert

@akattow

/in/kattow/

hello@kattow.dev

KAT TOW • OCTOBER 2020 • CWITC

A couple of housekeeping notes before we get started:

I live and work in Milwaukee as a software developer, although really I work on web apps. I am mostly in the frontend, I do some backend. But I pretty strictly do web. So the rest of this talk will focus a bit more on web accessibility, because that's what I am familiar with. But the same general principals apply across technology, whether it's a website, a mobile app, or a desktop app.

The language

Person-first or identity-first?

a11y = accessibility

There are two primary ways in English that we can talk about disability and people - person-first and identity-first language. For example, you might say "a person with a disability" (person-first) or "a disabled person" (identity-first). The correct linguistic choice is always **the choice an individual or community prefers**.

Different people and communities have complex reasons they prefer one over the other, and that discussion is out of scope. This talk isn't about a specific type of disability or community, so there is no 'right answer'. I'll try to use both person-first and identity-first language, I may favor one or another without realizing it.

a11y is a numeronym for accessibility. It is most commonly used to refer specifically to web accessibility. You will see a11y in links I include on my slides and in a lot of online resources.

What is disability?

What comes to your mind when you think of disability? A common image is a person with some kind of physical impairment, such as a wheelchair user. Or maybe you're reminded of so-called "invisible" disabilities, like deafness or learning impairments.

The medical model

KAT TOW • OCTOBER 2020 • CWITC

Most people, when they think of disability, think of something like this - people in wheelchairs, folks who are blind or deaf. These are attributes or characteristics that people have; needing a wheelchair is a *feature* of a *person*. This is known as the "medical model" of disability. This mental model is obviously very common, and it has some important uses - medicine is the obvious one. However, it isn't particularly helpful for designing and developing digital products. When we think of these attributes as rare edge cases that apply to only a few potential users, it's easy to develop an attitude that accessibility is an "edge case" that isn't a priority for the product, for design, for the development team.

The social model

KAT TOW • OCTOBER 2020 • CWITC

So to develop a *new* mental model, I'm going to ask everyone to participate in an exercise in imagination. So, imagine a world in which almost everyone you know is blind. Your coworkers are blind. Your neighbors are all blind. Most people are blind. And in this world, since most people are blind, almost all written word is in braille. Library books are in braille. Restaurant menus are in braille. Government pamphlets about essential services are in braille. And you don't read braille.

In this world, being blind is not a disadvantage, it's the status quo. Both this imagined world *and* the world around us have been designed *by and for* people who fit the status quo. You, as a non-braille reader, are at a huge disadvantage in this alternate reality. But that disadvantage isn't due to a medical condition - it is a **symptom of design**. The world around you didn't consider the needs of non-braille readers, and so your ability to thrive and even survive is significantly impaired.

This is known as the "social model" of disability; users have unique abilities and needs, and society (or in our case, technology) can support or ignore them.

Poor design disables users.

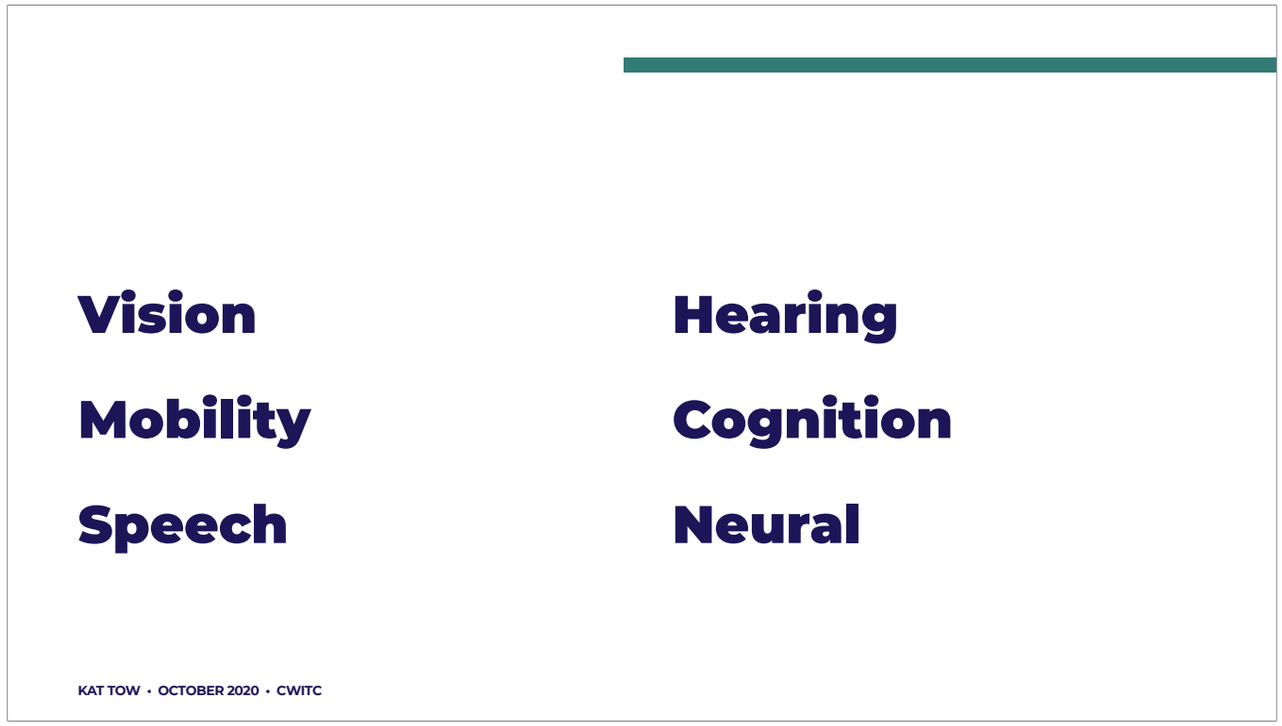
Good design enables all users.

