

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

In this paper, we will showcase a few examples of how blind and sighted developers have been working together to make Visual Studio Code more accessible. Through our collaboration, which is evident in the GitHub issues, pull requests, review process, and insider's releases we've created together over the past few months, we hope to help other open source developers use these methods to create more accessible development environments. Accessible version of this article can be found at <https://bit.ly/vscodea11y>

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

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, 12 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. However, blind developers¹ are not able to take advantage of the many features that graphical user interface (GUI)-based IDEs offer. 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 and compile terminals. 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

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

the non-visual workaround of accessing an IDE with a screen reader. Although there is a community of blind programmers called Program-L that helps each other with their struggles, 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, 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 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

2.1 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).

Accessibility is and has been a core priority for VS Code since its inception. Without a screen reader user on the team, VS Code relies heavily upon the tremendous contributions of community members for insight and direction.

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 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 screenreader 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 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.

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

2.2 Biographies of the Authors

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, including 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, a community of blind programmers where blind programmers help and support each other. 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 VS Code software engineer. She has worked on the product since graduating from the University of North Carolina at Chapel Hill in 2020 with 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.

2.3 Collaboration Methods

JooYoung and Megan communicated issues via GitHub asynchronously for a few weeks before the two agreed that regular meetings might be a more efficient and productive approach. JooYoung's ideas and insights paired well with Megan's willingness to learn and drive to improve the product's accessibility.

During each meeting, JooYoung shared his screen and asked questions or offered suggestions, while Megan offered insight, asked questions, and took notes about bugs or features to be addressed.

These meetings allowed Megan to learn how JooYoung uses VS Code and helped him understand components of the product that might otherwise have been confusing or not discoverable.

The asynchronous communication on Github and via email continued; Megan would often send a follow up email summarizing the findings of their meetings before sharing it with the whole team and JooYoung would comment on the resultant issues if she missed anything or when testing the fixes.

3 CASE STUDIES

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

3.1 Terminal Buffer

As mentioned in Section 2.1, xterm.js, which is used as the terminal UI in VSCode, has a screen-reader accessibility mode that allows blind people to access the output of the terminal. However, there was a gap between simply being able to access information (accessibility) and being able to use it conveniently (usability), and this had led to ongoing complaints about the VSCode terminal among blind programmers, especially in Program-L. Here's how it works. Suppose you type and execute the command `echo hello; echo world`; in the terminal, you will see two lines of output on the terminal screen: `hello` and `world`. The existing accessibility mode of xterm.js outputted this content in screen reader speech using aria-live alert, and provided the ability to review the terminal output history line by line with the `Ctrl+UpArrow` and `Ctrl+DownArrow` keys. If the output is short and simple, this approach is fine. However, if the output is long, with complex error messages or computational results, a quick flash message through text-to-speech is not going to capture that much information in the human working memory. Another problem is that the `Ctrl+Up/DownArrow` navigation keys, which review the terminal history line by line, bundle the entire contents of the currently focused line into a single object and pass it to the screen reader, making it difficult to examine the terminal contents character by character or word by word. The blind user had to change the reading mode with the screen reader's virtual cursor (AKA, browse mode in NVDA; QuickNav mode in VoiceOver) to review the terminal content on a character-by-character and word-by-word basis. After the review, they (and we) had to turn off the screen reader's virtual cursor and revert to forms mode (focus mode in NVDA; QuickNav off in VoiceOver) to resume terminal input, causing considerable inconvenience and hassle.

JooYoung has started a conversation on the official Microsoft VSCode GitHub page to raise awareness of these issues and suggest new solutions (microsoft/vscode#98918: Terminal output div container should be more accessible for screen readers).

Megan worked on terminal shell integration, a feature which enables VS Code to understand what's going on in the terminal. This allows a user to navigate easily between commands, copy command output, and more. JooYoung shared his screen, tested the feature, and pointed out that the actual terminal buffer was still inaccessible for screen reader users - since it could not be navigated using arrow keys. He suggested that the output view provides a more accessible experience, so Megan discussed this with a colleague and determined that the same underlying component should be used for the terminal. This allowed the formerly "black box" of the terminal buffer to be navigable via arrow keys for screen reader users. JooYoung suggested that command navigation provided by shell integration could also be used in this accessible buffer and Megan implemented that as well.

