Accessibility in AI-Assisted Web Development

Peya Mowar
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
pmowar@andrew.cmu.edu

ABSTRACT

Despite extensive accessibility research, inaccessible websites remain stubbornly prevalent, partly due to limited accessibility awareness among web developers. The advent of code generation models presents an opportunity to guide accessibility-unaware developers toward enhanced accessibility practices. Our work examines the current state of accessibility in AI-assisted web development. Empirical evidence reveals code generation models serve as a double-edged sword, prompting a discussion on future research directions.

KEYWORDS

Web Accessibility, Web Content Accessibility Guidelines, Human-AI Interaction, Code Generation

ACM Reference Format:
Peya Mowar. 2024. Accessibility in AI-Assisted Web Development. In *The 21st International Web for All Conference (W4A '24), May 13–14, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3677846.3679054

1 INTRODUCTION

The web is indispensable for critical daily activities such as education, entertainment, and social interaction. Thus, there is active research and extensive prior work on web accessibility [4, 6, 14, 18, 22]. Despite these efforts, the actual progress has been minimal. Even today, most web pages do not comply with the accessibility standards outlined by the Web Content Accessibility Guidelines 2.0 (WCAG 2) [3]. Among the 1 million web pages evaluated by WebAim [7], a staggering 96.3% contained WCAG 2 failures, such as missing alternative text for images, unlabelled buttons, and low contrast text. Over the past four years, this value has only dropped by 1.5%.

Prior work has attributed this slow pace to a lack of awareness among industrial technology professionals and scarcity of resources for translating guidelines to practical code implementation [8, 11]. Patel *et al.* [12] suggested research into integrated development environment (IDE) plugins, to assist developers in addressing accessibility issues. Code generation models, already available as IDE plugins as illustrated in Figure 1, might have the potential to provide this assistance. Their usage offers the opportunity to evaluate their effectiveness in guiding developers toward improved accessibility practices. Conversely, there is a pressing need to study whether the increasing prevalence of these models introduces new accessibility-related vulnerabilities.

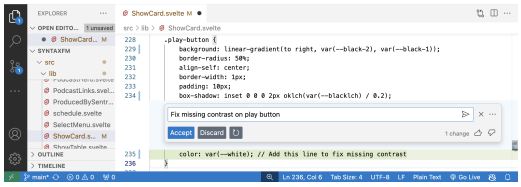

Figure 1: GitHub Copilot Plugin for Visual Studio Code

Thus, my thesis work aims to examine the current state of accessibility in AI-assisted web development. Specifically, it seeks to investigate the following broad research question: *"How does the accessibility of websites developed by industrial web developers (or proxies) with the assistance of code generative models compare to those developed without such assistance?"* This research will shed light on the effectiveness of AI developer tools in supporting web developers to adhere to current accessibility standards. Further, it will have broader implications on the evolution of developer practices with AI assistance and mitigation strategies to improve benchmarks of code language models on non-functional code requirements (such as accessibility, security, latency) [19].

2 RELATED WORK

This work draws and builds upon previous research on developer accessibility practices and the usability of AI-assisted development.

Developer Accessibility Practices. Prior studies have explored developers' perceptions of accessibility, revealing a shared empathy for accessibility considerations among all developers [16]. Still, accessibility is an afterthought in coding practices due to several factors. These include a lack of emphasis on accessibility in educational and industry standards [5, 10], the complexity of accessibility guidelines [8], and the perceived additional time, effort, and cost required for implementation [1]. Researchers have advocated for a "blended approach" [11], which treats accessibility as an integral non-functional requirement, akin to security and privacy, in UI/UX documentation. They have also suggested building appropriate IDE tooling for developers to assist them in catching and remediating accessibility violations [12].

Usability of AI-assisted Development. Recent usability research on AI-assisted development has examined the effectiveness and uptake of code suggestions among developers [9]. Findings indicate that developers did not frequently accept these suggestions as they did not appropriately consider the software requirements. This is because primary research on code generation models has mainly focused on functional correctness while often sidelining non-functional

W4A '24, May 13–14, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-1030-8/24/05.
https://doi.org/10.1145/3677846.3679054

requirements such as latency, maintainability, and security [19]. Consequently, there have been increasing concerns about the security implications of AI-generated code [17]. Such code, trained on vast datasets of human-written buggy code [13], is prone to contain code smells [15] and potentially introduce vulnerabilities [2]. Future research should focus on vulnerability mitigation strategies for better usability of these agents and improved efficiencies in AI-assisted development.

