

Coding Non-Visually in Visual Studio Code: Collaboration Towards Accessible Development Environment for Blind Programmers

JOOYOUNG SEO, School of Information Sciences, University of Illinois at Urbana-Champaign, USA

MEGAN ROGGE, Microsoft, USA

This paper delineates a fruitful collaboration between blind and sighted developers, aiming to augment the accessibility of Visual Studio Code (VSCode). Our shared journey is portrayed through examples drawn from our interaction with GitHub issues, pull requests, review processes, and insider’s releases, each contributing to an improved VSCode experience for blind developers. One key milestone of our co-design process is the establishment of an accessible terminal buffer, a significant enhancement for blind developers using VSCode. Other innovative outcomes include Git Diff audio cues, adaptable verbosity settings, intuitive help menus, and a targeted accessibility testing initiative. These tailored improvements not only uplift the accessibility standards of VSCode but also provide a valuable blueprint for open-source developers at large. Through our shared dedication to promoting inclusivity in software development, we aim for the strategies and successes shared in this paper to inspire and guide the open-source community towards crafting more accessible software environments.

CCS Concepts: • **Human-centered computing** → **Accessibility design and evaluation methods**.

Additional Key Words and Phrases: nonvisual programming, accessibility, integrated development environment, visual studio code

ACM Reference Format:

JooYoung Seo and Megan Rogge. 2018. Coding Non-Visually in Visual Studio Code: Collaboration Towards Accessible Development Environment for Blind Programmers. In . ACM, New York, NY, USA, 15 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

An integrated development environment (IDE) is an application that conveniently provides essential functions for the entire programming process, including source editing, compiling and interpreting, and debugging. IDEs have become an essential tool for not only software developers but also STEM engineers and data scientists in many fields to efficiently manage their computing environments [4, 6, 8]. However, blind developers¹ are not able to take advantage of the many features that graphical user interface (GUI)-based IDEs offer [13]. For example, syntax highlighting, code autocompletion and autosuggestion, diagnostics and linting, variable watches and breakpoints are underutilized even among experienced blind programmers, and many blind developers are still working manually with simple text like Notepad, along with runtime

¹We use the identity-first language (i.e., blind people) instead of the person-first language (i.e., people with visual impairments or vision loss) when addressing this population, guided by the perspective of the National Federation of the Blind.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Association for Computing Machinery.

Manuscript submitted to ACM

and compile terminals [2, 3, 10]. Behind this problem are intertwined issues of accessibility and learnability. Because different IDEs use different architectures and have different levels of accessibility compliance, blind developers face a new learning curve each time they use an IDE. Blind developers also face the additional challenge of learning the non-visual workaround of accessing an IDE with a screen reader [10]. Although there is a community of blind programmers called Program-L [9] where blind programmers help and support each other, IDEs remain a daunting barrier for blind people.

These difficulties are a major socio-technical barrier to blind developers reaching their full potential in the computing field and to social and professional participation. From the perspective of the social model [11], which recognizes that an individual’s disability may stem from structures and cultures that sociotechnically limit their access rather than from physical, sensory, cognitive, or emotional issues, we can see that IDE accessibility issues are no longer a group-specific problem that blind people must endure, but a collective task for the technology community to reduce barriers together. Specifically, to address these issues, blind and sighted developers need to work together to understand the challenges that blind developers face in using IDEs and then collaboratively find ways to address those challenges. This perspective is consistent with the “interdependent framework” [5, 7] that other accessibility researchers have advocated to move away from the dependency of accessibility on the individual with disabilities and instead view accessibility as a shared responsibility of people with and without disabilities and the environment surrounding them.

This paper is the empirical product of blind and sighted developers who have thought deeply about these issues and actively collaborated. We describe how the first author, who is blind, and the second author, who is sighted, have been working together to make the open source IDE Visual Studio Code (VSCode) non-visually accessible and what specific accessibility features have been implemented as a result of our collaboration. In the following sections, we start with some background on how our collaboration began, then present our methods and deliverables. Finally, we’ll share some insights from our collaboration.

2 BACKGROUND: VISUAL STUDIO CODE AND ACCESSIBILITY

Visual Studio Code is a lightweight, free, and powerful open-source code editor² which runs on the desktop and on the web. It is available for Windows, macOS, and Linux. It has built-in support for JavaScript, TypeScript, and Node.js and a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET, etc.). Accessibility has been a core priority for VSCode since its inception. Among the many architectural elements of VSCode, the following, in particular, has contributed to its accessibility. First, VSCode is a cross-platform application built with the Chromium-based Electron Framework. In other words, VSCode is an application built using web technologies, which gives it the flexibility to follow web accessibility guidelines [1] and respond to the accessibility of various screen readers and assistive technologies regardless of the operating system. Second, Monaco, the primary editor of VSCode, has its own screen reader compatibility mode, which is designed to be selectively turned on and off depending on the user’s intent. Third, Microsoft’s xterm.js terminal, used by VSCode, also provides a separate screen reader accessibility switch in accordance with the Web Accessibility Guidelines. Finally, VSCode is an open-source project where anyone can suggest and fix features on GitHub, and a daily insiders version is built

²In this paper, the terms integrated development environment and code editor are used interchangeably.

so that real users can quickly use the alpha version and provide feedback to the developers, which in turn leads to a higher quality, user-centered stable version.

The accessibility benefits of VSCode and tips on how to take advantage of them have been shared among members of the Program-L mailing list, a community of blind programmers. In addition, due to its growing popularity among blind programmers, there has been a recent spate of research and development of accessible plug-ins based on VSCode [12, 16]. Nevertheless, the fact that VSCode is accessible compared to other IDEs does not necessarily mean that it is easy for blind programmers to use. For example, there is still a constant stream of questions on Program-L about VSCode, not only about its basic usage, but also about features that have already been made accessible in VSCode, such as the terminal, debugging, and the Jupyter Notebook extension, which suggests that many blind programmers are often frustrated by the tricky usability of VSCode accessibility. The following section describes how the authors of this paper collaborated to address this usability issue of VSCode accessibility.