Poor design is what *disables* users, and good design *enables all* users.

Neither the "medical model" or the "social model" are perfect - but if we want to design really great products and technology, the social model is going to be far more useful.

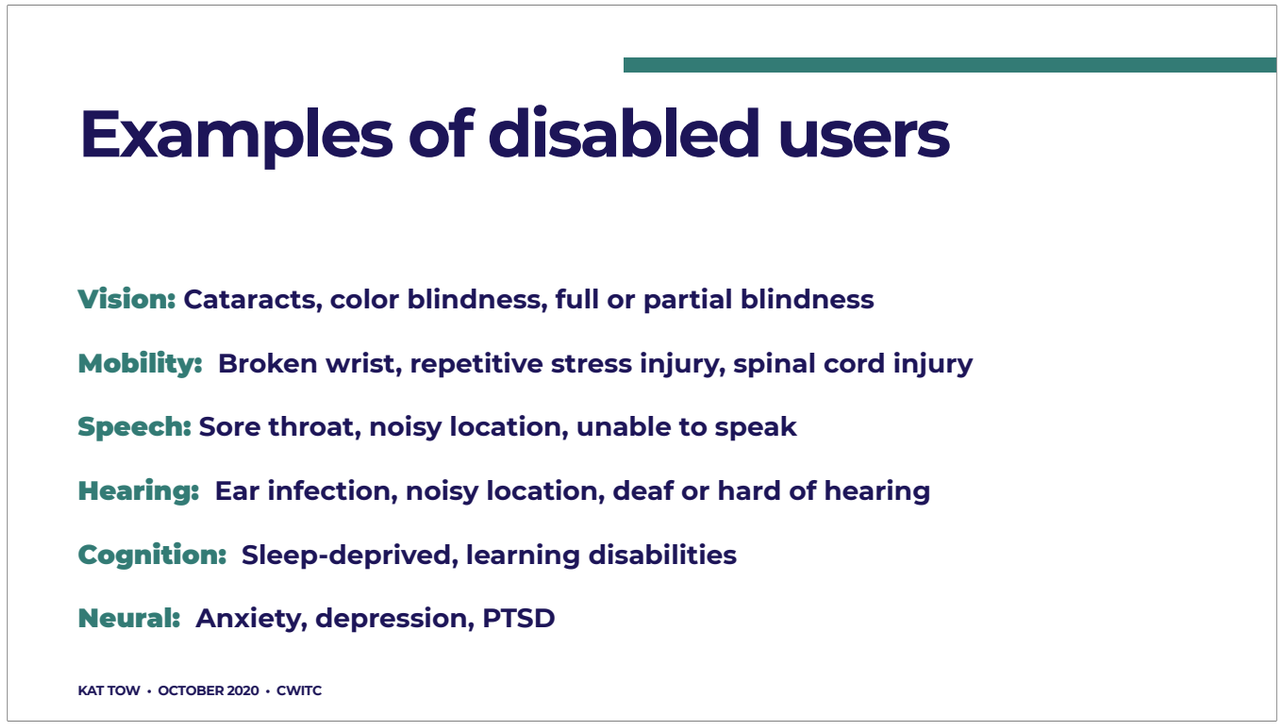
Who is disabled?

So now we have a better mental model to think about disability in our products and technology. But, we still need to understand who is disabled by poor design.



There are a few broad categories that we can use to think about disability - both in terms of what our design is enabling or disabling, and to help us think of different impairments our users may have.

And I want to point out here that even when we call certain qualities, such as blindness, impairments or disabilities, we should remember that in our social model, is a linguistic shorthand for "a quality that some systems are not designed to accommodate." And that includes temporary impairments as well.



Here we have just a few examples of disabled users.

- Vision: Cataracts, color blindness, full or partial blindness
- Mobility: Broken wrist, repetitive stress injury, spinal cord injury
- Speech: Sore throat, noisy location, unable to speak
- Hearing: Ear infection, noisy location, deaf or hard of hearing
- Cognition: Sleep-deprived, learning disabilities
- Neural: Anxiety, depression, PTSD

For any given website, app, or other technology, some of these conditions make it more difficult to use that tech. In more severe cases, poorly-designed technology can't be used at all by some users.

Time for another imagination exercise - imagine breaking both your wrists or hands - maybe in a traffic accident. How would you continue to use your computer or mobile phone while in casts? The good news is that these days, most hardware - phones, laptops, etc - have voice control built-in. Siri, Cortana, etc. The bad news is that many of your favorite or essential websites might not work well with that technology.

Anyone can become disabled, even temporarily. And although the cast example is pretty easy to see, many disabilities are not visible. Whether or not you realize it, you almost certainly know people who deal with impairments every day. We as technologists are responsible for making sure that our products don't disable users, whatever life throws at them.

Why is this important?

So why is this important?

Hopefully you agree that it’s important because supporting users with both permanent and temporary disabilities is the right thing to do. But what’s the scope? How many people are disabled?

