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.

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 developers[1] 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

---

[1] We 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.

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 editor[2] 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, and more. 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

---

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

version is built 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 daily 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 his accessibility patches. Our communication began asynchronously via GitHub, debating on issues and potential solutions. Following a few weeks of this pattern, we 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 VSCode 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 worked well for short and simple outputs, but for lengthy outputs with intricate error messages or computational results, a swift speech-to-text message was insufficient for capturing substantial information in human working memory.

An additional concern is that `Ctrl+Up/DownArrow` navigation keys, designed to review terminal history, delivered the entire contents of the focused line to the screen reader as a single object. This made a 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 (see 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 supporting 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 "read only" 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. In Figure 1, a screenshot depicts the terminal accessible buffer in action. A screen reader is shown focusing on an error message from a task terminal and audibly announcing: "[watch-client] [12:41:01] Error: /Users/meganrogge/Repos/vscode/vscode/src/vs/workbench/contrib/accessibility/browser/accessibilityContributions.ts(198,63): ')' expected.". Users can navigate the displayed content using standard arrow keys without having to switch between different navigation modes.

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 users to utilize Git as a GUI, and VSCode is a very popular IDE that supports a collaborative environment using Git.

Git's `git diff` feature enables users to track changes and compare differences between files or commits in an asynchronous collaborative setting. Using this command, new additions are marked in green with a + prefix, while deletions appear in red with a - prefix. VSCode ensured that the `git diff` feature is accessible to screen reader users. When users had the desired files or commits open for comparison, they could quickly navigate to the differences using the `F7` key (Go to Next Difference) and `Shift+F7` (Go to
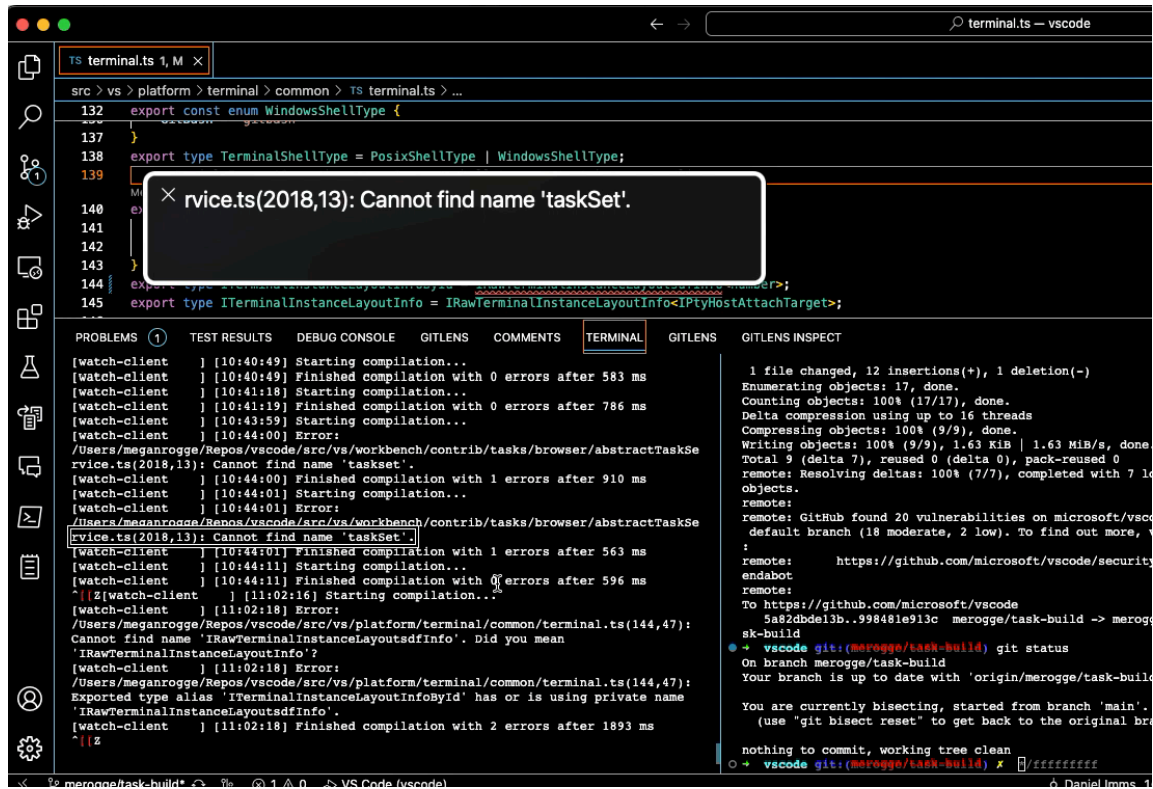
Fig. 1. The terminal accessible buffer.