More specifically, he proposed replacing the terminal output with a text editor buffer that allowed for standard arrow-key navigation. This process took over a year and a lot of technical trial and error and collaborative testing. For example, we started with a remarkup to redirect the terminal output web container, which was initially designated as "list", to the aria "document" or "textbox" landmarks, but

were unable to achieve satisfactory results due to varying levels of screen reader and platform support for aria. Next, we remodeled the terminal output into a text area with “contenteditable” and “readonly” attributes, but this was not very compatible with the screen reader’s speech buffer. Finally, we created a separate accessible terminal buffer by passing the terminal output to VSCode’s native Monaco editor, which is guaranteed accessible, leading to a successful feature that ensures satisfactory accessibility and usability on any operating system or screen reader. This terminal buffer feature was officially released in VSCode stable version xx and has been very well received by many users in the Program-L community.

For example, with VoiceOver on, the terminal buffer is entered for a task terminal. A line with an error is focused and read by VoiceOver (?@fig-buffer).

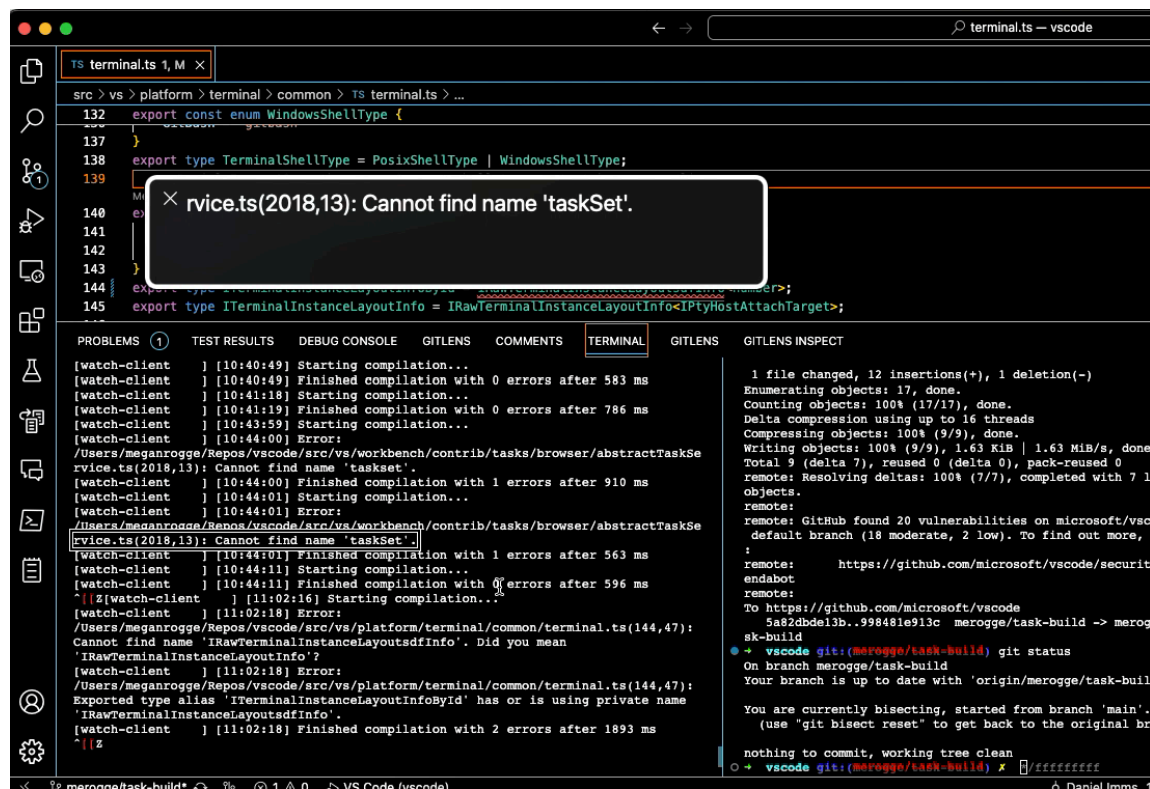


Fig. 1. VoiceOver reads “rvice.ts(2018,13: Cannot find name ‘taskSet’”

3.2 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 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](#): [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. 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 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.

3.3 Verbosity Settings, 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 (`?@fig-help`). A screen reader user can use arrow keys to read the content line by line, character by character.