3 EMPIRICAL STUDY SETUP

In this study, we focus specifically on the empirical evaluation of GitHub Copilot as an extension in a natural software development environment, i.e., Visual Studio Code. Copilot is an AI developer tool by GitHub, trained on publicly available code from GitHub repositories, that generates contextually relevant code suggestions. We conduct a comparative analysis by reviewing GitHub Copilot's code suggestions against the developer's source code for active websites in real-world web development scenarios. For this study, we assume that the code committed by the developer on GitHub lacks any AI assistance and hence serves as our ground truth. We follow the steps below to assess GitHub Copilot's code suggestions.

Step 1: Task Selection. We select six actively updated websites that span various categories defined by the IAB Content Taxonomy [21], such as business, education, and entertainment, etc., open-sourced on GitHub. These sites exhibit varying levels of adherence to the Web Content Accessibility Guidelines (WCAG) 2.0 standards. Our selection of web development tasks focuses on the recently resolved "UI enhancement" issues on these websites, specifically choosing those with the potential for at least one of the six most frequently occurring WCAG 2 failures [7] and requiring modifications to a single file.

Step 2: Prompt Definition. First, we revert the source code to the commit just before the developer addressed the issue. Then, we prompt GitHub Copilot with the functional requirements by providing the issue description and relevant file paths as comments. We ensure that no other file is opened as an additional context window, and prompt precisely at the exact location in the same file where the original edits were made. We keep accepting Copilot's suggestions until no further recommendations are provided. In instances where the developer had modified existing lines of code, we select those code lines and utilize the Copilot Chat feature for prompting. This setup aims to mirror the developer's interaction with Copilot as closely as possible.

Step 3: Code Evaluation. The final step involves comparing the developer source code with the suggestions applied from Copilot. We limit our evaluation of WCAG failures to the modifications highlighted in the code difference file, disregarding errors outside these bounds. Through qualitative analysis, we ascertain whether Copilot's interventions mitigate existing accessibility barriers or inadvertently introduce new ones.

4 PRELIMINARY RESULTS

Opportunities. Copilot frequently generated placeholder attributes to enhance accessibility, such as consistently appending an `alt` attribute to the `img` tag, thereby encouraging developers to provide alternative text for images. However, these placeholders often lacked content; when populated, the provided text was usually not pertinent. Therefore, a future direction for my thesis would be to explore the adoption of such nudges among developers. Moreover, Copilot demonstrated capability in addressing specific accessibility requirements, mainly when directly prompted or when these requirements coincided with functional requirements. An example includes the task of enhancing button contrast against the website background. Copilot effectively identified and remedied the contrast issues, although it did not strictly adhere to the WCAG-recommended contrast ratios. This indicates Copilot's potential to improve web accessibility, albeit with limitations such as requiring explicit instruction.

Challenges. Copilot's output heavily depended on the existing accessibility of a website: it produced inaccessible elements for websites already lacking in accessibility and tried to mimic accessible design patterns in more compliant websites, though not always successfully. Similarly, human developers, also contextual learners, recognized and incorporated accessibility as a crucial aspect of design when they encountered accessible websites, striving to create accessible elements themselves. However, based on our limited dataset, Copilot has not yet been able to exceed the accessibility standards set by professional developers. Further, we additionally noticed instances where Copilot hallucinated inaccessible elements, unrelated to the functional requirements, extrapolated from irrelevant contexts. For example, Copilot added unlabelled YouTube video links for research papers in a research portfolio, which were never provided as assets. Thus, another thread in my thesis will explore these vulnerabilities and developers' reactions to them in more detail.

5 FUTURE WORK AND CONCLUSION

As discussed above, the initial efforts in this research were concentrated on a qualitative accessibility evaluation of code generation models on web development tasks. As the next step, we plan to conduct a comprehensive task-based study with industrial web developers or proxies such as CS students or freelance developers [20]. This study will assess how these developers leverage Copilot in web development tasks, focusing on code compliance with WCAG 2 guidelines. Recognizing that both humans and Copilot learn contextually, we will also explore the effects of task sequencing on accessible versus inaccessible websites, aiming to understand if and how developers and Copilot can mutually enhance their learning and application of accessibility standards. Additionally, we intend to automate the evaluation process and identify quantitative accessibility metrics that accurately reflect the nuances of AI-assisted web development.

6 ACKNOWLEDGEMENTS

I would like to thank my advisors, Prof. Jeffrey Bigham and Prof. Aaron Steinfeld, for their invaluable guidance and support. I am also grateful to the Carnegie Mellon University (CMU) Accessibility Group. This work is approved by IRB #2024-030 and supported by the CMU School of Computer Science.