3 METHODS

3.1 Author Profiles and Collaboration Context

The first author of this paper is blind with only light perception, currently working as an assistant professor in the School of Information Sciences at the University of Illinois at Urbana-Champaign. At the university, he teaches introductory data science courses using R and Python to undergraduate and graduate students. As a lifelong non-visual programmer, he has experience with a variety of IDEs, including Visual Studio, Eclipse, and Net Bean, and text editors such as Emacs/Emacspeak, VIM, and NotePad++, on Linux, Mac, and Windows operating systems, using a variety of screen readers (e.g., JAWS, NVDA, Narrator, VoiceOver, and Orca) and refreshable braille displays. He is a certified professional in accessibility core competencies (CPACC) from the International Association of Accessibility Professionals and has contributed code to a number of open-source data science projects to improve screen reader accessibility, including RStudio IDE Server and the web-based data science dashboard Shiny, reproducible technical publishing systems (e.g., R Markdown, bookdown, and Quarto), and the data table package gt. He is also a member of Program-L. In this community, he has experienced first-hand the challenges that blind programmers face in using IDEs and how they overcome them by interacting with other blind programmers and participating in discussions. To improve these community-wide challenges, he created his first issue on the Microsoft VSCode public GitHub site on May 31, 2020, and has since created a total of 164 contributions (87 issues; 76 post comments and mentions; 1 pull request) to actively suggest usability improvements for blind programmers in VSCode and interact with other open source developers (see Appendix Section A for more details).

The second author is a VSCode software engineer. She has worked on the product since graduating from the University of North Carolina at Chapel Hill in 2020 with the highest distinction and highest honors for her research and work with Dr. Gary Bishop on semi-automated gaming for users with a wide range of disabilities. About 10 months ago, Megan requested to take over responsibility for the product’s accessibility. Since then, she has been working closely with JooYoung and the community to understand accessibility issues and collaborate on solutions.

3.2 Co-Design and Expert Review

Our collaborative approach utilized the strategies of co-design and expert review. The co-design methodology fosters a joint creation process between the user and developer, enabling the developer to grasp the user’s requirements and, in turn, develop a product aligning with these needs [15]. In this framework, JooYoung acted as an expert, given his multi-faceted role as a regular VSCode user, an experienced open-source contributor, a data science educator, and an accessibility professional. He outlined his varied computing experiences to Megan and swiftly assessed her accessibility patches.

Their communication began asynchronously via GitHub, debating on issues and potential solutions. Following a few weeks of this pattern, they mutually agreed that scheduled meetings could prove more efficient and productive. JooYoung’s wealth of ideas and insights complemented Megan’s eagerness to learn and her drive to enhance the product’s accessibility. In these sessions, JooYoung demonstrated his use of VS Code by sharing his screen on Zoom, posing queries, and suggesting alterations. Conversely, Megan provided her insights, questioned various aspects, and noted down bugs or features requiring attention. These exchanges facilitated Megan’s understanding of JooYoung’s usage of VS Code and enabled JooYoung to comprehend the product components, which could otherwise remain confusing or undiscovered.

Despite the implementation of regular meetings, asynchronous communication via GitHub and email persisted. Megan regularly composed follow-up emails encapsulating their meeting discoveries prior to circulating them to the entire team. JooYoung further scrutinized these issues, providing comments if anything was overlooked or during the fix-testing process.

4 CO-DESIGNED DELIVERABLES

While nearly all VS Code accessibility fixes and features within the past year are products of this collaboration, below are several of the highlights.

4.1 Terminal Buffer

As discussed in Section 2, xterm.js, the terminal UI utilized by VSCode, incorporates a screen-reader accessibility mode for blind people. However, a discernible gap emerged between accessibility (the ability to access information) and usability (the convenience of use), which led to recurring concerns among blind programmers.

Consider the following scenario: you type and execute the command `echo hello; echo world;` in the terminal. You will observe `hello` and `world` as two separate lines of output. The existing accessibility mode of xterm.js presented this content through a screen reader using an aria-live alert and permitted a line-by-line review of the terminal output history with the `Ctrl+UpArrow` and `Ctrl+DownArrow` keys. This works well for short and simple outputs, but for lengthy outputs with intricate error messages or computational results, a swift speech-to-text message is insufficient for capturing substantial information in human working memory.

An additional concern is that `Ctrl+Up/DownArrow` navigation keys, designed to review terminal history, deliver the entire contents of the focused line to the screen reader as a single object. This makes detailed examination of terminal contents on a character or word basis challenging. Blind users had to switch the reading mode using the screen reader’s virtual cursor (i.e., browse mode in NVDA; QuickNav mode in

VoiceOver) to review the terminal content more thoroughly. To resume terminal input, they had to disable the virtual cursor and return to forms mode (focus mode in NVDA; QuickNav off in VoiceOver), leading to significant inconvenience.

JooYoung initiated a discussion on the official Microsoft VSCode GitHub page, bringing attention to these issues and proposing solutions (microsoft/vscode#98918: Terminal output div container should be more accessible for screen readers). Megan, meanwhile, developed terminal shell integration, a feature allowing VS Code to comprehend terminal activities, facilitating user-friendly command navigation, command output copying, and more. JooYoung demonstrated that the terminal buffer remained inaccessible for screen reader users, as it didn't support arrow key navigation. He proposed that the output view's accessible experience be integrated into the terminal. Upon discussing with a colleague, Megan incorporated the same underlying component into the terminal, making the previously inaccessible terminal buffer navigable via arrow keys for blind users.

More specifically, he suggested replacing the terminal output with a text editor buffer that supported standard arrow-key navigation. The implementation, requiring over a year of technical experimentation and collaborative testing, yielded fruitful results. Initial efforts to redirect the terminal output web container, designated as "list", to aria "document" or "textbox" landmarks proved unsatisfactory due to varying screen reader and platform support levels for aria. The terminal output was then converted into a text area with "contenteditable" and "readonly" attributes, which also did not gel with the screen reader's speech buffer. Eventually, we created a separate accessible terminal buffer by transferring the terminal output to VSCode's native Monaco editor, ensuring optimal accessibility and usability for all blind users on all platforms and screen readers. This feature, well-received by many blind users in the Program-L community, was officially introduced in the VSCode stable version 1.75. Figure 1 illustrates how the terminal buffer operates, demonstrating VoiceOver focusing on an error line in a task terminal and reading it out loud.

4.2 Git Diff and Audio Cues

Git has been around for decades as a version control tool like SVN, but its popularity has really taken off with the rise of open-source social coding platforms based on Git, such as GitHub and GitLab. Naturally, there have been many personal and social needs for blind people to utilize Git in collaborative environments. git is originally a Unix-based command-line tool, so in terms of accessibility, blind people can use a screen reader to fully utilize Git in a terminal. However, since Git has over 100 core Git commands, and the number of possible combinations could be in the millions, using Git via the command line takes a lot of effort and time to become proficient. In response, various tools have emerged that allow you to use Git as a GUI, and VSCode is a very popular IDE that supports a collaborative environment using Git.