Previous Difference). These differences were prefixed with a + or - sign to denote additions or deletions, respectively. Of course, this approach was fine from an accessibility standpoint, but there was room for improvement in terms of 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 to the screen reader.

To ameliorate this, JooYoung suggested adding non-visual, non-speech, and audible affordances to `git diff` in addition to +- signs, so that blind people can hear and understand them easily (see microsoft/vscode#147226). Audio cues, often referred to as "earcons", are non-speech sound effects that enhance accessibility. While VSCode and Microsoft's Visual Studio have only recently started supporting them, their importance in non-visual programming was highlighted decades ago by TV Raman, a blind computer scientist. He introduced the concept of earcons as a counterpart to visual icons when he developed Emacspeak [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 his personal experience, so he 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 the 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 important information for screen reader users such as commands to run like "Run Recent Command (Ctrl+R)" (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

JooYoung consistently provided feedback on GitHub, highlighting issues with both old and new features in VS Code. His active engagement and insights were pivotal in spotlighting an overlooked area. Megan, a key accessibility member of the VSCode team, noticed that despite testing new features on every platform - MacOS, Linux, and Windows at the end of each month before a release, the screen reader experience had been neglected. Inspired by JooYoung's observations, Megan advocated for a new protocol: after a feature is released, it will be tested with screen readers in the next iteration. Additionally, she initiated retroactive tests on features to rectify this historical oversight.
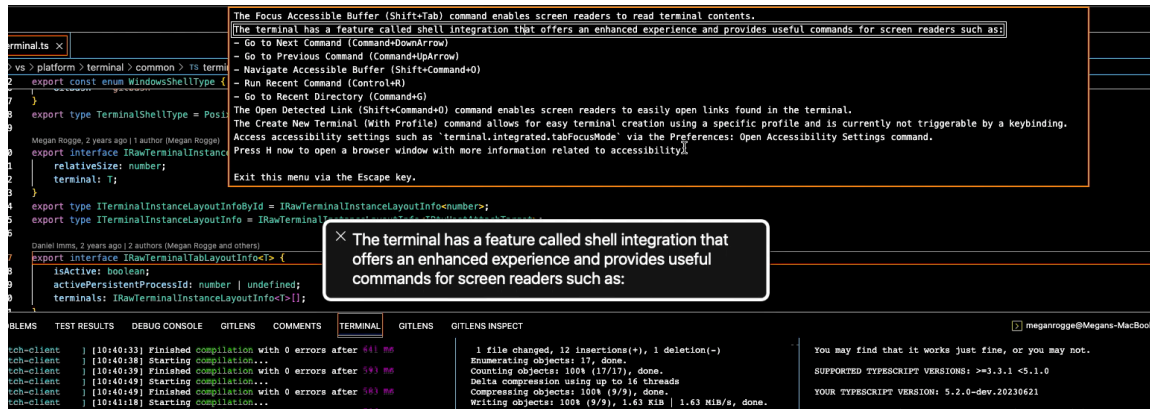
Fig. 2. The terminal help menu

5 DISCUSSION AND CONCLUSION

In our collaborative journey, blending the expertise of both sighted and blind developers, we've unearthed pivotal insights about open-source accessibility. Our endeavors with Visual Studio Code serve as a case in point. Megan and JooYoung's livestream seminar accentuated the profound impact of merging accessibility considerations with open-source development.

Open-source platforms are foundational in the tech realm. Our reach and influence cascade into multiple offshoot applications. Hence, embedding accessibility in these parent projects can have a magnified effect, promoting inclusivity across numerous derivative platforms. The participatory nature of open-source projects, welcoming feedback from a diverse array of users, is both a strength and a challenge. JooYoung's collaboration, while external to Microsoft, highlights this open engagement. Yet, a pertinent concern is the potential marginalization of voices unfamiliar with platforms like GitHub. For open-source teams, this underscores the necessity to proactively engage with and seek feedback from these underrepresented communities. Such engagement is not just about hearing, but understanding and integrating feedback. The synergy between Megan and JooYoung exemplifies the potential outcomes when such engagements are cultivated. A recurring theme from our experience is the importance of proactive accessibility considerations. Post-design modifications often present challenges, emphasizing the need for early integration of accessibility measures. Drawing these threads together, our co-design experience underscores the imperative of fostering an inclusive ethos in open-source development. By ensuring a platform that is receptive to diverse voices, we can move closer to a universally accessible coding ecosystem.