85.3 million

2014 survey from the Social Security Administration (SSA)

According to a 2014 survey from the Social Security Administration (SSA), 85.3 million Americans are living with disabilities. That is over 1 in 4 Americans. This does not include people with temporary impairments, like broken wrists. It does includes people with impairments and restrictions that are not classified as 'severe'. About 55.2 million Americans, or 17.6% of the population, had a severe disability at the time of this survey.

27% vs. 2%

Are you still supporting Internet Explorer?

Has your development team spent years supporting Internet Explorer? IE has only 2% global usage. Why spend your development hours prioritizing those 2% of global users over the 17.6% of Americans with severe disabilities?

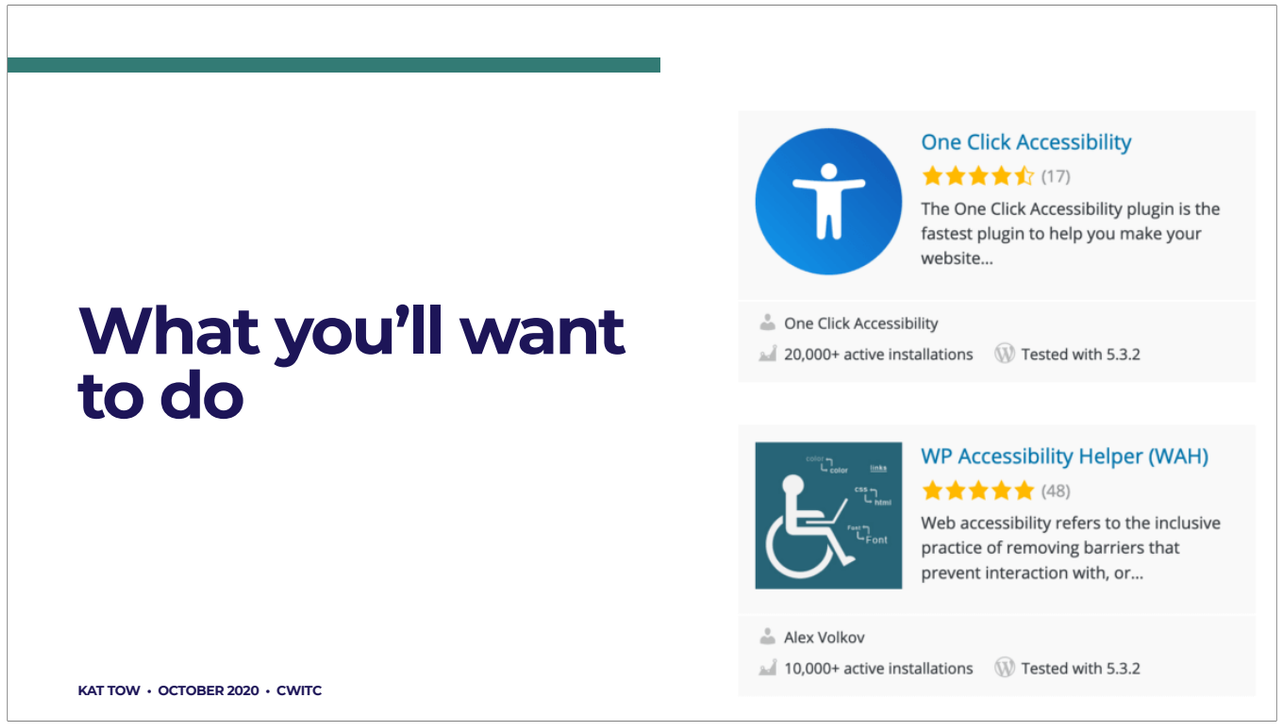
We know that not every disability will impact web and technology use, but a significant portion will. 1 in 4 Americans have a disability. A significant portion of 1 in 4 Americans is a lot of potential and current users. And when we don't consider accessibility in our products, we may be making it difficult or even impossible for a lot of users to access our products.

In the past few years, the United States has seen a massive uptick in web accessibility lawsuits

The National Law Review - When Good Sites Go Bad

Finally, if product reach isn't enough of a motivator, consider this: the United States has seen a massive uptick in web accessibility lawsuits. For quite a long time, there wasn't a lot of guidance in the law about whether the Americans with Disabilities Act (ADA) required accommodations on websites and other digital platforms - and if so, to what extent? But since some initial rulings in 2017, and spurred by the case of Robles v. Dominos Pizza in 2019, a growing number of companies are facing lawsuits and liability risks from inaccessible websites. Is a lawsuit something your company can afford?

Hopefully I've convinced you that accessibility is really important. It can help a large number of your current and potential users. Many accessibility improvements will make your product better for all users. Ignoring accessibility puts you at risk for lawsuits.



You may be so convinced that you want to make your product accessible right now. You might go home and start researching. And you're going to find some plugins.

Overlays, helpers, tools that claim that if you add them to your website, they'll handle making it accessible for you. That sounds amazing!

Unfortunately, they aren't all they're cracked up to be. I've included a couple of articles in the resources at the end of the slides, but accessibility plugins are, at best, a band-aid over the problem, not a real solution. At worst, they can actually cause more confusion for users. There are some things that a plugin simply can't solve, like providing meaningful alt text for images and descriptive text for links.

The second half of this talk is going to cover what you can do to bake accessibility in. I'll cover what the most common problems are and how you can audit for them. And if you audit your website and find a ton of problems, you may still decide to use a plugin in the short-term, while you work on remediation.