Git provides a track changes feature that allows you to compare changes between files in an asynchronous collaborative environment, called `git diff`. Literally, the `git diff` command compares and shows the differences between a file and a file, or between a commit and a commit, with newly added lines in green and + and removed lines in red and - prefixed. VSCode had always provided an accessible `git diff` function for screen reader users. With the files or commits users want to compare open, pressing F7 (Go to Next Difference) and Shift+F7 (Go to Previous Difference) would skip to the area where the differences are, prefixing the line with the change with a + or - sign to indicate the nature of the change. Of course, this approach was fine from an accessibility standpoint, but there was room for improvement in terms of

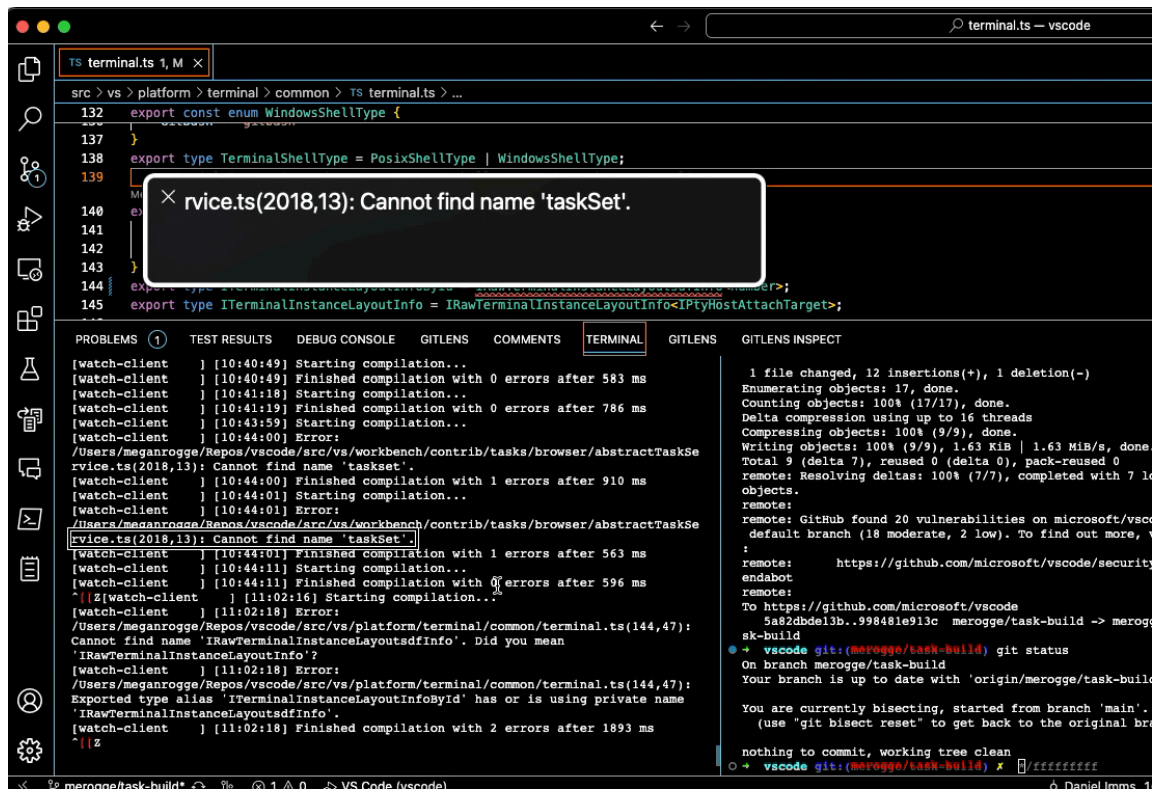


Fig. 1. VoiceOver reads “rvice.ts(2018,13: Cannot find name ‘taskSet’”

usability and convenience for blind users. For example, visual affordances like color coding and +- signs in `git diff` allowed sighted people to skim quickly, but blind people had to listen to additional speech prefixes, pronounced + (plus) and - (dash), serially and wait for information before each change. Furthermore, depending on the punctuation pronunciation settings of the screen reader, the +- sign could be omitted and delivered to the screen reader.

To address this, JooYoung suggested adding non-visual, non-speech, and audible affordances to `git diff` in addition to +- signatures, so that blind people can hear and understand them easily [microsoft/vscode#147226](https://github.com/microsoft/vscode/issues/147226): [Accessibility] Consider adding audio cues for diffs (added / deleted code). Audio cues are non-speech sound effects, an accessibility feature that VSCode and Microsoft’s other IDE, Visual Studio, have just begun to support, and TV Raman demonstrated their usefulness in non-visual programming many years ago when he developed Emacspeak, referring to them as earcons as an alternative to icons [14]. For example, audio cues allow the editor to quickly recognize if the current line of code contains an error or a warning, instead of just saying “error” or “warning” verbally, the editor will read out the unique sound associated with the error or warning. These sounds can also be delivered in parallel with text-to-speech information from a screen reader, allowing blind programmers to quickly perceive the context of the code, similar to the benefits of

quickly scanning code with different color coding for those who receive visual feedback on code with their eyes.

JooYoung had several Zoom meetings with Megan and Amnon Freidlin (Microsoft’s sound designer), and through an iterative process, finalized the three audio cues used in the `git diff` context. These were the diff line Inserted sound, which is heard when something new is added (+), the diff line Deleted sound, which is heard when something existing is removed (-), and the diff line Modified sound, which is heard when something existing is modified (+-, -+). Our success came with some trial and error. For example, an early problem was that the Diff Line Inserted and Diff Line Deleted sounds had a similar range and texture, making it difficult to distinguish between them. JooYoung realized that this was a common complaint in Program-L beyond her personal experience, so she worked with the sound designer to test and finalize a sample file that was as self-explanatory as possible and didn’t interfere with the sound of screen reader speech. Of course, we had to leave the potential issue of the static audio cues we chose not being able to adequately accommodate users with hearing impairments in certain ranges as a future work in progress, but this feature greatly improved the usability of our non-visual programming.

4.3 Verbosity Settings and Help Menus

JooYoung created issues pointing out places where minor tweaks to the order or content of an aria label could yield massive productivity improvements for screen reader users. Megan fixed some such instances and pointed team members toward others, providing guidance about best practices going forward.

Megan started self hosting with a screen reader shortly after this in order to proactively identify other problems. She felt overwhelmed by the noise and noticed some content was repeated ad nauseum, so created an issue and sought the feedback of JooYoung, who suggested that screen reader verbosity settings remedy this and a similar approach could be applied to VS Code’s aria content.