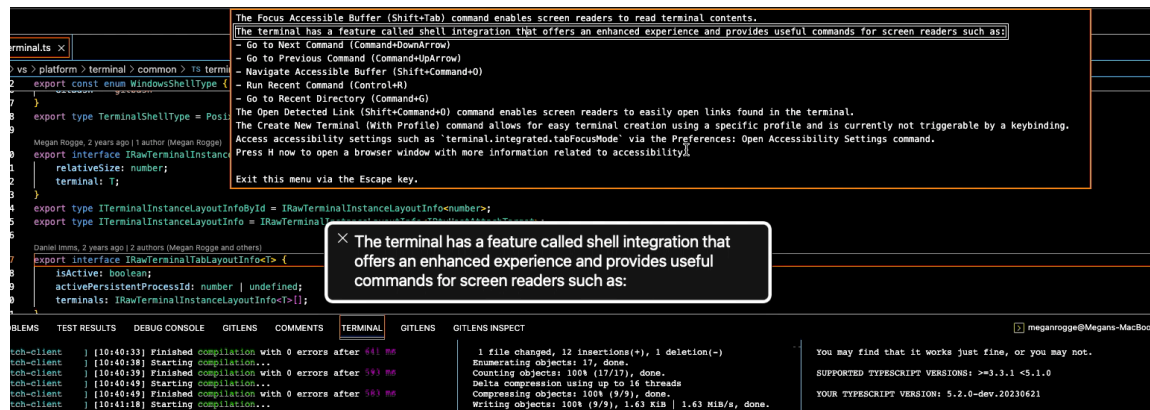


Fig. 2. A line in the help menu is focused and VoiceOver reads “The terminal has a feature called shell integration that offers an enhanced experience and provides useful commands for screen readers such as:”

3.4 Copilot and Inline Suggestions

- Recognizing the power of AI, JooYoung was excited to try out Copilot. He was disappointed to find that the inline suggestions were not accessible. He created an issue and Megan fixed it.

3.5 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.

4 DISCUSSION AND CONCLUSION

- Reflection on what we have learned from this collaboration.

Screen reader users have unique insight into accessibility issues and provide creative solutions to technical problems. Collaborating with screen reader users improves understanding of accessibility issues (for the sighted developer) and features (for the screen reader user), increases the drive to address issues, and dramatically improves the accessibility of a product. Increased accessibility results in a better product for all. For example, while the terminal accessible buffer was designed for screen reader users, others have suggested they benefit from the feature.

- What we recommend for other open-source communities for better accessibility.

It is important to share findings so that other developers can learn and apply the knowledge to their applications. Megan and JooYoung participated in an accessibility presentation via livestream which reached 159,769 people within just the first two weeks. Additionally, JooYoung will be leading the team in a deep dive about his experience using VS Code. Megan believes getting a glimpse of JooYoung's experience will help the team understand the importance of and challenges related to accessibility.

Considering accessibility early on in a feature's implementation is critical so that screen reader users can experience the latest technology without delay. Working closely with JooYoung, Megan has witnessed that screen reader users can be totally blocked if this is not considered or addressed properly. As such, accessibility fixes should be prioritized over most others.

5 ACKNOWLEDGMENTS

ACKNOWLEDGMENTS

We thank ... - Isidor Nikolic - Kai Maetzel - Daniel Imms - Raymond Zhao - Roberto Perez - José Vilmar Estácio de Souza - VS Code team - Amnon Freidlin

REFERENCES

A APPENDIX: VSCODEACCESSIBILITY 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...`
issue	microsoft/vscode	186754	open	data science audio and text graph for visually impaired person
issue	microsoft/vscode	186679	open	Alert that the help hint has been disabled bug, accessibility
issue	microsoft/vscode	186678	open	Change accessible buffer command navigation keybinding for scre

