Accessibility Support in Web Frameworks

Michael Longley
mxl2761@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Yasmine N. Elglaly
Western Washington University
Bellingham, Washington, USA
elglaly@wwu.edu

ABSTRACT

Despite the existence of accessibility testing tools, software still largely inaccessible mainly due to lack of awareness among developers and issues with existing tools [14, 18]. This motivated us to evaluate the accessibility support of development tools that do not require specific accessibility knowledge such as web frameworks. We tested the accessibility support of three JavaScript web frameworks; Angular, React, and Vue. For each of the three frameworks, we built a web application with 32 pages, each of which violated a single accessibility guideline. We found that only React generated a warning for one of the accessibility violations that is lack of label for non-text content. The rest of the accessibility violations went unnoticed by the three frameworks.

CCS CONCEPTS

• **Human-centered computing → Accessibility**.

KEYWORDS

web frameworks, accessibility guidelines

ACM Reference Format:
Michael Longley and Yasmine N. Elglaly. 2021. Accessibility Support in Web Frameworks. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '21), October 18–22, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3441852.3476531

1 INTRODUCTION

The professional responsibility of software developers calls on them to build accessible software [2]. There exist several accessibility tools to support developers in this endeavor [7, 22], in addition to a large set of accessibility guidelines [5]. However, developers' lack of awareness [14] and issues with existing tools [18] make inaccessible software [3, 25] an ugly reality. While there are several automated accessibility testing tools [7, 22], we wanted to learn whether existing front-end frameworks that developers frequently use and does not require specific accessibility knowledge, support developers in creating accessible web applications as they develop (during writing code) and not after the fact.

We identified three front-end JavaScript frameworks that are frequently used by web developers in recent years [19, 20]; Angular [4], React [16], and Vue [21]. We define "accessibility support" as producing an error or warning if the developer violates an accessibility guideline during development. We used WCAG 2.1 as our reference for accessibility guidelines [24]. We built a web application using each of the 3 web frameworks which violated an accessibility guideline, one per web page. We implemented the violations using the WCAG failure conditions [23] when possible and designed our own otherwise. The source code of the 3 inaccessible web applications is available on github [1]. We checked if any of the frameworks provide feedback to the developers, e.g., error or warning, when an accessibility guideline is violated. We also tested each page using WAVE Google Chrome plugin [22], a tool that evaluates the accessibility of websites. We used WAVE to validate the viability of our test cases. We found that React was the only web framework that was able to detect only one violation, which is lack of label for an image. WAVE was able to detect 6 violations.

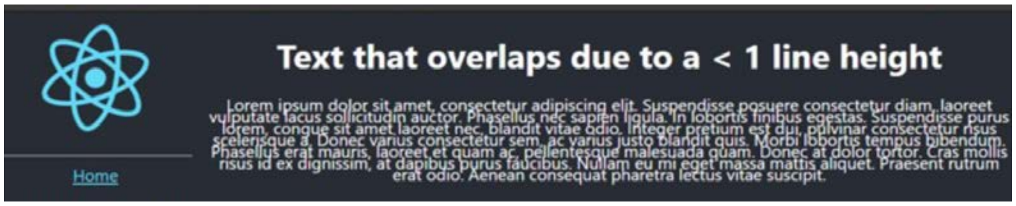
2 RELATED WORK

Web frameworks are libraries developers can use to more easily develop web applications. These libraries are used to abstract functionality that is consistent across all web applications such as page navigation and templating systems. By including these features in frameworks, it reduces the complexity of the individual websites and increases their maintainability. The frameworks we are evaluating, Angular, Vue, and React, have had a good record of analysis with respect to several metrics, e.g., performance, maintainability, and complexity [11–13, 15]. However, to the best of our knowledge there is no prior work on their accessibility as a metric. Previous work explained how accessibility problems in websites developed using JavaScript can be addressed [1, 6, 9, 10]. This is an indicator that JavaScript frameworks are good candidates for accessibility support analysis.