Additionally, JooYoung shared that while it was helpful to meet and learn about the new features via our meetings, most screen reader users did not have this luxury. Megan and her colleague, Daniel, brainstormed about a discoverable way for screen reader users to find out about terminal features. Upon terminal focus, an aria label conveyed how to access the terminal’s accessibility help menu. To reduce noise, this hint could be disabled with a verbosity setting. Since then, help menus and verbosity settings have been added for the Copilot inline and panel chat, notebook, and other features. For example, the terminal accessibility help menu contains helpful information for screen reader users such as commands to run (Figure 2). A screen reader user can use arrow keys to read the content line by line, character by character.

4.4 Accessibility Testing Initiative

The VS Code team tests new features at the end of every month before each release. Megan noticed that while the team tested each platform - MacOS, Linux, and Windows, they were not testing the screen reader experience. A new protocol has been established to ensure better coverage going forward; the iteration following a feature’s release, the team will test the feature using screen readers. Retroactive testing of features is currently underway to make up for this historical oversight. JooYoung’s creation of issues about old and new features alike inspired and justified this initiative.

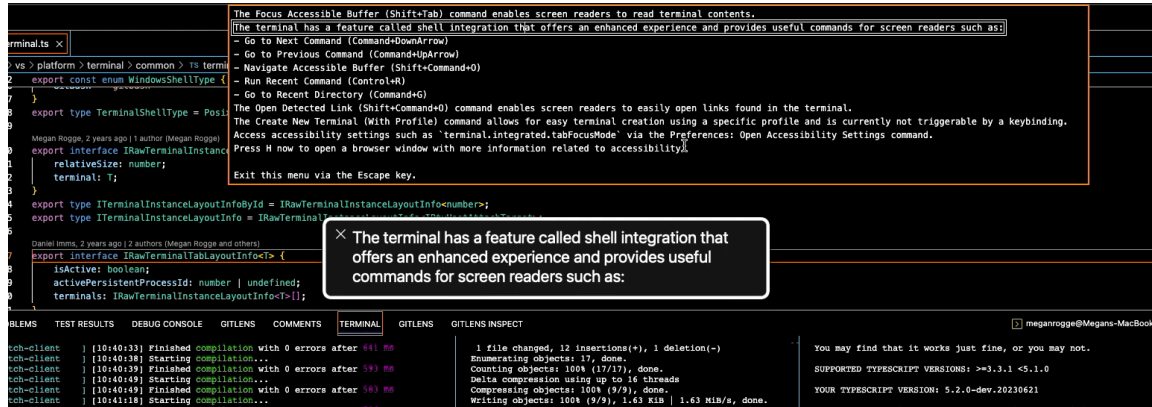


Fig. 2. A line in the help menu is focused and VoiceOver reads “The terminal has a feature called shell integration that offers an enhanced experience and provides useful commands for screen readers such as:”

5 DISCUSSION AND CONCLUSION

Our ongoing collaboration between sighted and blind developers underlines the importance and potential of improving accessibility in open-source tools, specifically illustrated by our work on Visual Studio Code. A notable success is the terminal accessible buffer, a solution initially created for screen reader users, that proved to be beneficial for a wider user base as seen here. Furthermore, knowledge sharing, epitomized by Megan and JooYoung’s livestream presentation, is a cornerstone in promoting an inclusive, democratic coding and programming culture.

The urgency of addressing accessibility early in the development process has been a vital lesson from our collaboration. Postponing such efforts can create significant barriers for screen reader users, therefore prioritizing these enhancements is crucial.

Moreover, we are enthusiastically working on Copilot, an AI-based, Language Model (LLM) feature, with a dedicated focus on its accessibility for blind programmers. Anticipating the considerable potential of Copilot, we are committed to ensuring that blind programmers can utilize this technology without delay, rather than having to wait for subsequent accessibility improvements.

Drawing these threads together, our co-design efforts emphasize the necessity of creating a more equitable coding environment where all programmers, regardless of their disabilities, can participate fully. Our ongoing engagement and activities on GitHub aim to serve as a motivation and guide for other open-source developers towards similar endeavors of inclusive development.

ACKNOWLEDGMENTS

We extend our gratitude to Program-L, an online community of blind programmers, for their invaluable feedback and testing of VS Code, as well as their insightful accessibility suggestions. We also appreciate the VS Code team for their commitment to accessibility. Special thanks to Isidor Nikolic, Kai Maetzel, and Daniel Imms for their dedication and invaluable insights; to Raymond Zhao and Roberto Perez for enhancing the site’s accessibility; to José Vilmar Estácio de Souza and Amnon Freidlin for their diligent testing and collaboration.

