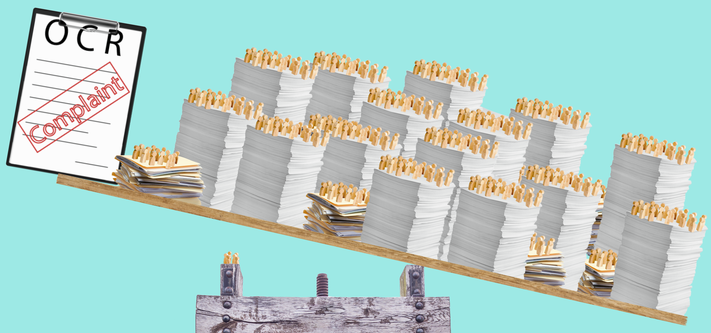
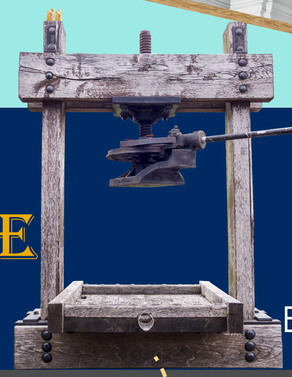
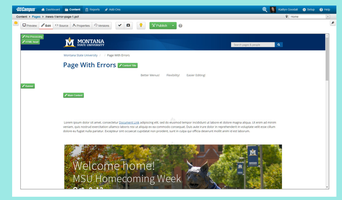
Hacking LEMONS Into Accessible LEMONADE

Squeezed By Your Content Editing Environment

Kaitlyn Goodall, Montana State University Web & Digital Communications







If your CMS renders **your** HTML to show editors what they're working on...

... you can do...

Anything you can dream of!

Success is a journey, telling a story

September 27, 2017
Office of Civil Rights Complaint Letter

December 2017
Prelim exploration of scanning code and server side scanner begins

January 2018
Alpha server side scanner

March 2018
Prelim CMS accessibility UI sketching

April 2018
Developing CMS accessibility task tool

May 2, 2018
voluntary resolution agreement

June 2018
CMS a11y task tool alpha

August 2018
Beta a11y task tool release to production with group access controls limited to testers

October 2018
RC1 of CMS A11y task tool, video for training recorded

November 2018
New Accessibility training released to CMS content editors, related task tool activation on training completion per person

January 9, 2019
141,797 errors

March 2019
Nearly 100% active editors trained and now using the Accessibility Task Tool

May 2019
Editing privileges for people who have not taken training are turned off.

May 30, 2019
OCR Deadline for training and all new content to comply

July 10, 2019
61,517 errors

September 26, 2019
57,710 errors

80% of remaining errors are "document" errors: desktop documents which have not yet been verified for accessibility or removed/ replaced with web content if inaccessible.

May 30, 2021
OCR Full compliance report due

Most of our content editors are just that: content editors, with related word processing skills and usually no html/etc knowledge.

Most of them do web content work as a small fraction of their wider job, whether that might be as an administrative role, a faculty member, or anything else.

WCAG 2.1 is technically complex, requiring a basis in related technical fields and jargon. Related training can easily run 8 hours for an experienced html developer or designer (WebAIM, etc).

How many hours does it take to train someone without any related background to a point where they have adequate HTML/CSS expertise?

At best, 8 hours x 800 people = **6,400 hours spent**, across the University. How many hours will be spent by staff running training? Will outside trainers be instead paid to come in? How much will that cost? How many hours of related support will be needed? How many new editors are trained each year (answer: roughly 200).

Trying to train them to expert levels is neither fair to them, nor sustainable for the University

At Montana State University, we looked seemingly impossible odds in the face and found in them the basis for crafting our own engine of success, by focusing on charting a course to an outcome where every person's strengths could be leveraged to empower everyone else, rather than weighing each other down.

You can too.

"the team"
6
W&D developers, support staff, and departmental administrators

vs

1 closed source CMS (and 1 OCR letter)
>1,600 registered CMS editors (~800 active)
>150,000 total accessibility issues (more daily!)
>500,000 pieces of web content (more daily!)

Crushed by numbers

(at 1 error per 5 minutes, that would take six people an entire year of doing nothing else.)

...and desktop documents are **much** slower...