But I want you to get excited about finding and fixing accessibility yourself, and about building accessible features the first time. Building good testing and development habits is a much more sustainable choice that is going to make you a better developer and make your product better and more stable. And like anything else in tech, it's something you can work on incrementally. So let's get started.

How to get started

Web accessibility is not new. The Web Content Accessibility Guidelines, or W-C-A-G, or wcag were first established in 1999, over 20 years ago. Despite this, accessibility has a reputation for being extraordinarily difficult to implement.

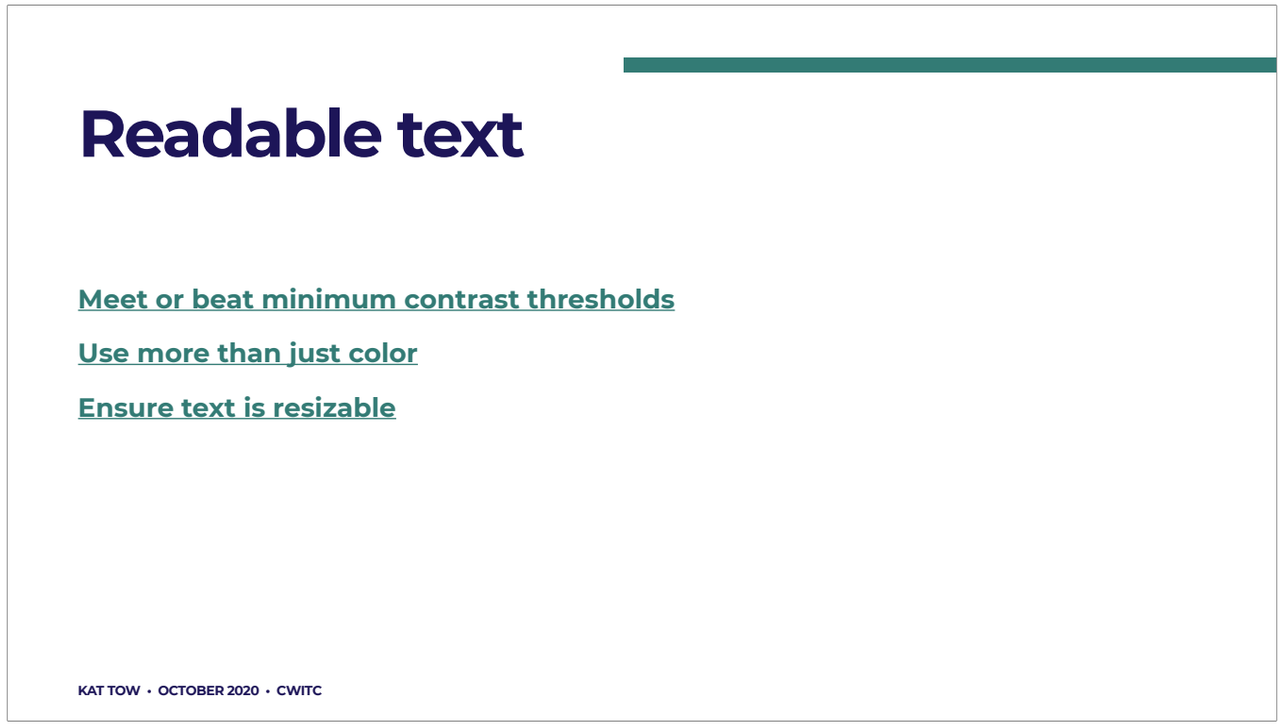
In fact, building a website or an app is actually extraordinarily difficult. Most people can't do it. But it's something you can learn. You need to learn not only the language, but also the boundaries and expectations. The same thing is true of accessibility.

And just like learning to code, or design, or manage products, you need to understand that you won't go from zero to hero. It takes time to learn, and it takes time to build. But every incremental step you take, every sprint or cycle, is adding value to your product.

Critical Paths
&
Common Problems

So what's the first step? Like any project, you'll want to identify your product's most critical paths. What do users absolutely need to be able to accomplish? For example, if you're building a restaurant's website, your most critical path is for users to order food online.

Targeting the most important user stories will help you create clear goals, make manageable pieces of work, and demonstrate measurable improvements.



For developers and designers new to accessibility, it can be hard to even know what accessibility issues to look for. And oftentimes, the easiest issues to test for are also the easiest to fix. While this list isn't exhaustive, there are a few of the most common problems that every team should be on the lookout for.

Readable text.

Make sure all your website's text meets minimum contrast thresholds. Low-contrast text can be difficult for anyone with poor vision to read. Additionally, avoid using only color to identify information or tools (e.g., links), and be sure text is resizable.

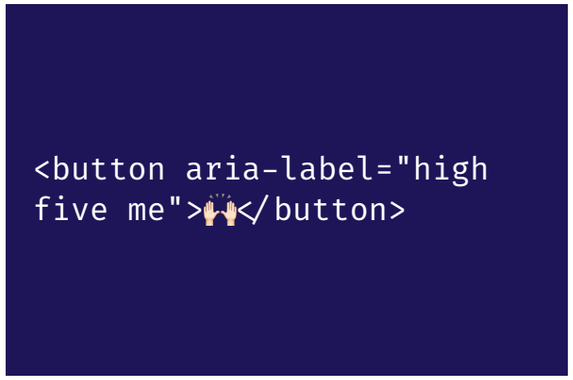
Text alternatives

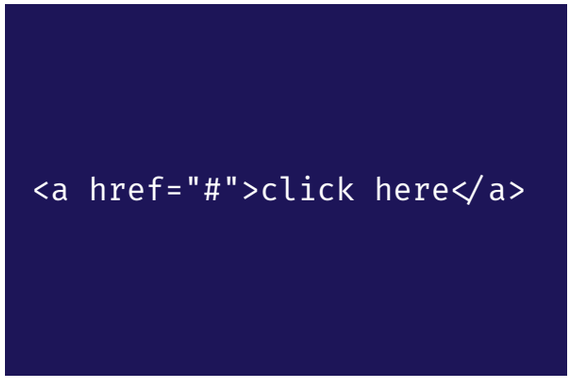
```
alt="a golden retriever uses a computer"

alt=""

cute_dog_102919_prod_final2.jpg
```


Text alternatives.