ACKNOWLEDGMENTS

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, Yury 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. 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

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.

10

```
$ 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, ac
issue    microsoft/vscode    186754  open     data science audio and text graph for visually impaired person   feature
issue    microsoft/vscode    186679  open     Alert that the help hint has been disabled  bug, accessibility   2023-06
issue    microsoft/vscode    186678  open     Change accessible buffer command navigation keybinding for screen reade
issue    microsoft/vscode    186676  open     add accessible view provider for inline chat response    feature-request
issue    microsoft/vscode    186675  open     when next/previous ghost text suggestion is shown, we don't alert scree
issue    microsoft/vscode    186673  closed   have accessible view for ghost text completions feature-request, access
issue    microsoft/vscode    186659  closed   Sticky scroll for screen reader users    feature-request, accessibility
issue    microsoft/vscode    186514  closed   Closing accessibility hint doesn't stop VoiceOver from reading it    bug
issue    microsoft/vscode    185705  closed   Consider providing screen reader with the chat response for inline chat
issue    microsoft/vscode    185691  closed   Consider which audio cues for chat experience should be enabled by defa
issue    microsoft/vscode    185565  open     Accessibility: Cannot turn off audio cues on a language level    feature
issue    microsoft/vscode    185371  open     Review usage of `aria-live: assertive`, `alert` throughout the code bas
issue    microsoft/vscode    185155  open     Alert screen reader users that something has occurred when `clear` is u
pr  microsoft/vscode    185153  merged  prevent screen reader from reading a user's chat request on enter        202
issue    microsoft/vscode    184357  open     [Accessibility]: Make syntax highlight accessible to screen reader user
issue    microsoft/vscode    184176  closed   Add notebook accessibility help menu     feature-request, verified, acce
issue    microsoft/vscode    184173  closed   Accessibility: Take out extra messages from Notebook verbosity FALSE
issue    microsoft/vscode    183567  open     Explore improvements to notifications when using a screen reader     ac
issue    microsoft/vscode    183363  closed   Make accessibility help generic feature-request, accessibility, on-test
issue    microsoft/vscode    183030  closed   Reading suggestions or autocomplete of extensions    bug, verified, acce
issue    microsoft/vscode    182682  open     Merge editor accessibility    accessibility, merge-editor 2023-06-21T12:
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, access
issue    microsoft/vscode    181139  open     Accessibility: Make Tab key focus restricted to the currently open view
issue    microsoft/vscode    181060  closed   BAccessibility: Pressing Shift+Tab key in Ctrl+F moves to terminal area
issue    microsoft/vscode    180970  closed   provide alt text for image outputs  feature-request, accessibility, ver
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 2023-05-23T
issue    microsoft/vscode    180725  open     interacting with components should be consistent     accessibility, exte
issue    microsoft/vscode    180653  closed   [Accessibility]: Present content first in References Treeview    help wa
issue    microsoft/vscode    180221  open     Accessibility: Ctrl+Down/UpArrows does not work in Tree find control
issue    microsoft/vscode    180216  closed   [Accessibility]: Typing characters does not move focus in File Explorer
issue    microsoft/vscode    180176  open     [Accessibility]: Consider replacing audioCues.lineHasInlineSuggestion v
issue    microsoft/vscode    180083  open     Accessibility: Make Debug Console follow terminal tabFocusMode    accessi
issue    microsoft/vscode    180049  open     [Accessibility]: Support filtering symbol types in Document Sylbol View
issue    microsoft/vscode    179981  open     Focus does not stay in the editor area after sending selection to termi
issue    microsoft/vscode    179979  closed   Accessibility: Ctrl+Up/DownArrows does not work in terminal created in
```