Reframe your view

Turn 6 into **806**

Empower every person to be your team

806 **people**, solving 1 error every 5 minutes, could **theoretically** be done in **2 work days**, and even... never publish another error... !???!!

We're all about education!

But training content editors into HTML/CSS accessibility experts is not realistic or sustainable

remember! we're here to help **PEOPLE**: don't change your framing just to lose sight of THEIR perspectives and lived experience!

Ingredients For Success: Accessible & Universal Design concepts...

Accept people as they are:
find solutions for **their** current abilities and needs; don't force them to fit **your preconceptions** of someone who fixes accessibility errors.

1. Reduce the ability to make errors
Constraint is a powerful but easily unpopular usability design tool. Try to offer new and improved but more tightly constrained atomic design options within your editing environment, to give something positive while taking away other options.

2. Don't make perfect the enemy of good
Accept that you will need to make compromises, but try to do so with your focus on who they will affect the most, what they entail, and how that looks at a systematic level in terms of cost.

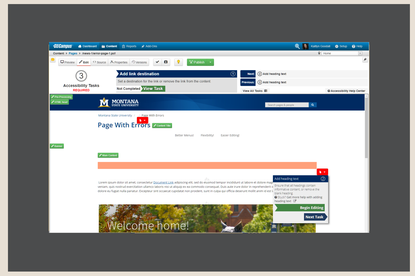
3. Use your mandate in positive ways
If you need explicit policy, budget, or anything else, use your OCR letter and any related agreement by stressing what you **will do** with those tools as why they are necessary to the University's mission.

4. Explore outside options from the start
Don't be afraid to spend money: remember, any tool that lets your content editors remediate accessibility issues themselves is saving money overall.

5. What if nothing gets you far enough?
How much time would it take to develop something? What is that cost? What would ongoing year to year costs be? How does that compare with licensing costs? Except now you're left with one giant issue, even if this looks good...

What **can** you do when boxed in by a closed source editing environment?

(...practical limits notwithstanding, but that's still so much!)



The entire DOM is open to you, and then some. You can evaluate it, you can manipulate it, you can build on it. Javascript can do far more than just some fancy blinky text. You can even monkeypatch XMLHttpRequest to change XHRs (... but probably shouldn't)!

Translate the technical into practical, familiar contexts for people: e.g. tasks, not errors.

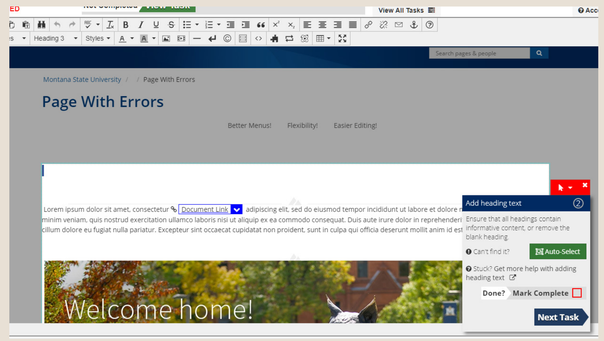
If you have a documented API, great, but you can still do fine without one: your browser's network panel or other sniffing tools like Fiddler can expose everything you need.

Integrate with your preexisting support and documentation systems... both for the people using your software and also for your own sake! Don't leave yourself maintaining the same data in multiple places: you could even AJAX in directly from your documentation.

Your browser is your friend in more ways than just letting you inject tools for people: did you know you can prototype html and CSS directly in your browser? This becomes crucial when working over a closed third party system, where you need to be aware of adverse interactions and also won't have common build tools like a servlet with auto-refresh.

Don't overload yourself at the outset: get a minimum viable product off the ground first.

Keep thinking outside of the box, from within it.



You're not alone, and neither are your content editors: with an easy remediation tool now in your hands, consider hiring some student workers to speed through stale content and maybe even to help with desktop document remediation efforts (the next frontier).

Where do you want to go next?

Once you've started, you have a platform of success not just for your content editors, but also for wherever you want to take it.

You can go from static to live scanning, even of live, open editors.

How about some Machine Vision to decide whether alt text adequately captures any prominent image text, and to flag images with too much text?

At times, opportunity rises out of situations that at first glance appear to only hold adversity.

MONTANA STATE UNIVERSITY
Mountains & Minds

https://www.montana.edu/web/projects/accessibility/