# Using the CS50 CodeSpaces with a Screen Reader

Your CS50 CodeSpace is a completely cloud-based development environment with everything you need for the course ready to use in your browser. It is based on the popular Visual Studio Code -- an open-source programmer's integrated development environment. VSCode for short runs on Windows, Mac and Linux. Your CS50 CodeSpace is running in your browser and on Ubuntu Linux which you can access from the VSCode terminal window. The advantage of using a CodeSpace over using VSCode on your own computer is that you need not own the computer, for example you might be using a computer at a library or school. Another advantage is that everything is configured for you already so you can concentrate on completing your CS50 work.

## Enabling Accessibility

If you were running VSCode on your own computer, it could auto-detect when a screen reader is also running and configure itself to enable its accessible mode. However, because it is running in a browser, this does not happen automatically. Also because you are running in a browser, that browser's keyboard shortcuts often override the numerous keyboard shortcuts VSCode provides for quick access to most features. Your first task then is to enable full-screen mode on your browser to tell it to ignore its native keystrokes. To do this on most browsers, press the F11 key. (Later, to exit full-screen mode, hold the Escape key for a second or two.)

Sometimes, when you switch to other tasks outside of your browser, the Full-Screen mode will turn itself off. You can always press F11 to re-enable it.

### Your Screen Reader Has Modes

Now that you are in full-screen mode, most keystrokes intended for VSCode will be active. With your screen reader, be sure you know how to move between its virtual cursor or browse mode to forms or focus mode. In your screen reader’s browse or virtual cursor mode, you are viewing the website as if it were a document, with a cursor that lets you review by character, word, line, or html element. Your screen reader provides many powerful keystroke-based ways to navigate, but in this mode you are reviewing and in general, unable to interact with a web page.

In Focus, or Forms mode, you are interacting directly with the website. The screen reader has detached itself from most keystrokes, but that means you are dependent on the web-based application to give you the clues you need to know where you are in that app. Your screen reader is still running, but its ability to review the screen has been mostly disabled. With most tasks in VSCode, you’ll want to have focus for forms mode on but you can use the virtual cursor or browse mode to always review and insure you know what’s onscreen and where you are located.

### Turning VSCode Screen Reader Mode On

Your next task, now that you have access to all VSCode keyboard shortcuts and the browser’s interface hidden, is to open the settings tab by holding control and pressing the comma key. An accessible tree view will appear. You move through one level with up and down arrows, using left and right arrows to expand or collapse a level. Some items have a level which you must reach by pressing Enter. There are so many settings that the tree view begins with a table of contents listing all major groups. The only setting you need to alter though at first is accessibility. You can find it either by arrowing to search, typing in access and pressing Enter. Or you can navigate there by arrowing to the Text editor group, moving down in to that group and finding accessibility which is its first item. Note that a search field is above the tree view, and if you tab past it you can reach a button to clear the search. If you continue to tab however, you will move your focus out of the tree view and the settings panel altogether.

The setting for Screen Reader compatibility is set to Auto. You need to change it to On by simply pressing Enter when your focus is on that item. Your settings take effect immediately; you do not need to save them.

## Layout of your CodeSpace

You first should understand the CodeSpaces screen layout and how to navigate it.

On the desktop, located at the top of the window is the menu bar which runs horizontally with submenus that drop down when you click or press a relevant keystroke. From left to right they are: File, Edit, Selection, View, Go, Run, Terminal and Help. Pressing Alt will move your focus there, and left and right arrow presses navigate you through this menu bar while up and down arrow presses navigate through the submenus attached to each main menu item.

But in your CodeSpace, you have an application menu with only some of these items, as CS50 has configured the CodeSpace to give you only what’s needed for your course.

### Tabs and Panels

At the top of the screen in your CodeSpace, (and in the desktop version of VSCode, directly below the menu bar) are various panels or what are often called tabs. Some are open and some are closed. In whatever way they are arranged, when you leave your CodeSpace they will appear exactly in the same arrangement when you return. On your first use, a getting started tab is open.