REFERENCES

- [1] [n.d.]. Web Content Accessibility Guidelines (WCAG) 2.1. <https://www.w3.org/TR/WCAG21/>.
- [2] Khaled Albusays and Stephanie Ludi. 2016. Eliciting Programming Challenges Faced by Developers with Visual Impairments: Exploratory Study. In *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '16)*. Association for Computing Machinery, New York, NY, USA, 82–85. <https://doi.org/10.1145/2897586.2897616>
- [3] Khaled Albusays, Stephanie Ludi, and Matt Huenerfauth. 2017. Interviews and Observation of Blind Software Developers at Work to Understand Code Navigation Challenges. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '17)*. Association for Computing Machinery, New York, NY, USA, 91–100. <https://doi.org/10.1145/3132525.3132550>
- [4] Raghavendra Rao Althar and Debabrata Samanta. 2021. Building Intelligent Integrated Development Environment for IoT in the Context of Statistical Modeling for Software Source Code. In *Multimedia Technologies in the Internet of Things Environment*, Raghvendra Kumar, Rohit Sharma, and Prasant Kumar Pattnaik (Eds.). Springer, Singapore, 95–115. https://doi.org/10.1007/978-981-15-7965-3_7
- [5] Cynthia L. Bennett, Erin Brady, and Stacy M. Branham. 2018. Interdependence as a Frame for Assistive Technology Research and Design. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '18)*. Association for Computing Machinery, New York, NY, USA, 161–173. <https://doi.org/10.1145/3234695.3236348>
- [6] James H. Cross and T. Dean Hendrix. 2007. jGRASP: An Integrated Development Environment with Visualizations for Teaching Java in CS1, CS2, and Beyond. *Journal of Computing Sciences in Colleges* 23, 2 (Dec. 2007), 170–172.
- [7] Lilian De Greef, Dominik Moritz, and Cynthia Bennett. 2021. Interdependent Variables: Remotely Designing Tactile Graphics for an Accessible Workflow. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, Virtual Event USA, 1–6. <https://doi.org/10.1145/3441852.3476468>
- [8] Jan Janssen, Sudarsan Surendralal, Yuri Lysogorskiy, Mira Todorova, Tilmann Hickel, Ralf Drautz, and Jörg Neugebauer. 2019. Pyiron: An Integrated Development Environment for Computational Materials Science. *Computational Materials Science* 163 (June 2019), 24–36. <https://doi.org/10.1016/j.commatsci.2018.07.043>
- [9] Jazette Johnson, Andrew Begel, Richard Ladner, and Denae Ford. 2022. Program-L: Online Help Seeking Behaviors by Blind and Low Vision Programmers. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–6. <https://doi.org/10.1109/VL/HCC53370.2022.9833106>
- [10] S. Mealin and E. Murphy-Hill. 2012. An Exploratory Study of Blind Software Developers. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Innsbruck, 71–74. <https://doi.org/10.1109/VLHCC.2012.6344485>
- [11] Mike Oliver. 2013. The Social Model of Disability: Thirty Years On. *Disability & Society* 28, 7 (Oct. 2013), 1024–1026. <https://doi.org/10.1080/09687599.2013.818773>
- [12] Venkatesh Potluri, Maulishree Pandey, Andrew Begel, Michael Barnett, and Scott Reitherman. 2022. CodeWalk: Facilitating Shared Awareness in Mixed-Ability Collaborative Software Development. In *The 24th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, Athens Greece, 1–16. <https://doi.org/10.1145/3517428.3544812>
- [13] Venkatesh Potluri, Priyan Vaithilingam, Suresh Iyengar, Y. Vidya, Manohar Swaminathan, and Gopal Srinivasa. 2018. CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–11. <https://doi.org/10.1145/3173574.3174192>
- [14] T. V. Raman. 1996. Emacspeak—a Speech Interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '96)*. Association for Computing Machinery, New York, NY, USA, 66–71. <https://doi.org/10.1145/238386.238405>
- [15] Elizabeth B.-N. Sanders and Pieter Jan Stappers. 2008. Co-Creation and the New Landscapes of Design. *CoDesign* 4, 1 (March 2008), 5–18. <https://doi.org/10.1080/15710880701875068>
- [16] Bernhard Stöger, Klaus Miesenberger, Walther Neuper, Makarius Wenzel, and Thomas Neumayr. 2022. Designing an Inclusive and Accessible Mathematical Learning Environment Based on a Theorem Prover. In *Computers Helping People with Special Needs (Lecture Notes in Computer Science)*, Klaus Miesenberger, Georgios Kouroupetroglou, Katerina Mavrou, Roberto Manduchi, Mario Covarrubias Rodriguez, and Petr Penáz (Eds.). Springer International Publishing, Cham, 47–55. https://doi.org/10.1007/978-3-031-08648-9_7
- [1] [n.d.]. Web Content Accessibility Guidelines (WCAG) 2.1. <https://www.w3.org/TR/WCAG21/>.
- [2] Khaled Albusays and Stephanie Ludi. 2016. Eliciting Programming Challenges Faced by Developers with Visual Impairments: Exploratory Study. In *Proceedings of the 9th International Workshop on Cooperative and Human*

- Aspects of Software Engineering (CHASE '16)*. Association for Computing Machinery, New York, NY, USA, 82–85. <https://doi.org/10.1145/2897586.2897616>
- [3] Khaled Albusays, Stephanie Ludi, and Matt Huenerfauth. 2017. Interviews and Observation of Blind Software Developers at Work to Understand Code Navigation Challenges. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '17)*. Association for Computing Machinery, New York, NY, USA, 91–100. <https://doi.org/10.1145/3132525.3132550>
 - [4] Raghavendra Rao Althar and Debabrata Samanta. 2021. Building Intelligent Integrated Development Environment for IoT in the Context of Statistical Modeling for Software Source Code. In *Multimedia Technologies in the Internet of Things Environment*, Raghvendra Kumar, Rohit Sharma, and Prasant Kumar Pattnaik (Eds.). Springer, Singapore, 95–115. https://doi.org/10.1007/978-981-15-7965-3_7
 - [5] Cynthia L. Bennett, Erin Brady, and Stacy M. Branham. 2018. Interdependence as a Frame for Assistive Technology Research and Design. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '18)*. Association for Computing Machinery, New York, NY, USA, 161–173. <https://doi.org/10.1145/3234695.3236348>
 - [6] James H. Cross and T. Dean Hendrix. 2007. jGRASP: An Integrated Development Environment with Visualizations for Teaching Java in CS1, CS2, and Beyond. *Journal of Computing Sciences in Colleges* 23, 2 (Dec. 2007), 170–172.
 - [7] Lilian De Greef, Dominik Moritz, and Cynthia Bennett. 2021. Interdependent Variables: Remotely Designing Tactile Graphics for an Accessible Workflow. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, Virtual Event USA, 1–6. <https://doi.org/10.1145/3441852.3476468>
 - [8] Jan Janssen, Sudarsan Surendralal, Yury Lysogorskiy, Mira Todorova, Tilmann Hickel, Ralf Drautz, and Jörg Neugebauer. 2019. Pyron: An Integrated Development Environment for Computational Materials Science. *Computational Materials Science* 163 (June 2019), 24–36. <https://doi.org/10.1016/j.commatsci.2018.07.043>
 - [9] Jazette Johnson, Andrew Begel, Richard Ladner, and Denae Ford. 2022. Program-L: Online Help Seeking Behaviors by Blind and Low Vision Programmers. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–6. <https://doi.org/10.1109/VL/HCC53370.2022.9833106>
 - [10] S. Mealin and E. Murphy-Hill. 2012. An Exploratory Study of Blind Software Developers. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Innsbruck, 71–74. <https://doi.org/10.1109/VLHCC.2012.6344485>
 - [11] Mike Oliver. 2013. The Social Model of Disability: Thirty Years On. *Disability & Society* 28, 7 (Oct. 2013), 1024–1026. <https://doi.org/10.1080/09687599.2013.818773>
 - [12] Venkatesh Potluri, Maulishree Pandey, Andrew Begel, Michael Barnett, and Scott Reitherman. 2022. CodeWalk: Facilitating Shared Awareness in Mixed-Ability Collaborative Software Development. In *The 24th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, Athens Greece, 1–16. <https://doi.org/10.1145/3517428.3544812>
 - [13] Venkatesh Potluri, Priyan Vaithilingam, Suresh Iyengar, Y. Vidya, Manohar Swaminathan, and Gopal Srinivasa. 2018. CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–11. <https://doi.org/10.1145/3173574.3174192>
 - [14] T. V. Raman. 1996. Emacspeak—a Speech Interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '96)*. Association for Computing Machinery, New York, NY, USA, 66–71. <https://doi.org/10.1145/238386.238405>
 - [15] Elizabeth B.-N. Sanders and Pieter Jan Stappers. 2008. Co-Creation and the New Landscapes of Design. *CoDesign* 4, 1 (March 2008), 5–18. <https://doi.org/10.1080/15710880701875068>
 - [16] Bernhard Stöger, Klaus Miesenberger, Walther Neuper, Makarius Wenzel, and Thomas Neumayr. 2022. Designing an Inclusive and Accessible Mathematical Learning Environment Based on a Theorem Prover. In *Computers Helping People with Special Needs (Lecture Notes in Computer Science)*, Klaus Miesenberger, Georgios Kouroupetroglou, Katerina Mavrou, Roberto Manduchi, Mario Covarrubias Rodriguez, and Petr Penáz (Eds.). Springer International Publishing, Cham, 47–55. https://doi.org/10.1007/978-3-031-08648-9_7