Anything essential that isn't text needs a text alternative. Every image that conveys important meaning to the user needs alt text. For example, images of products for sale should have descriptive alt text. And images that don't convey meaning should have empty alt tags. Why? Because without an empty alt tag (alt="") screen readers will announce the entire filename (cute_dogs_102919_prod_final2.jpg). Video and audio need text alternatives as well.

Links, buttons, & native HTML



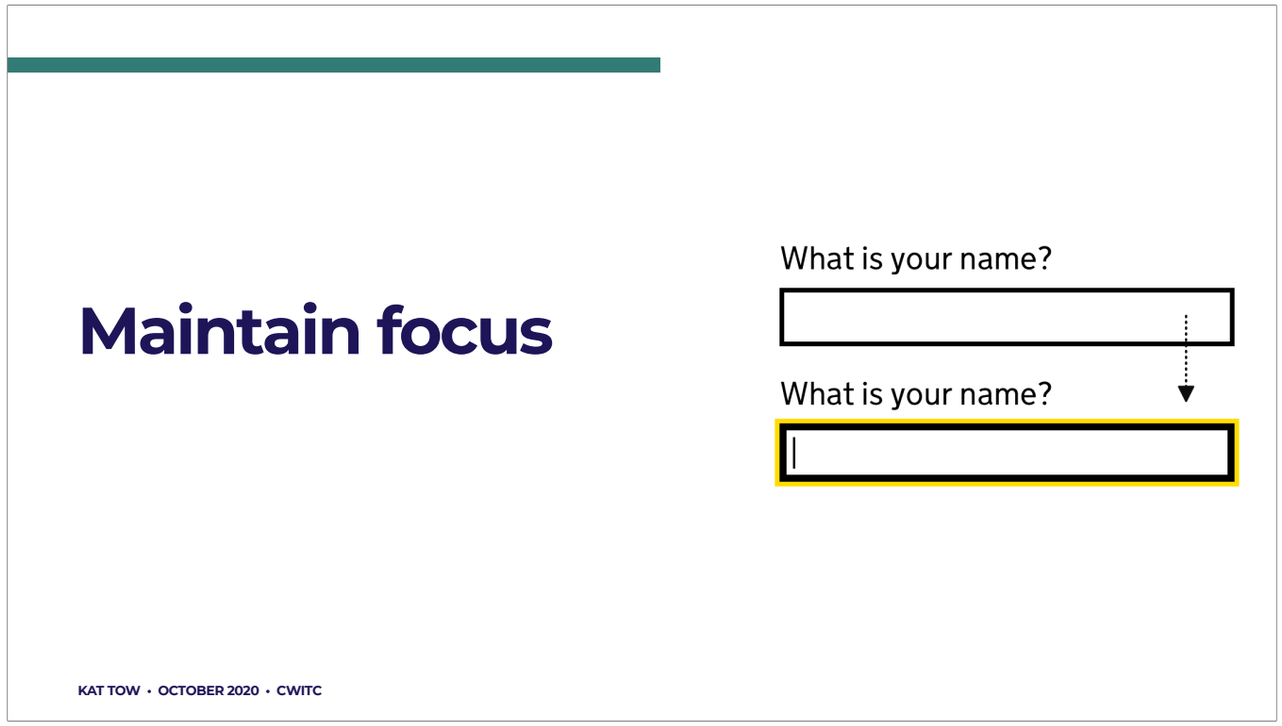


Descriptive and accessible links and buttons.

Make sure that all interactive elements contain either text or alt text that informs users what that link or button will do. Additionally, all links, buttons, and other interactive elements must be reachable using a keyboard (more on this in a moment).

Label your forms.

Forms are typically part of a critical user path. Whether it's a form to sign up or order a pizza, forms are critical to most sites. Recent design trends have made forms very difficult for some users by removing label elements in favor of things like placeholder attributes. While this can be annoying and difficult for any forgetful user (for instance, users with memory problems,) screen readers won't announce placeholder attributes at all, leaving users who rely on them totally confused. This one is pretty specific, but it's become a common problem. Every form element needs a corresponding label.



Maintain focus.

Less of a "do" and more of a "don't" - don't disable focus styles! It's fine to overwrite the default focus styles to something consistent across browsers, and maybe even something more inline with your brand and design. But focus styles help users with mobility issues, vision issues, and cognition issues keep track of where they are - don't disable them.

Be chill.

Check your website for flashing or fast animations and movement (I'm looking at you, auto-playing video banners). This kind of content can cause reactions as common as motion sickness and as severe as seizures. If the content must stay, be sure you warn users and make it easy to stop playing (again, looking at you video banners).

Detecting issues

I just covered a lot of things. Some or maybe even all of them may have been new to you. It can seem overwhelming, especially when I tell you that those are just some of the most common problems - there are plenty more.