11

```
issue   microsoft/vscode    179970  closed  Accessibility: Make Ctrl+RightArrow expand all in tree views
issue   microsoft/vscode    179969  closed  Make filtering more flexible in Problem View    info-needed, en
issue   microsoft/vscode    179967  open    Accessibility: Generalize Ctrl+DownArrow and Ctrl+UpArrow to al
issue   microsoft/vscode    179964  open    Accessibility: Improve Problem View search input    accessibili
issue   microsoft/vscode    179718  closed  search result aria label should prioritze content over location
issue   microsoft/vscode    179717  closed  Problems aria label should prioritize content over location fea
issue   microsoft/vscode    179716  open    Allow configuring what is included in the accessible buffer fea
issue   microsoft/vscode    179283  closed  [Accessibility]: Make "Go to line" announce focused line after
issue   microsoft/vscode    179272  closed  [Accessibility]: Add shortcut keys to jump between executed com
issue   microsoft/vscode    179123  closed  [Accessibility]: `Search: Find in Files, Control+Shift+F` could
issue   microsoft/vscode    178935  closed  [Accessibility] Audio cues stopped working in Chrome    accessi
issue   microsoft/vscode    178915  closed  [Accessibility] Make sticky scroll view line indentation access
issue   microsoft/vscode    177755  closed  [Accessibility]: Switching editor (Ctrl+Tab) does not work from
issue   microsoft/vscode    177697  closed  `Terminal: navigate accessible buffer` does not work sometimes
issue   microsoft/vscode    177696  closed  Inline suggestion is read twice by the screen reader    bug, ve
issue   microsoft/vscode    177694  closed  add command to repeat most recent notification  accessibility,
issue   microsoft/vscode    177029  closed  [Accessibility]: `Set Selection Anchor` and `Select from Anchor
issue   microsoft/vscode    176779  open    Make error in line audio cue configurable    feature-request, ac
issue   microsoft/vscode    176521  open    [Accessibility] Support task completion/failure audio cues in C
issue   microsoft/vscode    176293  closed  Prefer SVG renderers for image output to improve screen reader
issue   microsoft/vscode    176292  open    improve screen reader context and navigation of Cell outputs
issue   microsoft/vscode    176290  open    consider default keybindings for go to next / previous cell inp
issue   microsoft/vscode    176286  closed  `allowNavigateToSurroundingCells` should be false when screen r
issue   microsoft/vscode    176242  open    Notify screen reader users that a VS Code update is available
issue   microsoft/vscode    175986  open    Allow VS Code extensions to trigger audio cues  feature-request
pr  microsoft/vscode    175823  merged  provide screen reader with inline suggestions       2023-04-21T23:2
issue   microsoft/vscode    175743  open    Output of Jupyter notebook cells is not intuitively accessible
issue   microsoft/vscode    175432  closed  [Accessibility] Pressing Ctrl+M key (toggle tabFocusMode) shoul
issue   microsoft/vscode    175348  open    Refine error on line audio cue  feature-request, accessibility
issue   microsoft/vscode    175341  closed  [Accessibility] Some thoughts on error in line audio cue    bug
issue   microsoft/vscode    175282  open    [Accessibility] Do not use title attribute when labeling button
issue   microsoft/vscode    175177  closed  [Accessibility] Remove repeated word from the terminal help bug
issue   microsoft/vscode    175175  closed  [Accessibility] "Go to Recent Directory (Control+G)" instructio
issue   microsoft/vscode    175162  closed  assign different default keybinding for focusing the accessible
issue   microsoft/vscode    175140  closed  Add a command that accepts a notification's default action  fea
issue   microsoft/vscode    175111  closed  [Accessibility] Redundant read-only terminal buffer needs to be
issue   microsoft/vscode    175105  closed  [Accessibility] Reconsider the UI design for {"editor.screenRea
issue   microsoft/vscode    175014  closed  Replace diff line modified/deleted/ inserted audio cues with pu
issue   microsoft/vscode    175013  closed  On focus of the accessible buffer, if the last command failed,
issue   microsoft/vscode    175012  closed  Use more succinct audio cue when terminal command fails feature
```

```
issue    microsoft/vscode     175011   closed   position the cursor at the end of the accessible buffer by default   bug
issue    microsoft/vscode     174857   closed   [Accessibility] Line-by-line audio cues are not played when column posi
issue    microsoft/vscode     174800   closed   [Accessibility] Shift+Tab is always forced to go to a11y terminal buffe