417	issue	microsoft/vscode	186676	open	add accessible view provider for inline chat response	feature-request
418	issue	microsoft/vscode	186675	open	when next/previous ghost text suggestion is shown, we don't alert screen reader	
419	issue	microsoft/vscode	186673	closed	have accessible view for ghost text completions	feature-request, accessibility
420	issue	microsoft/vscode	186659	closed	Sticky scroll for screen reader users	feature-request, accessibility
422	issue	microsoft/vscode	186514	closed	Closing accessibility hint doesn't stop VoiceOver from reading it	bug
423	issue	microsoft/vscode	185705	closed	Consider providing screen reader with the chat response for inline chat	
424	issue	microsoft/vscode	185691	closed	Consider which audio cues for chat experience should be enabled by default	
425	issue	microsoft/vscode	185565	open	Accessibility: Cannot turn off audio cues on a language level	feature-request
427	issue	microsoft/vscode	185371	open	Review usage of `aria-live: assertive`, `alert` throughout the code base	
428	issue	microsoft/vscode	185155	open	Alert screen reader users that something has occurred when `clear` is used	
429	pr	microsoft/vscode	185153	merged	prevent screen reader from reading a user's chat request on enter	2023-06-21T12:12:12Z
430	issue	microsoft/vscode	184357	open	[Accessibility]: Make syntax highlight accessible to screen reader users	
431	issue	microsoft/vscode	184176	closed	Add notebook accessibility help menu	feature-request, verified, accessibility
432	issue	microsoft/vscode	184173	closed	Accessibility: Take out extra messages from Notebook verbosity	FALSE
433	issue	microsoft/vscode	183567	open	Explore improvements to notifications when using a screen reader	accessibility
434	issue	microsoft/vscode	183363	closed	Make accessibility help generic	feature-request, accessibility, on-test
435	issue	microsoft/vscode	183030	closed	Reading suggestions or autocomplete of extensions	bug, verified, accessibility
436	issue	microsoft/vscode	182682	open	Merge editor accessibility	accessibility, merge-editor 2023-06-21T12:12:12Z
437	pr	microsoft/vscode	182666	merged	outweigh normal editor accessibility help menu	2023-06-30T23:23:35Z
438	issue	microsoft/vscode	181732	closed	Accessibility: Make drag-and-drop accessible via keyboard	bug, accessibility
439	issue	microsoft/vscode	181139	open	Accessibility: Make Tab key focus restricted to the currently open view	
440	issue	microsoft/vscode	181060	closed	BAccessibility: Pressing Shift+Tab key in Ctrl+F moves to terminal area	
441	issue	microsoft/vscode	180970	closed	provide alt text for image outputs	feature-request, accessibility, verified
442	pr	microsoft/vscode	180776	merged	fix windows quick fixes	2023-06-09T23:22:27Z
443	issue	microsoft/vscode	180729	closed	Investigate merge editor accessibility	bug, accessibility 2023-05-23T12:12:12Z
444	issue	microsoft/vscode	180725	open	interacting with components should be consistent	accessibility, extended
445	issue	microsoft/vscode	180653	closed	[Accessibility]: Present content first in References Treeview	help wanted
446	issue	microsoft/vscode	180221	open	Accessibility: Ctrl+Down/UpArrows does not work in Tree find control	
447	issue	microsoft/vscode	180216	closed	[Accessibility]: Typing characters does not move focus in File Explorer	
448	issue	microsoft/vscode	180176	open	[Accessibility]: Consider replacing audioCues.lineHasInlineSuggestion with	
449	issue	microsoft/vscode	180083	open	Accessibility: Make Debug Console follow terminal tabFocusMode	accessibility
450	issue	microsoft/vscode	180049	open	[Accessibility]: Support filtering symbol types in Document Symbol View	
451	issue	microsoft/vscode	179981	open	Focus does not stay in the editor area after sending selection to terminal	
452	issue	microsoft/vscode	179979	closed	Accessibility: Ctrl+Up/DownArrows does not work in terminal created in	
453	issue	microsoft/vscode	179970	closed	Accessibility: Make Ctrl+RightArrow expand all in tree views	accessibility
454	issue	microsoft/vscode	179969	closed	Make filtering more flexible in Problem View	info-needed, error-list
455	issue	microsoft/vscode	179967	open	Accessibility: Generalize Ctrl+DownArrow and Ctrl+UpArrow to all input	
456	issue	microsoft/vscode	179964	open	Accessibility: Improve Problem View search input	accessibility, error-list
457	issue	microsoft/vscode	179718	closed	search result aria label should prioritize content over location	feature-request
458	issue	microsoft/vscode	179717	closed	Problems aria label should prioritize content over location	feature-request

