Table of Contents

Introduction

1 Research goal and methodology

1.1 Research goal

1.2 Research questions

RQ1: How good is the knowledge about accessibility standards, tools and best practices in the company currently?

RQ2: What kind of errors can be caught by running automated accessibility tests on a component library?

RQ3: To what extent can integrating automated testing to a component library's development pipeline help improve its compliance with WCAG standards?

RQ4: What are the biggest problems of integrating automated accessibility testing to a component library development workflow?

1.3 Methodology

2 Background

**TODO: test

2.1 Web accessibility

What is it and why should we care about it?

2.1.1 What standards should be followed?

2.2 Automated testing

2.2.1 How does it work?

2.2.2 Types of tools

2.2.3 Limitations of automated tests

3 Case study

(Byrne-Harber, 2021)
Steps

1. Set up automates accessibility issue detection in storybook adding a11y-addon + find a way to generate report of all issues.
   Data gathered from automated testing report:

   - How many occurrences in the accessibility violations report. This will show how many violations where detected from all the examples. Might contain the same issue multiple times.
   - How many unique issues will only count different violations for each component.
   - How many passed checks – this together with violations will show how many things where tested for each component – Most component have more than one story – the list will contain all different passed checks listed
   - How many valid checks – are the passed checks relevant to the component – only count the ones that are related to the component that the example is about.

2. Manual accessibility audit with other team members. We already had the add-on set up, and we used storybook preview of components for testing, so we looked at the violations reported by the addon-a11y also.

3. Comparison between manual and automated report

4. Research other possibilities. Testing new storybook test-runner to automatically run all a11y tests. How can we ignore some issues without losing the info.

5. Set up a new solution in a new component library. Will it help to ensure better accessibility from the beginning?

Pipedrive has been developing a sales CRM using mostly typescript and React. Accessibility has never been high priority and at this point it is not very easy to get started. We have a design system and a React based component library to keep the look and UX consistent. This seems like a good place to start with solving accessibility issues. The library is used widely in the company and developer from different teams contribute to it. If a button in the reusable library gets fixed 95% of the buttons in the web app that our customers use should be improved.
This should not be taken as a way to solve all accessibility problems, but a good first step to get started. Making changes in a reusable UI library should have a wide impact on the products overall accessibility and without reaching some basic level there first it could be hard to start testing views in the final product.

3.1 Current state of awareness about accessibility in the company

First I sent out a survey in our company Slack channels (see figure 1) to understand what is the knowledge and general approach on web accessibility in the company. It was shared in one front end developers channel with 212 members, designers channel with 73 members, accessibility channel with 31 members and in out component library channel with 124 members. Some people might also be in more than one of these channels. The aim was to reach people in the company who would be most likely to be using these tools and who would be likely contributors to the library. There were 7 questions and some of they also included a field for free text to give more details on the subject if they wanted.
In total 20 people replied to the survey - 6 designers and 14 developers, including one engineering manager. This does not give a full overview of the company, but it should give a good insight into what general opinions regarding accessibility might be. It is likely that developers and designers that are more involved with our component library and/or interested in accessibility where more likely to respond.
The results show that 10% of people who responded think their knowledge on accessibility is very good, while most think that knowledge level is average. 35% of responders know where do find resources about accessibility standards, 15% don't know and 50% know, but think they need more. They don't see that accessibility has high priority in the company currently, but at the same time 40% of responders think that following accessibility standards should be prioritized. There were several comments in the free text sections expressing saying they appreciate this subject being opened.

3.2 Adding tests to our component library

The next step was adding some automated test to our component library development workflow and observing their usefulness. The aim was to find something that we can integrate to our development workflow without a need to learn a new tool or add unnecessary complexity.
We use Storybook for our UI development. It is an open source software for UI development tool that allows you to work on one component at a time (Chromatic, n.d.-b). It allows us to render isolated UI components without integrating them to the final product right away. Our developers use it to test out the components they are developing locally. We also have a version of storybook available for anyone in the company to see with all the components. If this is the first place where elements will be rendered it seems logical to try to find a way to start testing them there.
To show a component you need to write a story - this means use it the way you would use it in the real world, and it will be rendered in a browser inside Storybook UI. It also has a sidebar for navigating between different examples, controls for additional tools and documentation. This means using a browser extension for accessibility tests would also test the UI around the actual example.