A APPENDIX: VSCODE ACCESSIBILITY DISCUSSIONS ON GITHUB

The following are JooYoung’s GitHub contributions, including issues, pull requests, comments, and mentions, related to accessibility up to the time of this paper’s submission.

```
$ gh search issues --repo microsoft/vscode --include-prs --involves jooyoungseo -L 200
```

```

issue microsoft/vscode 186857 open [Accessibility] Checkboxes are unlabeled in `Export Profile...` bug
issue microsoft/vscode 186754 open data science audio and text graph for visually impaired person feat
issue microsoft/vscode 186679 open Alert that the help hint has been disabled bug, accessibility 2023-06-21T17:23:35Z
issue microsoft/vscode 186678 open Change accessible buffer command navigation keybinding for screen reader feature-request, verified, a
issue microsoft/vscode 186676 open add accessible view provider for inline chat response feature-request, verified, a
issue microsoft/vscode 186675 open when next/previous ghost text suggestion is shown, we don't alert screen reader users feature-request, verified, a
issue microsoft/vscode 186673 closed have accessible view for ghost text completions feature-request, accessibility, on-t
issue microsoft/vscode 186659 closed Sticky scroll for screen reader users feature-request, accessibility, on-t
issue microsoft/vscode 186514 closed Closing accessibility hint doesn't stop VoiceOver from reading it feature-request, verified, a
issue microsoft/vscode 185705 closed Consider providing screen reader with the chat response for inline chat feature-request, verified, a
issue microsoft/vscode 185691 closed Consider which audio cues for chat experience should be enabled by default feature-request, verified, a
issue microsoft/vscode 185565 open Accessibility: Cannot turn off audio cues on a language level feature-request, verified, a
issue microsoft/vscode 185371 open Review usage of `aria-live: assertive`, `alert` throughout the code feature-request, verified, a
issue microsoft/vscode 185155 open Alert screen reader users that something has occurred when `clear` is pressed feature-request, verified, a
pr microsoft/vscode 185153 merged prevent screen reader from reading a user's chat request on enter feature-request, verified, a
issue microsoft/vscode 184357 open [Accessibility]: Make syntax highlight accessible to screen reader user feature-request, verified, a
issue microsoft/vscode 184176 closed Add notebook accessibility help menu feature-request, verified, a
issue microsoft/vscode 184173 closed Accessibility: Take out extra messages from Notebook verbosity FALSE feature-request, verified, a
issue microsoft/vscode 183567 open Explore improvements to notifications when using a screen reader feature-request, verified, a
issue microsoft/vscode 183363 closed Make accessibility help generic feature-request, accessibility, on-t
issue microsoft/vscode 183030 closed Reading suggestions or autocomplete of extensions bug, verified, a
issue microsoft/vscode 182682 open Merge editor accessibility accessibility, merge-editor 2023-06-21T17:23:35Z
pr microsoft/vscode 182666 merged outweigh normal editor accessibility help menu 2023-06-30T23:23:35Z
issue microsoft/vscode 181732 closed Accessibility: Make drag-and-drop accessible via keyboard bug, accessibility, on-t
issue microsoft/vscode 181139 open Accessibility: Make Tab key focus restricted to the currently open view feature-request, verified, a
issue microsoft/vscode 181060 closed BAccessibility: Pressing Shift+Tab key in Ctrl+F moves to terminal area feature-request, verified, a
issue microsoft/vscode 180970 closed provide alt text for image outputs feature-request, accessibility, on-t
pr microsoft/vscode 180776 merged fix windows quick fixes 2023-06-09T23:22:27Z
issue microsoft/vscode 180729 closed Investigate merge editor accessibility bug, accessibility 2023-06-21T17:23:35Z
issue microsoft/vscode 180725 open interacting with components should be consistent accessibility, on-t
issue microsoft/vscode 180653 closed [Accessibility]: Present content first in References Treeview help feature-request, verified, a
issue microsoft/vscode 180221 open Accessibility: Ctrl+Down/UpArrows does not work in Tree find controls feature-request, verified, a
issue microsoft/vscode 180216 closed [Accessibility]: Typing characters does not move focus in File Explorer feature-request, verified, a
issue microsoft/vscode 180176 open [Accessibility]: Consider replacing audioCues.lineHasInlineSuggestions with audioCues.lineHasInlineSuggestions feature-request, verified, a
issue microsoft/vscode 180083 open Accessibility: Make Debug Console follow terminal tabFocusMode accessibility, on-t
issue microsoft/vscode 180049 open [Accessibility]: Support filtering symbol types in Document Symbol View feature-request, verified, a
issue microsoft/vscode 179981 open Focus does not stay in the editor area after sending selection to terminal feature-request, verified, a