469	issue	microsoft/vscode	179716	open	Allow configuring what is included in the accessible buffer fea
470	issue	microsoft/vscode	179283	closed	[Accessibility]: Make "Go to line" announce focused line after
471	issue	microsoft/vscode	179272	closed	[Accessibility]: Add shortcut keys to jump between executed com
472	issue	microsoft/vscode	179123	closed	[Accessibility]: `Search: Find in Files, Control+Shift+F` could
473	issue	microsoft/vscode	178935	closed	[Accessibility] Audio cues stopped working in Chrome accessi
474	issue	microsoft/vscode	178915	closed	[Accessibility] Make sticky scroll view line indentation access
475	issue	microsoft/vscode	177755	closed	[Accessibility]: Switching editor (Ctrl+Tab) does not work from
476	issue	microsoft/vscode	177697	closed	`Terminal: navigate accessible buffer` does not work sometimes
477	issue	microsoft/vscode	177696	closed	Inline suggestion is read twice by the screen reader bug, ve
478	issue	microsoft/vscode	177694	closed	add command to repeat most recent notification accessibility,
479	issue	microsoft/vscode	177029	closed	[Accessibility]: `Set Selection Anchor` and `Select from Anchor
480	issue	microsoft/vscode	176779	open	Make error in line audio cue configurable feature-request, ac
481	issue	microsoft/vscode	176521	open	[Accessibility] Support task completion/failure audio cues in C
482	issue	microsoft/vscode	176293	closed	Prefer SVG renderers for image output to improve screen reader
483	issue	microsoft/vscode	176292	open	improve screen reader context and navigation of Cell outputs
484	issue	microsoft/vscode	176290	open	consider default keybindings for go to next / previous cell inp
485	issue	microsoft/vscode	176286	closed	`allowNavigateToSurroundingCells` should be false when screen r
486	issue	microsoft/vscode	176242	open	Notify screen reader users that a VS Code update is available
487	issue	microsoft/vscode	175986	open	Allow VS Code extensions to trigger audio cues feature-request
488	pr	microsoft/vscode	175823	merged	provide screen reader with inline suggestions 2023-04-21T23:2
489	issue	microsoft/vscode	175743	open	Output of Jupyter notebook cells is not intuitively accessible
490	issue	microsoft/vscode	175432	closed	[Accessibility] Pressing Ctrl+M key (toggle tabFocusMode) shoul
491	issue	microsoft/vscode	175348	open	Refine error on line audio cue feature-request, accessibility
492	issue	microsoft/vscode	175341	closed	[Accessibility] Some thoughts on error in line audio cue bug
493	issue	microsoft/vscode	175282	open	[Accessibility] Do not use title attribute when labeling button
494	issue	microsoft/vscode	175177	closed	[Accessibility] Remove repeated word from the terminal help bug
495	issue	microsoft/vscode	175175	closed	[Accessibility] "Go to Recent Directory (Control+G)" instructio
496	issue	microsoft/vscode	175162	closed	assign different default keybinding for focusing the accessible
497	issue	microsoft/vscode	175140	closed	Add a command that accepts a notification's default action fea
498	issue	microsoft/vscode	175111	closed	[Accessibility] Redundant read-only terminal buffer needs to be
499	issue	microsoft/vscode	175105	closed	[Accessibility] Reconsider the UI design for {"editor.screenRea
500	issue	microsoft/vscode	175014	closed	Replace diff line modified/deleted/ inserted audio cues with pu
501	issue	microsoft/vscode	175013	closed	On focus of the accessible buffer, if the last command failed,
502	issue	microsoft/vscode	175012	closed	Use more succinct audio cue when terminal command fails feature
503	issue	microsoft/vscode	175011	closed	position the cursor at the end of the accessible buffer by defa
504	issue	microsoft/vscode	174857	closed	[Accessibility] Line-by-line audio cues are not played when col
505	issue	microsoft/vscode	174800	closed	[Accessibility] Shift+Tab is always forced to go to all y termin
506	issue	microsoft/vscode	174798	closed	[Accessibility] Remove redundant 4-5 blank lines from the termi
507	issue	microsoft/vscode	174797	closed	[Accessibility] python repl content is not parsable in the all y
508	issue	microsoft/vscode	174793	closed	[Accessibility] Consider adding a setting to preserve focus in