Storybook has a wide ecosystem of add-ons and one of them is addon-a11y. It uses Deque's axe-core to audit the html rendered for an isolated component (Chromatic, n.d.-a). Axe is an accessibility testing engine for websites and other HTML-based user interfaces. It includes WCAG 2.0 and 2.1 on level A and AA and promises to catch 57% of WCAG issues on average. (Deque Systems, 2023) This should be taken with a grain of salt, because we are intending to test isolated element and not the whole webpage and other studies comparing accessibility testing tools effectiveness have found the coverage to be lower (**needs citation**). This seemed like the best solution in our situation so addon-a11y was added to our component library's storybook. The tool is visible in the sidebar of every example. It shows all the checks that it has passed, all the violations that where found and any issues that could not be checked and might need manual testing. Every rule listed there also has reference to the html node that had the violation, explanation and links to Deque webpage with examples. This should be very useful in understanding and fixing these issues. It does not generate a report of all the issues found across the whole library for this another tool was used that will go through all the examples and generate a summary of all the violations found.

3.3 Manual accessibility audit

The second part is conducting a manual accessibility audit in the same library and comparing the results with the automatically generated test report results to determine what are its strengths and weaknesses and if testing isolated components poses any limitations.

3.4 Limitations running automated tests on a component library

3.5 Future plans/improvements

In the current version of Storybook there is no easy way to run all these tests in the development workflow. This will become much easier in the next major version. I have tested out this solution on a test library, and it seems like it would definitely be an improvement. Running the tests in CI would ensure that they are run every time someone makes a change and not only when we choose to. We could also block changes that don't pass the required accessibility checks.

4 Results

5 Discussion

5.1 Limitations of the study

5.2 Future research

Conclusion

References

Byrne-Harber, S. (2021). *Giving a damn about accessibility: A candid and practical handbook for designers.* UX Collective. https://www.accessibility.uxdesign.cc/

Chromatic. (N.d.-a). *Accessibility tests*. maintained by Chromatic. Retrieved 03/02/2023, from https://storybook.js.org/docs/react/writing-tests/accessibility-testing/

Chromatic. (N.d.-b). *Introduction to storybook for react*. maintained by Chromatic. Retrieved 03/01/2023, from https://storybook.js.org/docs/react/get-started/introduction

Deque Systems. (2023). *Axe-core*. Retrieved 03/02/2023, from https://github.com/dequelabs/axe-core

** All references are listed remove before submitting

Appendices

List of Figures and Tables

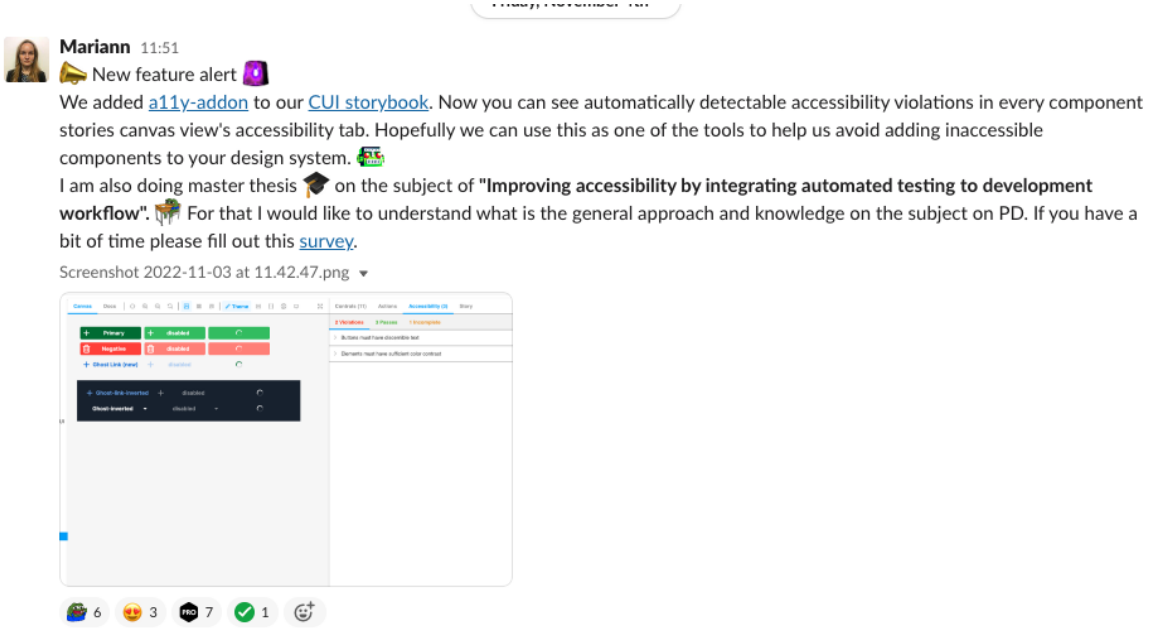

Figure 1: Message that was shared in the company Slack channel