issue    microsoft/vscode     174798   closed   [Accessibility] Remove redundant 4-5 blank lines from the terminal a11y
issue    microsoft/vscode     174797   closed   [Accessibility] python repl content is not parsable in the a11y termina
issue    microsoft/vscode     174793   closed   [Accessibility] Consider adding a setting to preserve focus in a11y ter
pr  microsoft/vscode    174606   merged   add setting for aria-live assertive alert for ghost text        2023-04-07T
issue    microsoft/vscode     174368   closed   Play audio cue when a command exits with non-zero code  feature-request
issue    microsoft/vscode     174367   closed   Mention `Terminal: Create Terminal with Profile` in terminal a11y help
issue    microsoft/vscode     174365   closed   Suggest screen reader users migrate from `cmd prompt` -> `pwsh` feature
issue    microsoft/vscode     174362   open     Next suggestion isn't read  bug, accessibility  2023-03-13T09:34:44Z
issue    microsoft/vscode     174360   open     When in `tabFocusMode`, assign a different keybinding for inline sugges
issue    microsoft/vscode     174359   open     Add more audio cues feature-request, accessibility  2023-03-02T16:46:57
issue    microsoft/vscode     174079   closed   Add symbol provider for terminal accessible buffer   feature-request, o
issue    microsoft/vscode     173622   closed   No indication of ghost text and actions via screen reader    accessibili
issue    microsoft/vscode     173532   closed   I can no longer access terminal accessibility buffer with orca  bug, ve
issue    microsoft/vscode     173452   closed   [Accessibility] Add page up/down support for accessible buffer   feature
issue    microsoft/vscode     173451   closed   [Accessibility] Make `editor.action.toggleTabFocusMode` configurable i
issue    microsoft/vscode     172606   closed   [Accessibility] Go to next/previous change commands don't provide delet
issue    microsoft/vscode     172582   closed   [Accessibility] Terminal a11y buffer is not automatically updated    bug
issue    microsoft/vscode     172525   closed   [Accessibility] Error audio cues are not played on a character level.
issue    microsoft/vscode     172523   closed   [Accessibility]: Audio Cues are notplayed a against swift arrow navigat
issue    microsoft/vscode     172465   closed   Reduce noise for screen reader users     feature-request, verified, acce
issue    microsoft/vscode     172458   closed   in diff view, line selection shouldn't happen on cursor move      bug, ve
issue    microsoft/vscode     172399   closed   tab has to be pressed twice to go back to the terminal buffer from acce
pr  microsoft/vscode    172276   merged   xterm@5.2.0-beta.21     2023-03-11T23:23:27Z
issue    microsoft/vscode     172204   closed   Screen reader accessibility mode reads terminal contents character by c
issue    microsoft/vscode     172149   open     [Accessibility] Command history is not readable in terminal input field
issue    microsoft/vscode     172024   closed   [Accessibility] Ctrl+M (editor.action.toggleTabFocusMode) does not work
issue    microsoft/vscode     172007   closed   Terminal accessibility buffer does not read output upon enter    bug, ve
issue    microsoft/vscode     172006   closed   Make terminal accessibility buffer read only     upstream, accessibility
issue    microsoft/vscode     171918   closed   [Accessibility] Support Home and End keys in Open Detected Link view
issue    microsoft/vscode     171916   closed   [Accessibility] Ctrl+Shift+O does not close Open Detected Link view *as
issue    microsoft/vscode     171914   closed   [Accessibility] Make terminal a11y buffer even more accessible   accessi
issue    microsoft/vscode     171755   open     Code lens is not accessible via screen reader    accessibility, under-di
issue    microsoft/vscode     171544   closed   sometimes audio cues don't play when going to next/previous difference
issue    microsoft/vscode     171429   closed   [Accessibility] Diff editor cursor position is not preserved after esca
issue    microsoft/vscode     171426   open     [Accessibility] Diff editor cursor position is not preserved after esca
issue    microsoft/vscode     171256   closed   [Accessibility] Trigger diff audio cues against standard arrow key navi
issue    microsoft/vscode     171253   open     [Accessibility] Allow users to customize the audio cue play priority
```

```
issue   microsoft/vscode    171200   closed   support screen reader reading the line and audio cues when go t
issue   microsoft/vscode    171199   open     Accessibility getting started experience    feature-request, ac
pr microsoft/vscode    170985   merged   support screen reader reading the line when go to next/previous dif