Even more overwhelming can be the thought of trying to audit your product. If you have just one website with only a few pages, maybe that's not so bad. But for folks with massive websites, or multiple sites, or agencies who work on new sites every year, the prospect of manually checking the code for every part of a website can be totally unfeasible.

Luckily, there are several tools to help you streamline and even automate parts of accessibility auditing, which I am about to cover. But before I do, I also need to say that none of these tools are sufficient on their own or together. They will make your audits easier and faster, but some manual testing will always be necessary to ensure your site is accessible.

Linting

pa11y

eslint-plugin-jsx-a11y

Linting is a great way to prevent accessibility bugs before they hit production. If you're writing React, you should check out eslint-plugin-jsx-a11y to statically check your JSX elements. Regardless of your stack, you can also use pa11y in the command line to audit an entire site.

Automated tests

axe

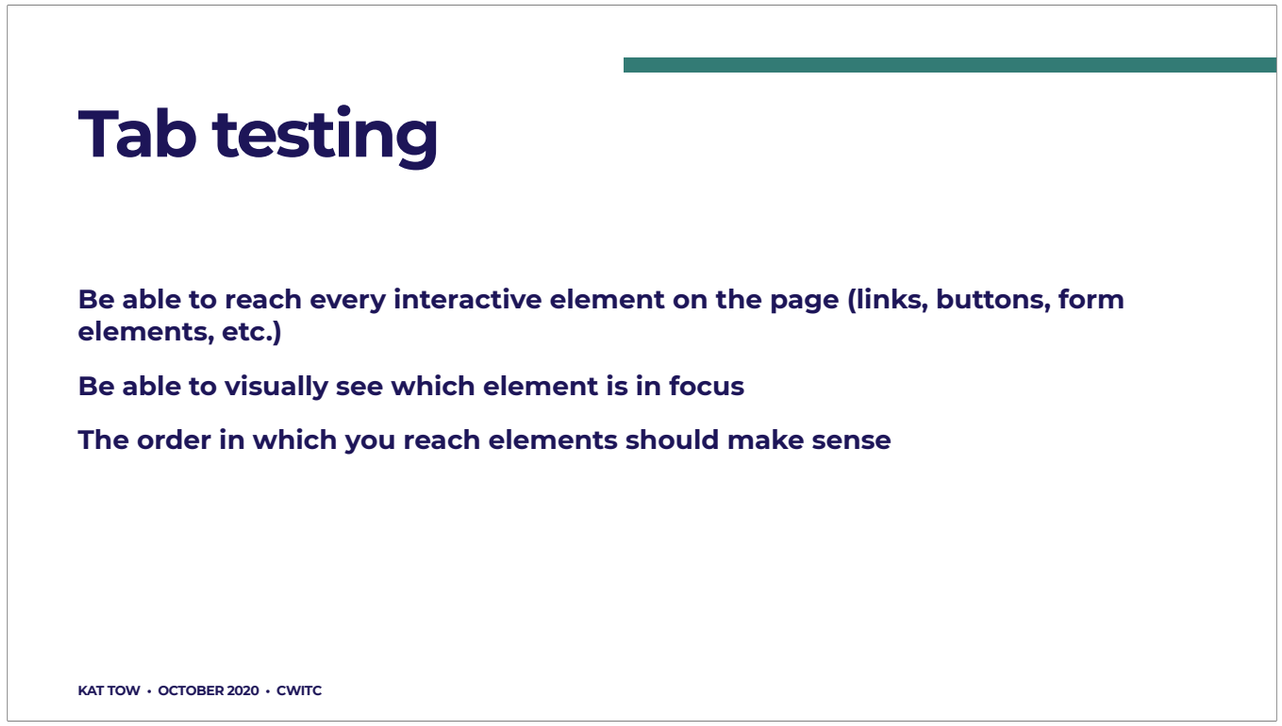
WAVE

unit, integration, e2e

In-browser testing can automate a lot of checks for you. These tools will find common errors like insufficient color contrast, missing alt text, duplicate landmarks, and more. Two of my favorites are axe, which is available as a Chrome extension and for Android testing, and WAVE, which has a web app, Chrome and Firefox extensions, and API services.

If your team is already incorporating browser-based audits and you're looking to level up, you can also think about adding accessibility checks into your unit, integration, and end-to-end tests. Most popular test runners have accessibility packages or tools you can use to incorporate some automated checks there as well.

And that's it for automated tests. There is certainly work being done right now on things like AI suggestions for alt text, but it's unlikely that we'll ever get to a point where we can fully automated every accessibility test we need. There are some things that we still need to do ourselves.



Tab testing is manual but fairly low-effort. In fact, once you start doing it, you may find you prefer tabbing for your normal web browsing! Tab testing just involves opening up your site in a browser and hitting the tab key. You should:

- Be able to reach every interactive element on the page (links, buttons, form elements, etc.)
- Be able to visually see which element is in focus
- The order in which you reach elements should make sense

Take note of any difficulties you experience so that you can investigate and remediate those problems later. Missing focus styles are the most common, so pay close attention for those.