519

520

521	pr	microsoft/vscode	174606	merged	add	setting for aria-live assertive alert for ghost text	2023-04-07
522	issue	microsoft/vscode	174368	closed		Play audio cue when a command exits with non-zero code	feature-request
523	issue	microsoft/vscode	174367	closed		Mention `Terminal: Create Terminal with Profile` in terminal a11y help	
524	issue	microsoft/vscode	174365	closed		Suggest screen reader users migrate from `cmd prompt` -> `pwsh` feature	
525	issue	microsoft/vscode	174362	open		Next suggestion isn't read	bug, accessibility 2023-03-13T09:34:44Z
526	issue	microsoft/vscode	174360	open		When in `tabFocusMode`, assign a different keybinding for inline sugges	
527	issue	microsoft/vscode	174359	open		Add more audio cues	feature-request, accessibility 2023-03-02T16:46:51Z
528	issue	microsoft/vscode	174079	closed		Add symbol provider for terminal accessible buffer	feature-request, or
529	issue	microsoft/vscode	173622	closed		No indication of ghost text and actions via screen reader	accessibili
530	issue	microsoft/vscode	173532	closed		I can no longer access terminal accessibility buffer with orca	bug, ve
531	issue	microsoft/vscode	173452	closed		[Accessibility] Add page up/down support for accessible buffer	feature
532	issue	microsoft/vscode	173451	closed		[Accessibility] Make `editor.action.toggleTabFocusMode` configurable in	
533	issue	microsoft/vscode	172606	closed		[Accessibility] Go to next/previous change commands don't provide delet	
534	issue	microsoft/vscode	172582	closed		[Accessibility] Terminal a11y buffer is not automatically updated	bug
535	issue	microsoft/vscode	172525	closed		[Accessibility] Error audio cues are not played on a character level.	
536	issue	microsoft/vscode	172523	closed		[Accessibility]: Audio Cues are notplayed a against swift arrow navigat	
537	issue	microsoft/vscode	172465	closed		Reduce noise for screen reader users	feature-request, verified, acco
538	issue	microsoft/vscode	172458	closed		in diff view, line selection shouldn't happen on cursor move	bug, ve
539	issue	microsoft/vscode	172399	closed		tab has to be pressed twice to go back to the terminal buffer from acco	
540	pr	microsoft/vscode	172276	merged		xterm@5.2.0-beta.21	2023-03-11T23:23:27Z
541	issue	microsoft/vscode	172204	closed		Screen reader accessibility mode reads terminal contents character by c	
542	issue	microsoft/vscode	172149	open		[Accessibility] Command history is not readable in terminal input field	
543	issue	microsoft/vscode	172024	closed		[Accessibility] Ctrl+M (editor.action.toggleTabFocusMode) does not work	
544	issue	microsoft/vscode	172007	closed		Terminal accessibility buffer does not read output upon enter	bug, ve
545	issue	microsoft/vscode	172006	closed		Make terminal accessibility buffer read only	upstream, accessibility
546	issue	microsoft/vscode	171918	closed		[Accessibility] Support Home and End keys in Open Detected Link view	
547	issue	microsoft/vscode	171916	closed		[Accessibility] Ctrl+Shift+O does not close Open Detected Link view *as	
548	issue	microsoft/vscode	171914	closed		[Accessibility] Make terminal a11y buffer even more accessible	accessi
549	issue	microsoft/vscode	171755	open		Code lens is not accessible via screen reader	accessibility, under-d
550	issue	microsoft/vscode	171544	closed		sometimes audio cues don't play when going to next/previous difference	
551	issue	microsoft/vscode	171429	closed		[Accessibility] Diff editor cursor position is not preserved after esca	
552	issue	microsoft/vscode	171426	open		[Accessibility] Diff editor cursor position is not preserved after esca	
553	issue	microsoft/vscode	171256	closed		[Accessibility] Trigger diff audio cues against standard arrow key nav	
554	issue	microsoft/vscode	171253	open		[Accessibility] Allow users to customize the audio cue play priority	
555	issue	microsoft/vscode	171200	closed		support screen reader reading the line and audio cues when go to next/p	
556	issue	microsoft/vscode	171199	open		Accessibility getting started experience	feature-request, accessibi
557	pr	microsoft/vscode	170985	merged		support screen reader reading the line when go to next/previous diff are us	
558	issue	microsoft/vscode	170971	closed		[Accessibility]: Allow users to replace default sound file	*duplicate
559	issue	microsoft/vscode	169853	closed		Explore plain content editable element for terminal buffer instead of r	
560	issue	microsoft/vscode	168746	open		[Accessibility] Word wrap does not work in diff view (F7 and Shift+F7)	