```

573	issue	microsoft/vscode	179979	closed	Accessibility: Ctrl+Up/DownArrows does not work in terminal
574	issue	microsoft/vscode	179970	closed	Accessibility: Make Ctrl+RightArrow expand all in tree views
575	issue	microsoft/vscode	179969	closed	Make filtering more flexible in Problem View info-needed,
576	issue	microsoft/vscode	179967	open	Accessibility: Generalize Ctrl+DownArrow and Ctrl+UpArrow to
577	issue	microsoft/vscode	179964	open	Accessibility: Improve Problem View search input accessib
578	issue	microsoft/vscode	179718	closed	search result aria label should prioritize content over locat
579	issue	microsoft/vscode	179717	closed	Problems aria label should prioritize content over location
580	issue	microsoft/vscode	179716	open	Allow configuring what is included in the accessible buffer
581	issue	microsoft/vscode	179283	closed	[Accessibility]: Make "Go to line" announce focused line aft
582	issue	microsoft/vscode	179272	closed	[Accessibility]: Add shortcut keys to jump between executed
583	issue	microsoft/vscode	179123	closed	[Accessibility]: `Search: Find in Files, Control+Shift+F` co
584	issue	microsoft/vscode	178935	closed	[Accessibility] Audio cues stopped working in Chrome acce
585	issue	microsoft/vscode	178915	closed	[Accessibility] Make sticky scroll view line indentation acc
586	issue	microsoft/vscode	177755	closed	[Accessibility]: Switching editor (Ctrl+Tab) does not work f
587	issue	microsoft/vscode	177697	closed	`Terminal: navigate accessible buffer` does not work sometim
588	issue	microsoft/vscode	177696	closed	Inline suggestion is read twice by the screen reader bug,
589	issue	microsoft/vscode	177694	closed	add command to repeat most recent notification accessibility
590	issue	microsoft/vscode	177029	closed	[Accessibility]: `Set Selection Anchor` and `Select from And
591	issue	microsoft/vscode	176779	open	Make error in line audio cue configurable feature-request,
592	issue	microsoft/vscode	176521	open	[Accessibility] Support task completion/failure audio cues i
593	issue	microsoft/vscode	176293	closed	Prefer SVG renderers for image output to improve screen read
594	issue	microsoft/vscode	176292	open	improve screen reader context and navigation of Cell outputs
595	issue	microsoft/vscode	176290	open	consider default keybindings for go to next / previous cell
596	issue	microsoft/vscode	176286	closed	`allowNavigateToSurroundingCells` should be false when scree
597	issue	microsoft/vscode	176242	open	Notify screen reader users that a VS Code update is availabl
598	issue	microsoft/vscode	175986	open	Allow VS Code extensions to trigger audio cues feature-req
599	pr	microsoft/vscode	175823	merged	provide screen reader with inline suggestions 2023-04-21T2
600	issue	microsoft/vscode	175743	open	Output of Jupyter notebook cells is not intuitively accessib
601	issue	microsoft/vscode	175432	closed	[Accessibility] Pressing Ctrl+M key (toggle tabFocusMode) sh
602	issue	microsoft/vscode	175348	open	Refine error on line audio cue feature-request, accessibili
603	issue	microsoft/vscode	175341	closed	[Accessibility] Some thoughts on error in line audio cue
604	issue	microsoft/vscode	175282	open	[Accessibility] Do not use title attribute when labeling but
605	issue	microsoft/vscode	175177	closed	[Accessibility] Remove repeated word from the terminal help
606	issue	microsoft/vscode	175175	closed	[Accessibility] "Go to Recent Directory (Control+G)" instruc
607	issue	microsoft/vscode	175162	closed	assign different default keybinding for focusing the accessi
608	issue	microsoft/vscode	175140	closed	Add a command that accepts a notification's default action
609	issue	microsoft/vscode	175111	closed	[Accessibility] Redundant read-only terminal buffer needs to
610	issue	microsoft/vscode	175105	closed	[Accessibility] Reconsider the UI design for {"editor.screen
611	issue	microsoft/vscode	175014	closed	Replace diff line modified/deleted/ inserted audio cues with

625	issue	microsoft/vscode	175013	closed	On focus of the accessible buffer, if the last command failed, play
626	issue	microsoft/vscode	175012	closed	Use more succinct audio cue when terminal command fails feature-request
627	issue	microsoft/vscode	175011	closed	position the cursor at the end of the accessible buffer by default
628	issue	microsoft/vscode	174857	closed	[Accessibility] Line-by-line audio cues are not played when column p
629	issue	microsoft/vscode	174800	closed	[Accessibility] Shift+Tab is always forced to go to ally terminal bu
630	issue	microsoft/vscode	174798	closed	[Accessibility] Remove redundant 4-5 blank lines from the terminal a
631	issue	microsoft/vscode	174797	closed	[Accessibility] python repl content is not parsable in the ally term
632	issue	microsoft/vscode	174793	closed	[Accessibility] Consider adding a setting to preserve focus in ally
633	issue	microsoft/vscode	174606	merged	add setting for aria-live assertive alert for ghost text 2023-04-
634	pr	microsoft/vscode	174368	closed	Play audio cue when a command exits with non-zero code feature-request
635	issue	microsoft/vscode	174367	closed	Mention `Terminal: Create Terminal with Profile` in terminal ally he
636	issue	microsoft/vscode	174365	closed	Suggest screen reader users migrate from `cmd prompt` -> `pwsh` feat
637	issue	microsoft/vscode	174362	open	Next suggestion isn't read bug, accessibility 2023-03-13T09:34:44Z
638	issue	microsoft/vscode	174360	open	When in `tabFocusMode`, assign a different keybinding for inline sug
639	issue	microsoft/vscode	174359	open	Add more audio cues feature-request, accessibility 2023-03-02T16:40:44Z
640	issue	microsoft/vscode	174079	closed	Add symbol provider for terminal accessible buffer feature-request
641	issue	microsoft/vscode	173622	closed	No indication of ghost text and actions via screen reader accessibil
642	issue	microsoft/vscode	173532	closed	I can no longer access terminal accessibility buffer with orca bug
643	issue	microsoft/vscode	173452	closed	[Accessibility] Add page up/down support for accessible buffer feat
644	issue	microsoft/vscode	173451	closed	[Accessibility] Make `editor.action.toggleTabFocusMode` configurabl
645	issue	microsoft/vscode	172606	closed	[Accessibility] Go to next/previous change commands don't provide de
646	issue	microsoft/vscode	172582	closed	[Accessibility] Terminal ally buffer is not automatically updated
647	issue	microsoft/vscode	172525	closed	[Accessibility] Error audio cues are not played on a character leve
648	issue	microsoft/vscode	172523	closed	[Accessibility]: Audio Cues are notplayed a against swift arrow nav
649	issue	microsoft/vscode	172465	closed	Reduce noise for screen reader users feature-request, verified, a
650	issue	microsoft/vscode	172458	closed	in diff view, line selection shouldn't happen on cursor move bug
651	issue	microsoft/vscode	172399	closed	tab has to be pressed twice to go back to the terminal buffer from a
652	pr	microsoft/vscode	172276	merged	xterm@5.2.0-beta.21 2023-03-11T23:23:27Z
653	issue	microsoft/vscode	172204	closed	Screen reader accessibility mode reads terminal contents character b
654	issue	microsoft/vscode	172149	open	[Accessibility] Command history is not readable in terminal input f
655	issue	microsoft/vscode	172024	closed	[Accessibility] Ctrl+M (editor.action.toggleTabFocusMode) does not v
656	issue	microsoft/vscode	172007	closed	Terminal accessibility buffer does not read output upon enter bug
657	issue	microsoft/vscode	172006	closed	Make terminal accessibility buffer read only upstream, accessibil
658	issue	microsoft/vscode	171918	closed	[Accessibility] Support Home and End keys in Open Detected Link vie
659	issue	microsoft/vscode	171916	closed	[Accessibility] Ctrl+Shift+O does not close Open Detected Link view
660	issue	microsoft/vscode	171914	closed	[Accessibility] Make terminal ally buffer even more accessible acc
661	issue	microsoft/vscode	171755	open	Code lens is not accessible via screen reader accessibility, under
662	issue	microsoft/vscode	171544	closed	sometimes audio cues don't play when going to next/previous differen
663	issue	microsoft/vscode	171429	closed	[Accessibility] Diff editor cursor position is not preserved after c