REFERENCES

[1] Humberto Lidio Antonelli, Sandra Souza Rodrigues, Willian Massami Watanabe, and Renata Pontin de Mattos Fortes. 2018. A survey on accessibility awareness of Brazilian web developers. In *Proceedings of the 8th international conference on software development and technologies for enhancing accessibility and fighting info-exclusion*. 71–79.

[2] Owura Asare, Meiyappan Nagappan, and N Asokan. 2023. Is github's copilot as bad as humans at introducing vulnerabilities in code? *Empirical Software Engineering* 28, 6 (2023), 129.

[3] Ben Caldwell, Michael Cooper, Loretta Guarino Reid, Gregg Vanderheiden, Wendy Chisholm, John Slatin, and Jason White. 2008. Web content accessibility guidelines (WCAG) 2.0. *WWW Consortium (W3C)* 290 (2008), 1–34.

[4] Paul T Chiou, Ali S Alotaibi, and William GJ Halfond. 2023. BAGEL: An Approach to Automatically Detect Navigation-Based Web Accessibility Barriers for Keyboard Users. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.

[5] Michael Crabb, Michael Heron, Rhianne Jones, Mike Armstrong, Hayley Reid, and Amy Wilson. 2019. Developing accessible services: Understanding current knowledge and areas for future support. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[6] Reinaldo Ferraz, Ana Duarte, João Bárbara, Adriano C.M. Pereira, and Wagner Meira. 2023. A platform to check website compliance with web accessibility standards. In *Proceedings of the 20th International Web for All Conference* (<conf-loc>, <city>Austin</city>, <state>TX</state>, <country>USA</country>, </conf-loc>) *(W4A '23)*. Association for Computing Machinery, New York, NY, USA, 75–78. https://doi.org/10.1145/3587281.3587289

[7] Policy Practice Institute for Disability Research. 2024. *The WebAIM Million*. Retrieved February 20, 2024 from https://webaim.org/projects/million/

[8] Jonathan Lazar, Alfreda Dudley-Sponaugle, and Kisha-Dawn Greenidge. 2004. Improving web accessibility: a study of webmaster perceptions. *Computers in human behavior* 20, 2 (2004), 269–288.

[9] Jenny T Liang, Chenyang Yang, and Brad A Myers. 2024. A large-scale survey on the usability of ai programming assistants: Successes and challenges. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–13.

[10] Lilu Martin, Catherine Baker, Kristen Shinohara, and Yasmine N Elglaly. 2022. The Landscape of Accessibility Skill Set in the Software Industry Positions. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*. 1–4.

[11] Maulishree Pandey and Tao Dong. 2023. Blending Accessibility in UI Framework Documentation to Build Awareness. In *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility*. 1–12.

[12] 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.

[13] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2022. Asleep at the keyboard? assessing the security of github copilot's code contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 754–768.

[14] Helen Petrie, Andreas Savva, and Christopher Power. 2015. Towards a unified definition of web accessibility. In *Proceedings of the 12th International Web for All Conference*. 1–13.

[15] Rohith Pudari and Neil A Ernst. 2023. From Copilot to Pilot: Towards AI Supported Software Development. *arXiv preprint arXiv:2303.04142* (2023).

[16] Cynthia Putnam, Kathryn Wozniak, Mary Jo Zefeldt, Jinghui Cheng, Morgan Caputo, and Carl Duffield. 2012. How do professionals who create computing technologies consider accessibility?. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility*. 87–94.

[17] Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. 2023. Lost at c: A user study on the security implications of large language model code assistants. *arXiv preprint arXiv:2208.09727* (2023).

[18] Brian Sierkowski. 2002. Achieving web accessibility. In *Proceedings of the 30th annual ACM SIGUCCS conference on User services*. 288–291.

[19] Manav Singhal, Tushar Aggarwal, Abhijeet Awasthi, Nagarajan Natarajan, and Aditya Kanade. 2024. NoFunEval: Funny How Code LMs Falter on Requirements Beyond Functional Correctness. *arXiv preprint arXiv:2401.15963* (2024).

[20] Mohammad Tahaei and Kami Vaniea. 2022. Recruiting Participants With Programming Skills: A Comparison of Four Crowdsourcing Platforms and a CS Student Mailing List. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (, New Orleans, LA, USA,) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 590, 15 pages. https://doi.org/10.1145/3491102.3501957

[21] Webshrinker. 2024. *IAB Categories*. Retrieved February 20, 2024 from https://docs.webshrinker.com/v3/iab-website-categories.html#iab-categories

[22] Yeliz Yesilada, Giorgio Brajnik, Markel Vigo, and Simon Harper. 2012. Understanding web accessibility and its drivers. In *Proceedings of the international cross-disciplinary conference on web accessibility*. 1–9.