Content analysis

~~Click here~~

Hemmingway

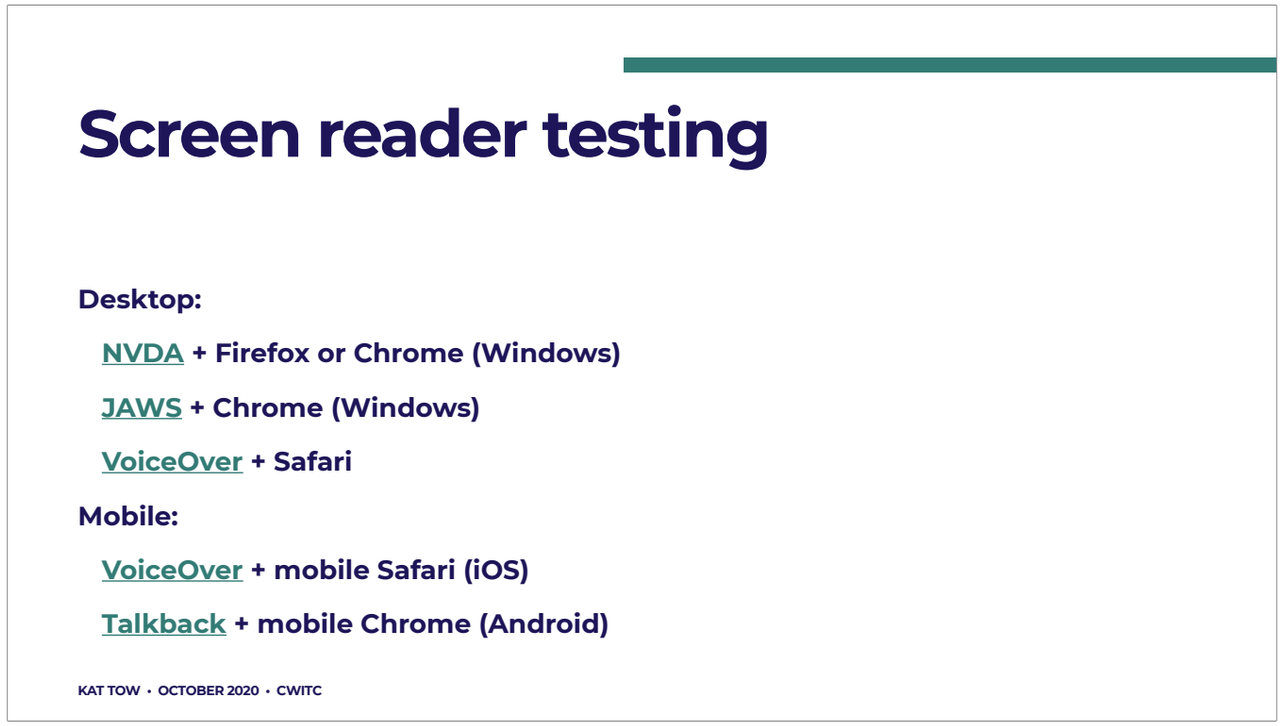
Grammarly

Content analysis can often be overlooked in accessibility recommendations. Especially for developers, it can be really easy to just focus on testing and remediating issues that we can fix in code. But ensuring content accessibility makes a big difference in supporting users' cognitive and neural diversity.

Most of this process has to be manual. Things like ensuring that link text is descriptive - no more 'Click here' links.

Some of it can be assisted - writing editors like Hemmingway (free!) and Grammarly (freemium) can help your team ensure that your content itself is more accessible and understandable.

Tab testing and content analysis are both manual, but can be fairly low-effort. They may take time, but they don't require a lot of additional expertise.



Screen reader testing can be very intimidating, especially for folks just getting started. Unlike every other auditing method I've covered, this will require you to learn a new skill. Navigating via screen reader can be hard for new users to pick up, but understanding how screen reader users experience your product is invaluable.

If you're brand new to accessibility testing, I recommend just choosing one screen reader (SR) software that works on your existing operating system. Get comfortable with one tool and with the process of using a screen reader. If you have a team that works on both Windows and Mac, that's great - maybe each team member can learn one screen reader, and you can diversify your testing that way. Just remember that testing thoroughly in one program is better than testing poorly in 3.

The most popular screen reader combinations are:

- NVDA + Firefox or Chrome (Windows)
- JAWS + Chrome (Windows)
- VoiceOver + Safari

