Baking Accessibility Into Our Web Workflows

Every so often I hear people complaining about accessibility and what should we do about it and more important whether we, as developers, should care about accessibility.

The tweet below made me think about this again in a larger context.

> Should commercial websites be required by law to implement a minimum level of WCAG? #a11y
>
> — dave.js (@_davejs) November 25, 2019

I'm reminded of a phrase that was mantra with the accessibility team at HTCTU while I was working on the Virtual Campus: *There is never time to do it right but there's always time to do it over*.

The Why

If we don't pay attention to accessibility sooner or later it'll come to bite us hard

The number of acccessibility-related lawsuits increased ignoring acccessibility for any size project is an invitation to lawsuits and, if not dealt with during development, can lead to needing to retrofit accessibikity into a project after it has launched or updated because now accessibility is a serious concern with legal and business ramifications.

US Federal requirements have been vague and hard to enforce, although the text of the Access Board latest rulemaking regarding section 508 is fairly clear:

> 2 Broad Application of Web Content Accessibility Guidelines 2.0
>
> The Revised 508 Standards and 255 Guidelines incorporate by reference the Web Content Accessibility Guidelines (WCAG) 2.0, a globally-recognized and technologically-neutral set of accessibility guidelines for Web content. For Section 508-covered ICT, all covered Web and non-Web content and

> software – including, for example, Web sites, intranets, word processing documents, portable document format documents, and project management software – is required, with a few specific exceptions, to conform to WCAG 2.0’s Level A and Level AA Success Criteria and Conformance Requirements. By applying a single set of requirements to Web sites, electronic documents, and software, the revised requirements adapt the existing 508 Standards to reflect the newer multifunction technologies (e.g., smartphones that have telecommunications functions, video cameras, and computer-like data processing capabilities) and address the accessibility challenges that these technologies pose for individuals with disabilities. For Section 255-covered ICT, electronic content and software that is integral to the use of telecommunications and customer premise equipment is required to conform to WCAG 2.0’s Level A and Level AA Success Criteria and Conformance Requirements. There are several exceptions related to non-Web documents and software.
>
> From Information and Communication Technology (ICT) Standards and Guidelines. Section 2: Broad Application of Web Content Accessibility Guidelines 2.0 as published in the Federal Register on January 18, 2017

Even if we had no clear guidelines, which the paragraph above provides, developers should get ahead of any requirement by matching an equivalent set of guidelines to what we think is required. I will address what this means in the context of automated testing below.

But it’s also a change of mindset about accessibility. It’s accepting that it’s an important part of our products and content and that it’s not just people with disabilities who will benefit from these acommodations.

The automated how

Rather than test for accessibility manually we should bake it into our development process; make it a requirement for successfully push to Git and then for the pull requests to be approved. This way, regardless of how big our project becomes we know we’re pushing accessible content and that, because of it, our application will be accessible every time we push it into version control or run CD/CI for our project.

Unlike the code in Incorporating accessibility evaluation into your build process I’ve chosen to concentrate on Gulp and use gulp-axe-webdriver to automate the accessibility check.

A standalone Gulp script to check for accessibility of one or more HTML documents under the `src/` directory (to any depth) against both WCAG AA requirements looks like this:

```
const gulp = require('gulp');
const axe = require('gulp-axe-webdriver');

fun'gulp-axe-webdriver'lity() {
  const options = {
    saveOutputIn: 'aXeReports/a11yReport.json',
    errorOnViolation: true,
    headless: false,
    rules: {},
   'aXeReports/a11yReport.json'howOnlyViolations: false,
    tags: ['wcag2aa'],
    threshold: 0,
    urls: ['src'wcag2aa'/**/*.html'],
    verbose: false
  };
  return axe(options);
}

exports.axe = checkAccessibility;
```

This is a good starting point as it forces us to review accessibility every time that we build our project but that may not be enough in all cases.

In addition to the gulp task that we can run manually, I’ve create a Git commit hoook, based on the example at codeinthehole, to run the task before allowing the commit to the repository to succeed. I do this because there are times when I’m lazy and don’t build a project before committing the code (I can always do it later) so the code doesn’t get tested for accessibility

```
# Stash non-commited changes
git stash -q --keep-index
# If node_modules doesn't exist
if [ ! -d "/node_modules/" ]
then
    "/node_modules/"# echo message to standard error
    echo "Directory node_modules not created"
    # and install
    npm install
fi
# Run gulp axe to check
gulp axe
RESULT=$?
# Pop the changes back to current directory
git stash pop -q
# If the exist code is not 0
# then exit with error code 1
[ $RESULT -ne 0 ] && exit 1
# Otherwise exit with error code 0
exit 0
## We're done!
```

Use this code to create a `pre-commit` file in your repository's `.git/hooks` directory either directly or by using a symbolic link; then make it executable running `chmod +x pre-commit`. The precommit hook will now intercept your commits and run `gulp axe` on your HTML content to check for accessibility issues.

If you haven't already, I also strongly suggest that you add `node_modules` to your repo's `.gitignore`.

Testing what we cannot automate

There are accessibility issues thatcan not be tested automatically. It's still up to each team to decide how to work those manual tests into the development process.

Links and resources

- Accessibility Testing Tools
- gulp-axe-webdriver
- axe core API
- Git Hooks