issue   microsoft/vscode    170971   closed   [Accessibility]: Allow users to replace default sound file  *du
issue   microsoft/vscode    169853   closed   Explore plain content editable element for terminal buffer inst
issue   microsoft/vscode    168746   open     [Accessibility] Word wrap does not work in diff view (F7 and Sh
pr microsoft/vscode    167349   closed   fix #167348: add aria-live      2023-02-16T17:31:02Z
issue   microsoft/vscode    167348   closed   [Accessibility] div.monaco-tokenized-source requires aria-live=
issue   microsoft/vscode    168814   open     Need a clearer landmark and label for notebook output area  fea
issue   microsoft/vscode    166518   closed   Add audio cues for Go to Next/ Previous Change commands feature
issue   microsoft/vscode    166472   open     [Accessibility] Add an option to allow Alt+F5 to jump to the ne
issue   microsoft/vscode    165863   closed   Hitting spacebar does not replay currently focused audio cue
issue   microsoft/vscode    165357   closed   [Accessibility] Audio Cues still doesn't work in github.dev ins
issue   microsoft/vscode    165161   open     [Accessibility] Open Folder dialog controls do not have acceler
issue   microsoft/vscode    164988   closed   [Accessibility]: Screen readers do not read currently focused l
issue   microsoft/vscode    163506   open     [Accessibility] Provide icon info to screen readers feature-req
issue   microsoft/vscode    160301   open     [Accessibility] Some long file content line is not correctly co
issue   microsoft/vscode    159029   open     Merge editor accessibility improvements accessibility, merge-ed
issue   microsoft/vscode    155919   closed   [Accessibility] Support `Live Share` audio cues in `Help: List
issue   microsoft/vscode    155655   closed   [Accessibility] For easier code navigation, add jump to next/pr
issue   microsoft/vscode    154027   closed   [Accessibility]: Terminal output is not read in real time on Ma
issue   microsoft/vscode    147607   closed   [Accessibility] Unlabelled `codicon` buttons    info-needed, ac
issue   microsoft/vscode    147386   closed   [Accessibility] Add audio cues for indentation levels   feature
issue   microsoft/vscode    147230   open     Play audio-cues for auto-suggestions    feature-request, access
issue   microsoft/vscode    147226   closed   [Accessibility] Consider adding audio cues for diffs (added / d
issue   microsoft/vscode    147190   closed   [Accessibility] Audio Cues doesn't work in web editor   bug, ve
issue   microsoft/vscode    143185   closed   Problems to access the preview of a markdown file using orca
issue   microsoft/vscode    142983   closed   [Terminal accessibility] JAWS does not speak anything against a
issue   microsoft/vscode    141529   closed   Webviews displaying results of an API call with Restclient exte
issue   microsoft/vscode    135920   closed   [Accessibility] "xterm-accessibility" class div does not have t
issue   microsoft/vscode    135035   closed   [Accessibility] GitHub Web Editor: Cannot configure accessibili
issue   microsoft/vscode    133876   closed   [Accessibility] Assign a keyboard shortcut key to Focus Termina
issue   microsoft/vscode    133805   closed   `Shift+Alt+R` for `Reveal in File Explorer` doesn't work when f
issue   microsoft/vscode    133773   closed   [Accessibility] "document" role is needed for "monaco-hover" cl
issue   microsoft/vscode    132275   closed   [Accessibility] Add "document" role to webview widget   verifie
issue   microsoft/vscode    131295   open     [Accessibility] Character is not read properly in terminal inpu
issue   microsoft/vscode    131090   closed   [Accessibility] NVDA and JAWS do not read focused auto-suggesti
issue   microsoft/vscode    130565   closed   Notify Screenreader Users When Inline Suggestions Or Decoration
issue   microsoft/vscode    121735   closed   Terminal input does not work with NVDA  bug, important, accessi
issue   microsoft/vscode    113482   closed   Tab code-completion does not work in terminal input against scr
```

```
issue    microsoft/vscode    111255   open     VS Code native notebook accessibility improvement    debt, accessibility
issue    microsoft/vscode    105425   closed   Garbage characters are inserted if you come back from terminal output t
issue    microsoft/vscode    103095   closed   Auto-complete popup puts redundant "item" prefix per suggested code for
issue    microsoft/vscode    98918    closed   Terminal output div container should be more accessible for screen read
issue    microsoft/vscode    95570    closed   Support terminal link keyboard navigation    feature-request, accessibil
issue    microsoft/vscode    90408    open     Feature request: Accessibility support for Jupyter notebooks in VSCode
```