677 issue microsoft/vscode 171426 open [Accessibility] Diff editor cursor position is not preserved
 678 issue microsoft/vscode 171256 closed [Accessibility] Trigger diff audio cues against standard arr
 679 issue microsoft/vscode 171253 open [Accessibility] Allow users to customize the audio cue play
 680 issue microsoft/vscode 171200 closed support screen reader reading the line and audio cues when g
 681 issue microsoft/vscode 171199 open Accessibility getting started experience feature-request,
 682 pr microsoft/vscode 170985 merged support screen reader reading the line when go to next/previous
 683 issue microsoft/vscode 170971 closed [Accessibility]: Allow users to replace default sound file
 684 issue microsoft/vscode 169853 closed Explore plain content editable element for terminal buffer i
 685 issue microsoft/vscode 168746 open [Accessibility] Word wrap does not work in diff view (F7 and
 686 pr microsoft/vscode 167349 closed fix #167348: add aria-live 2023-02-16T17:31:02Z
 687 issue microsoft/vscode 167348 closed [Accessibility] div.monaco-tokenized-source requires aria-li
 688 issue microsoft/vscode 168814 open Need a clearer landmark and label for notebook output area
 689 issue microsoft/vscode 166518 closed Add audio cues for Go to Next/ Previous Change commands feat
 690 issue microsoft/vscode 166472 open [Accessibility] Add an option to allow Alt+F5 to jump to the
 691 issue microsoft/vscode 165863 closed Hitting spacebar does not replay currently focused audio cue
 692 issue microsoft/vscode 165357 closed [Accessibility] Audio Cues still doesn't work in github.dev
 693 issue microsoft/vscode 165161 open [Accessibility] Open Folder dialog controls do not have acce
 694 issue microsoft/vscode 164988 closed [Accessibility]: Screen readers do not read currently focuse
 695 issue microsoft/vscode 163506 open [Accessibility] Provide icon info to screen readers feature-
 696 issue microsoft/vscode 160301 open [Accessibility] Some long file content line is not correctly
 697 issue microsoft/vscode 159029 open Merge editor accessibility improvements accessibility, merge
 698 issue microsoft/vscode 155919 closed [Accessibility] Support `Live Share` audio cues in `Help: Li
 699 issue microsoft/vscode 155655 closed [Accessibility] For easier code navigation, add jump to next
 700 issue microsoft/vscode 154027 closed [Accessibility]: Terminal output is not read in real time on
 701 issue microsoft/vscode 147607 closed [Accessibility] Unlabelled `codicon` buttons info-needed,
 702 issue microsoft/vscode 147386 closed [Accessibility] Add audio cues for indentation levels feat
 703 issue microsoft/vscode 147230 open Play audio-cues for auto-suggestions feature-request, acc
 704 issue microsoft/vscode 147226 closed [Accessibility] Consider adding audio cues for diffs (added
 705 issue microsoft/vscode 147190 closed [Accessibility] Audio Cues doesn't work in web editor bug,
 706 issue microsoft/vscode 143185 closed Problems to access the preview of a markdown file using orca
 707 issue microsoft/vscode 142983 closed [Terminal accessibility] JAWS does not speak anything agains
 708 issue microsoft/vscode 141529 closed Webviews displaying results of an API call with Restclient e
 709 issue microsoft/vscode 135920 closed [Accessibility] "xterm-accessibility" class div does not hav
 710 issue microsoft/vscode 135035 closed [Accessibility] GitHub Web Editor: Cannot configure accessib
 711 issue microsoft/vscode 133876 closed [Accessibility] Assign a keyboard shortcut key to Focus Term
 712 issue microsoft/vscode 133805 closed `Shift+Alt+R` for `Reveal in File Explorer` doesn't work whe
 713 issue microsoft/vscode 133773 closed [Accessibility] "document" role is needed for "monaco-hover"
 714 issue microsoft/vscode 132275 closed [Accessibility] Add "document" role to webview widget veri
 715 issue microsoft/vscode 131295 open [Accessibility] Character is not read properly in terminal i

729	issue	microsoft/vscode	131090	closed	[Accessibility] NVDA and JAWS do not read focused auto-suggestion i
730	issue	microsoft/vscode	130565	closed	Notify Screenreader Users When Inline Suggestions Or Decorations Av
731	issue	microsoft/vscode	121735	closed	Terminal input does not work with NVDA bug, important, accessibili
732	issue	microsoft/vscode	113482	closed	Tab code-completion does not work in terminal input against screen
733	issue	microsoft/vscode	111255	open	VS Code native notebook accessibility improvement debt, accessibi
734	issue	microsoft/vscode	105425	closed	Garbage characters are inserted if you come back from terminal outpu
735	issue	microsoft/vscode	103095	closed	Auto-complete popup puts redundant "item" prefix per suggested code
736	issue	microsoft/vscode	98918	closed	Terminal output div container should be more accessible for screen
737	issue	microsoft/vscode	95570	closed	Support terminal link keyboard navigation feature-request, access
738	issue	microsoft/vscode	90408	open	Feature request: Accessibility support for Jupyter notebooks in VSC
739					
740					
741					
742					
743					
744					
745					
746					
747					
748					
749					
750					
751					
752					
753					
754					
755					
756					
757					
758					
759					
760					
761					
762					
763					
764					
765					
766					
767					
768					
769					
770					
771					
772					
773					
774					
775					
776					
777					
778					
779					
780					