3 EVALUATION OF WEB FRAMEWORK ACCESSIBILITY SUPPORT

We created 3 inaccessible web applications using Angular Version 9.0.0., React version 16.12.0, and Vue version 2.6.11. Each framework was configured using the default configuration at the time of installation. All development was done in Visual Studio Code, by Microsoft. Each of the inaccessible web applications contained 32 pages, where each page contains an implementation of one accessibility violation. The accessibility violations we implemented represent failure conditions for the WCAG guidelines of levels A and AA. As a baseline for comparison, we evaluated the inaccessible web applications using WAVE plugin version 3.0.4. To create an accessibility violation, we implemented one of the guideline's failure conditions as described in the WCAG documentation [23]. We report on the implemented failure conditions in Table 1. For example,

[1]https://github.com/Web-Framework-Accessibility-Ratings



Text that overlaps due to a < 1 line height

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse posuere consectetur diam, laoreet vulputate lacus sollicitudin auctor. Phasellus nec sapien ligula. In lobortis finibus egestas. Suspendisse purus lorem, congue sit amet laoreet nec, blandit vitae odio. Integer pretium est dui, pulvinar consectetur risus scelerisque a. Donec varius consectetur sem, ac varius justo blandit quis. Morbi lobortis tempus bibendum. Phasellus erat mauris, laoreet et quam ac, pellentesque malesuada quam. Donec at dolor tortor. Cras mollis risus id ex dignissim, at dapibus purus faucibus. Nullam eu mi eget massa mattis aliquet. Praesent rutrum erat odio. Aenean consequat pharetra lectus vitae suscipit.

Figure 1: Example of an accessibility violation. The figure shows a web page with text space that is less than one point.

Table 1: The failure conditions we implemented for various accessibility guidelines and the framework response. A checkmark (✓) means the violation was automatically detected, and a tilde (~) means the violation was detected in some conditions.

WCAG Guideline	WCAG Failure Condition	Angular	React	Vue	WAVE
1.1.1 Non-text Content	F65	x	✓	x	✓
1.3.1 Info and Relationships	F92	x	x	x	x
1.3.2 Meaningful Sequence	F34	x	x	x	x
1.3.5 Identify Input Purpose	Designed	x	x	x	x
1.4.1 Use of Color	F73	x	x	x	x
1.4.2 Audio Control	F93	x	x	x	✓
1.4.3 Contrast (Minimum)	F83	x	x	x	✓
1.4.4 Resize Text	F69	x	x	x	x
1.4.5 Images of Text	Designed	x	x	x	x
1.4.11 Non-text Contrast	F78	x	x	x	x
1.4.12 Text Spacing	Designed	x	x	x	x
2.1.1 Keyboard	F55	x	x	x	x
2.1.2 No Keyboard Trap	F10	x	x	x	x
2.1.4 Character Key Shortcuts	Designed	x	x	x	x
2.2.1 Timing Adjustable	F40	x	x	x	x
2.2.2 Pause, Stop, Hide	F50	x	x	x	x
2.3.1 Three Flashes or Below Threshold	Designed	x	x	x	x
2.4.1 Bypass Blocks	Designed	x	x	x	x
2.4.3 Focus Order	F44	x	x	x	✓
2.4.4 Link Purpose (In Context)	F89	x	x	x	x
2.4.6 Headings and Labels	Designed	x	x	x	~
2.5.3 Label in Name	F96	x	x	x	x
3.1.1 Language of Page	Designed	x	x	x	✓
3.1.2 Language of Parts	Designed	x	x	x	x
3.2.2 On Input	F37	x	x	x	x
3.2.3 Consistent Navigation	F66	x	x	x	x
3.2.4 Consistent Identification	F31	x	x	x	x
3.3.1 Error Identification	Designed	x	x	x	x
3.3.2 Labels or Instructions	F82	x	x	x	x
4.1.1 Parsing	F77	x	x	x	x
4.1.2 Name, Role, Value	F59	x	x	x	x
4.1.3 Status Messages	Designed	x	x	x	✓

F65 is the instance of violation for the WCAG 1.1.1 Non-text Content guideline due to omitting the alt attribute or text alternative on img elements, area elements, and input elements of type "image" [8]. If no failure condition is provided by WCAG, we designed one, reported in Table 1 as "Designed". One such example is WCAG guideline 1.4.12, Text Spacing. We looked at what caused this guideline to fail. In this case we targeted the portion of the guideline saying that the line height must be at least 1.5 times the font size. With this in mind, we designed some CSS to violate this guideline, e.g. [*line-height: 0.75;*]. This CSS was applied to the content on the page shown in Figure 1 resulting in text which overlaps.