573	pr	microsoft/vscode	167349	closed	fix	#167348: add aria-live	2023-02-16T17:31:02Z
574	issue	microsoft/vscode	167348	closed	[Accessibility]	div.monaco-tokenized-source requires aria-live=	
575	issue	microsoft/vscode	168814	open		Need a clearer landmark and label for notebook output area	fea
576	issue	microsoft/vscode	166518	closed		Add audio cues for Go to Next/ Previous Change commands	feature
577	issue	microsoft/vscode	166472	open	[Accessibility]	Add an option to allow Alt+F5 to jump to the ne	
578	issue	microsoft/vscode	165863	closed		Hitting spacebar does not replay currently focused audio cue	
579	issue	microsoft/vscode	165357	closed	[Accessibility]	Audio Cues still doesn't work in github.dev ins	
580	issue	microsoft/vscode	165161	open	[Accessibility]	Open Folder dialog controls do not have acceler	
581	issue	microsoft/vscode	164988	closed	[Accessibility]	Screen readers do not read currently focused l	
582	issue	microsoft/vscode	163506	open	[Accessibility]	Provide icon info to screen readers	feature-req
583	issue	microsoft/vscode	160301	open	[Accessibility]	Some long file content line is not correctly co	
584	issue	microsoft/vscode	159029	open		Merge editor accessibility improvements	accessibility, merge-ed
585	issue	microsoft/vscode	155919	closed	[Accessibility]	Support `Live Share` audio cues in `Help: List	
586	issue	microsoft/vscode	155655	closed	[Accessibility]	For easier code navigation, add jump to next/pr	
587	issue	microsoft/vscode	154027	closed	[Accessibility]	Terminal output is not read in real time on Ma	
588	issue	microsoft/vscode	147607	closed	[Accessibility]	Unlabelled `codicon` buttons	info-needed, ac
589	issue	microsoft/vscode	147386	closed	[Accessibility]	Add audio cues for indentation levels	feature
590	issue	microsoft/vscode	147230	open		Play audio-cues for auto-suggestions	feature-request, access
591	issue	microsoft/vscode	147226	closed	[Accessibility]	Consider adding audio cues for diffs (added / d	
592	issue	microsoft/vscode	147190	closed	[Accessibility]	Audio Cues doesn't work in web editor	bug, ve
593	issue	microsoft/vscode	143185	closed		Problems to access the preview of a markdown file using orca	
594	issue	microsoft/vscode	142983	closed	[Terminal accessibility]	JAWS does not speak anything against a	
595	issue	microsoft/vscode	141529	closed		Webviews displaying results of an API call with RestClient	exte
596	issue	microsoft/vscode	135920	closed	[Accessibility]	"xterm-accessibility" class div does not have t	
597	issue	microsoft/vscode	135035	closed	[Accessibility]	GitHub Web Editor: Cannot configure accessibili	
598	issue	microsoft/vscode	133876	closed	[Accessibility]	Assign a keyboard shortcut key to Focus Termina	
599	issue	microsoft/vscode	133805	closed		`Shift+Alt+R` for `Reveal in File Explorer` doesn't work when f	
600	issue	microsoft/vscode	133773	closed	[Accessibility]	"document" role is needed for "monaco-hover" cl	
601	issue	microsoft/vscode	132275	closed	[Accessibility]	Add "document" role to webview widget	verifie
602	issue	microsoft/vscode	131295	open	[Accessibility]	Character is not read properly in terminal input	
603	issue	microsoft/vscode	131090	closed	[Accessibility]	NVDA and JAWS do not read focused auto-suggesti	
604	issue	microsoft/vscode	130565	closed		Notify Screenreader Users When Inline Suggestions Or Decoration	
605	issue	microsoft/vscode	121735	closed		Terminal input does not work with NVDA	bug, important, accessi
606	issue	microsoft/vscode	113482	closed		Tab code-completion does not work in terminal input against scr	
607	issue	microsoft/vscode	111255	open		VS Code native notebook accessibility improvement	debt, acces
608	issue	microsoft/vscode	105425	closed		Garbage characters are inserted if you come back from terminal	
609	issue	microsoft/vscode	103095	closed		Auto-complete popup puts redundant "item" prefix per suggested	
610	issue	microsoft/vscode	98918	closed		Terminal output div container should be more accessible for scr	
611	issue	microsoft/vscode	95570	closed		Support terminal link keyboard navigation	feature-request, ac
612	issue	microsoft/vscode	90408	open		Feature request: Accessibility support for Jupyter notebooks in	