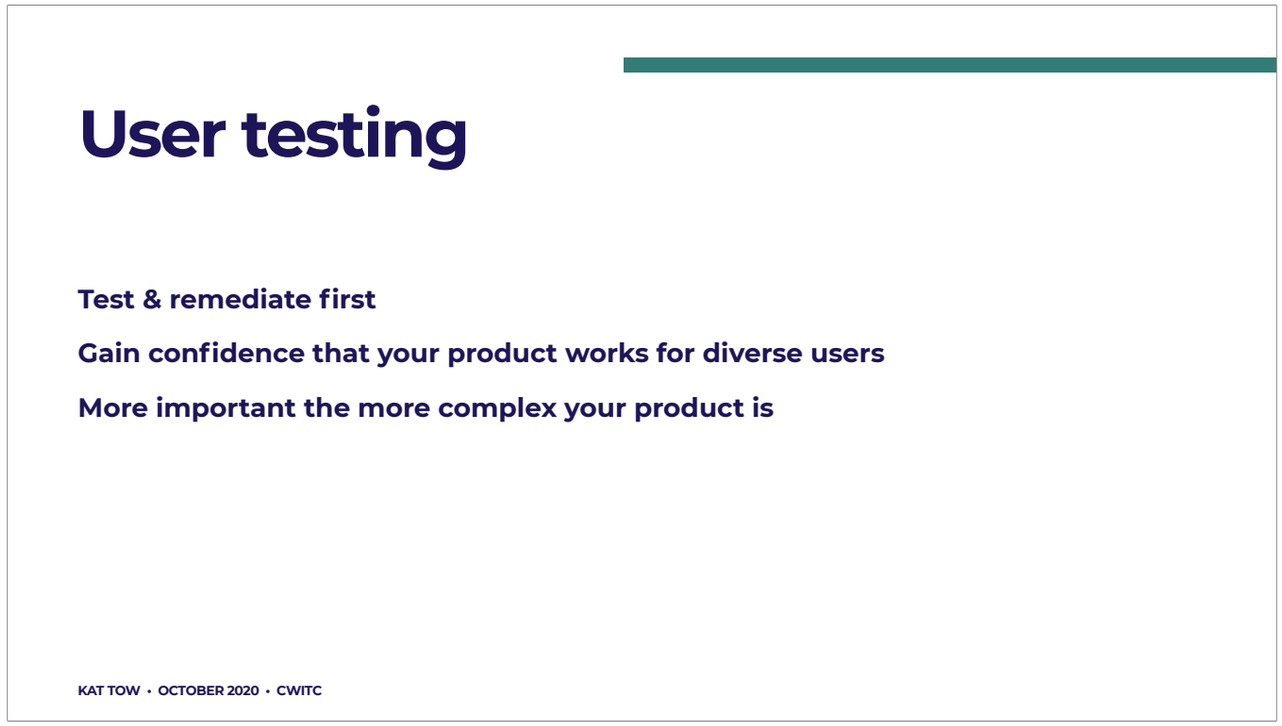
If you are testing mobile, you'll use want to use Voiceover on iOS with mobile Safari, or Talkback on Android with mobile Chrome. If you're doing cross-platform mobile apps, that is one situation where you really may want to get familiar with both screen readers.

All of these screen readers are free except JAWS, which does require an annual license.

As you prepare to start testing, the WebAIM Screen Reader Survey #8 (2019) can help you understand how people use screen readers. And when you do get started, you can use these keyboard shortcut cheatsheets to navigate using your keyboard. And as a reminder, if you want to follow those links later, these slides are available at the

web address in the footer.

The process of screen reader testing may uncover behaviors that confuse you. Some behavior may not be what you were expecting, but you may not know if it is what users who regularly rely on screen readers are expecting. The best way to find out is by conducting user testing.



First of all, user testing isn't free. Please don't waste your testers' time - test and remediate smaller issues first using all of the methods we've already covered.

But yeah, user testing. User testing helps you ensure your project actually works. It's more reliable than any manual or automated tests you can perform. And the larger and more complex your product is, the more important it becomes.

And whether you aren't doing any user testing yet at all or have a robust user research practice, it's important to include users with disabilities. People who rely on assistive technology every day may catch things you never even thought of.

If you need help recruiting users who rely on assistive technology, there are a few companies that specialize in recruiting users for accessibility user testing. Or maybe you already work with a user testing firm that can help you recruit assistive tech users. You'll need to do your own research to find the right user testing solution for your situation.

Where to go next

Like any subject in technology, there's a lot of deep learning and expertise that you can pursue in web accessibility. Personally, I am still learning a ton. I am far from an expert. I'm a developer who thinks this stuff is important, but I don't get to spend all my time auditing and remediating issues. I spend most of my time like many of you do. But I'm conscious of accessibility when I write new features. I've been able to integrate linting, in-browser audits, and two screen readers into my regular toolkit.

As you continue to learn about web accessibility, it's important to remember that this isn't the wild west. The Web Content Accessibility Guidelines (WCAG) were established over 20 years ago and was updated as recently as 2018. A draft for the latest updates was released in August 2020. The documentation can be intimidating to read, but it's important to remember that there are guidelines, expected behavior, and testable practices. This resource and others are available to help you, and I have a few slides at the very end with some of my favorites.

You can do it.

There's a lot of content in the WCAG, because... tech is complicated. That's what makes it so powerful. Whatever your product is, whether it's an app or a program or a website, I bet it's powerful. It's doing something amazing to make people's lives better or easier, something we could only dream of 20, 30 years ago. And you've worked hard to make it the best it can be, and to reach as many people as possible.

So don't let old code and passing design trends disable your users. Make sure that your product is enabling your users, all of your users. Make sure your slice of the Internet really is built for everyone.

General resources

WCAG

Web Accessibility Initiative at W3C

WebAIM

Current State of International Accessibility Standards

Tools and Techniques

The A11y Project

Accessibility for Teams

Design resources

Microsoft Inclusive Design Toolkit

Inclusive Design Principles

Designing For Accessibility And Inclusion

Browser testing

WAVE

axe

Lighthouse

Screen readers

NVDA + Firefox or Chrome (Windows)

JAWS + Chrome (Windows)

VoiceOver + Safari

VoiceOver + mobile Safari (iOS)

Talkback + mobile Chrome (Android)

WebAIM Screen Reader Survey #8 (2019)

SR keyboard shortcut cheatsheets

Courses

Web Accessibility by Google - free

W3 Web Accessibility - free for a few weeks

Deque University - paid

On plugins

[Honor the ADA: Avoid Web Accessibility Quick-Fix Overlays]

[Web Accessibility Overlays Don't Work]

[Toolbar Plugins/Widgets for Website Accessibility Aren’t ADA/508 Compliant]

[Overlays are not the solution to your accessibility problem]

Thank you!