We looked in several locations to determine if the violation had been detected by the framework, such as the console of the application, the code itself and the console of the website.

Table 2: Examples for how web frameworks may automatically check for accessibility violations.

WCAG Guideline	Recommendation for Automated Testing
1.3.1 Info and Relationships	Check for roles in the DOM
1.3.2 Meaningful Sequence	Check for uses of the <pre> tag
1.4.1 Use of Color	Check for duplicate stylings in CSS
1.4.2 Audio Control	Check for media tags without control attribute
1.4.3 Contrast (Minimum)	Look at contrast styling
1.4.12 Text Spacing	Look for styling that reduces line height below threshold
2.1.1 Keyboard	Check for blur functions and their triggers
2.1.2 No Keyboard Trap	Check for circular tab indexing
2.1.4 Character Key Shortcuts	Check for keyup event listeners
2.4.3 Focus Order	Check for positive tab indexes
3.1.1 Language of Page	Lang attribute should not be able to be removed
4.1.1 Parsing	Validate the HTML
4.1.2 Name, Role, Value	Check for onClick functions on <span> tags

4 RESULTS AND DISCUSSION

The 3 frameworks failed in alerting the developers when any of the 32 accessibility violations occurred with only one exception, where React produced a warning for the WCAG 1.1.1 Non-text Content violation (see Table 1). The warning was generated in 3 places, the code causing the warning was underlined and hovering would trigger a tool tip indicating the issue, a similar warning was found in the shell running, and in the Google Chrome Developer's Tools Console. Noticeably, WAVE was able to detect 6 issues out of the 32 violations.

The detection of the frameworks and WAVE for the accessibility violations was very low. This shortcoming is similar to the state of mobile development tools [17]. The gap between the WAVE plugin and the frameworks is also concerning. With the WAVE plugin being able to detect 5 more types of violations, it proves that those violations can be automatically detected. While this is not necessarily the case for all instances of accessibility violations, it seems that at least a large amount more than the framework is detecting can be automatically evaluated (see Table 2 for examples). Along with this, the frameworks have a large advantage over tools like WAVE because they have access to all of the underlying code of the web apps.

5 CONCLUSION

We examined the level of accessibility support provided to developers by 3 of the most popular JavaScript web frameworks. We intentionally built inaccessible web applications where each has 32 accessibility violations. Unfortunately, we found that the 3 frameworks provided almost no feedback on accessibility violations. This finding mirrors developers' concerns on the lack of resources that facilitate building accessible software [14]. We argue that improving the detection of accessibility violations in web frameworks is feasible for several accessibility guidelines and can lead to the development of more accessible websites. By integrating accessibility guidelines into web frameworks, we support developers by tools and knowledge in building more accessible websites. In the future, we will work with developers to explore how to best integrate accessibility into their existing suite of tools.

REFERENCES

[1] Shadi Abou-Zahra, Judy Brewer, and Shawn Lawton Henry. 2013. Essential components of mobile web accessibility. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*. 1–4.

[2] ACM 2018. *ACM Code of Ethics and Professional Conduct.* Retrieved June 22, 2021 from https://www.acm.org/code-of-ethics

[3] Tahani Alahmadi and Steve Drew. 2017. Subjective Evaluation of Website Accessibility and Usability: A Survey for People with Sensory Disabilities. In *Proceedings of the 14th International Web for All Conference* (Perth, Western Australia, Australia) *(W4A '17)*. Association for Computing Machinery, New York, NY, USA, Article 11, 4 pages. https://doi.org/10.1145/3058555.3058579

[4] Angular 2016. *Angular: The modern web developer's platform.* Retrieved June 22, 2021 from https://angular.io/

[5] Mars Ballantyne, Archit Jha, Anna Jacobsen, J Scott Hawker, and Yasmine N El-Glaly. 2018. Study of accessibility guidelines of mobile applications. In *Proceedings of the 17th international conference on mobile and ubiquitous multimedia*. 305–315.

[6] Iyad Abu Doush, Faisal Alkhateeb, Eslam Al Maghayreh, and Mohammed Azmi Al-Betar. 2013. The design of RIA accessibility evaluation tool. *Advances in Engineering Software* 57 (2013), 1–7.

[7] Marcelo Medeiros Eler, José Miguel Rojas, Yan Ge, and Gordon Fraser. 2018. Automated accessibility testing of mobile apps. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 116–126.

[8] F65 2018. *F65: Failure of Success Criterion 1.1.1 due to omitting the alt attribute.* Retrieved June 22, 2021 from https://www.w3.org/TR/WCAG20-TECHS/F65.html

[9] Nádia Fernandes, Ana Sofia Batista, Daniel Costa, Carlos Duarte, and Luís Carriço. 2013. Three web accessibility evaluation perspectives for RIA. In *Proceedings of the 10th International cross-disciplinary conference on web accessibility*. 1–9.

[10] Becky Gibson and Richard Schwerdtfeger. 2005. Dhtml accessibility: solving the javascript accessibility problem. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*. 202–203.

[11] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. 2012. Comparative evaluation of javascript frameworks. In *Proceedings of the 21st International Conference on World Wide Web*. 513–514.

[12] Daniel Graziotin and Pekka Abrahamsson. 2013. Making sense out of a jungle of JavaScript frameworks. In *International Conference on Product Focused Software Process Improvement*. Springer, 334–337.

[13] Carl Lawrence Mariano. 2017. Benchmarking javascript frameworks. (2017).

[14] Rohan Patel, Pedro Breton, Catherine M Baker, Yasmine N El-Glaly, and Kristen Shinohara. 2020. Why Software is Not Accessible: Technology Professionals' Perspectives and Challenges. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–9.

[15] Paruj Ratanaworabhan, Benjamin Livshits, and Benjamin G Zorn. 2010. Jsmeter: comparing the behavior of javascript benchmarks with real web applications. *WebApps* 10 (2010), 3–3.

[16] React 2013. *React: A JavaScript library for building user interfaces.* Retrieved June 22, 2021 from https://reactjs.org/

[17] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O Wobbrock. 2020. An Epidemiology-inspired Large-scale Analysis of Android App Accessibility. *ACM Transactions on Accessible Computing (TACCESS)* 13, 1 (2020), 1–36.

[18] Camila Silva, Marcelo Medeiros Eler, and Gordon Fraser. 2018. A survey on the tool support for the automatic evaluation of mobile accessibility. In *Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*. 286–293.

[19] StackOverflow 2019. *Stack Overflow Developer Survey 2019*. Retrieved June 22, 2021 from https://insights.stackoverflow.com/survey/2019

[20] StackOverflow 2020. *Stack Overflow Developer Survey 2020*. Retrieved June 22, 2021 from https://insights.stackoverflow.com/survey/2020

[21] Vue 2014. *Vue: The Progressive JavaScript Framework*. Retrieved June 22, 2021 from https://vuejs.org/

[22] WAVE 2001. *WAVE Web Accessibility Evaluation Tool - WebAIM*. Retrieved June 22, 2021 from https://wave.webaim.org/

[23] WCAG2.1 2018. *Understanding Techniques for WCAG Success Criteria*. Retrieved June 22, 2021 from https://www.w3.org/WAI/WCAG21/Understanding/understanding-techniques

[24] WCAG2.1 2018. *Web Content Accessibility Guidelines (WCAG) 2.1*. Retrieved June 22, 2021 from https://www.w3.org/TR/WCAG21/

[25] Shunguo Yan and PG Ramachandran. 2019. The current status of accessibility in mobile apps. *ACM Transactions on Accessible Computing (TACCESS)* 12, 1 (2019), 1–31.