You can close unwanted tabs by pressing CTRL-W. If you forget to be in full-screen mode, you will find the CTRL-W keystroke closes your browser window and your CodeSpace, so be sure to enable full-screen mode before closing tabs with the keyboard. The control-J keystroke will hide these panels as well, except for the editor panel which will still remain open. Each panel has a toolbar with a button to maximize it for improved viewing. If you use magnification, you might also want to look through settings to adjust font size, colors and contrast – many themes and choices are available.

Typical tabs are:

* Problem Panel (CTRL+SHIFT+M)
* Output Panel (CTRL+SHIFT+U)
* Integrated Terminal (CTRL+` (grave accent)
* Debug Console (CTRL+SHIFT+Y)
* Explorer (CTRL+SHIFT+E)—File and folder management
* Search (CTRL+SHIFT+F)—Global search and replace across open folders using plain text or regular expressions
* Source Control (CTRL+SHIFT+G)—Git source control for maintaining code repositories
* Run (CTRL+SHIFT+D)—Features used during debugging, such as variables, call stacks, and breakpoints
* Extensions (CTRL+SHIFT+X)—Browsing, installation, and management of extensions from the Extension Marketplace

 These last five items: Explorer, Search, Source Control, Run, and Extensions are located on the activity bar at the far lefthand side of the screen. They open on the sidebar to its right.

But many other tabs can open as well. For example: CTRL-SHFTP or F1 will display the control pallet, Control-Comma opens user settings and CTRL-K-CTRL-S opens the keyboard settings dialog where you can view and alter keyboard shortcuts. A cheat sheet of all default keyboard shortcuts is available in accessible PDF at this link:

 https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf

As a user with a visual impairment, having so many tabs open can be more confusing than helpful. You can navigate between them with the F6 or Shift-F6 key, but it's often just easier to close one as soon as you no longer need it. Remember, if your F6 key fails to move you between panels, it is likely that the browser has grabbed that keystroke because your full-screen mode is not enabled.

### Editor and Terminal

The main portion of the screen is devoted to the code editor, which is where you write your program. Your screen reader will properly voice what's needed as you navigate by character, word or line, using your arrows and control arrow keystrokes. Many other keyboard shortcuts make writing code easier as well.

You will run, test and submit your code from the terminal panel. The control-up and control-down keystrokes navigate your focus in the terminal buffer, but the screen reader does not always voice the output. You will find as the terminal buffer grows the screen reader will only voice it partially. You can always issue the Linux command “clear” to clear the terminal view , giving your screen reader less to voice. You will need to switch your screen reader in to virtual cursor or browse mode to read as well as copy text from the terminal. This can be tedious, so consider as you begin to write longer programs, to test them locally on your own machine and then copy them back in to your CodeSpace to submit them. The advantage of using a CodeSpace though is that you don’t need to configure your own development environment locally, or even own your own computer to take the course. So it’s your decision to know what is more convenient for your situation.

At the bottom of the window is the status bar. You can navigate there with repeated presses of F6 and arrow through its content, pressing Space on any item you wish to expand.

## Tracking Indentation

When writing code, your screen reader has helpful features to inform you of indentation levels. In JAWS, you will find this in both the quick settings and settings center by searching for the word indent. Call up the Quick settings with the JAWS key plus the letter V. Activate Settings center by pressing the JAWS key with F6.

You can also set up a JAWS sound scheme to play tones at various indentation levels, for example the note C for no indentation, the note D for a level 1 indentation, a note E for level 2 indentation and so on. This lets you skim through code rapidly and know where blocks of indented text start and end. You can google for Jaws Sound Schemes to learn how this works.

In NVDA, pull up Preferences, Settings and document formatting. Tab to "line indentation reporting". A combo box list four choices: Off, speech, tones and both speech and tones. The pitch of tones rises as the indents grow deeper.

VoiceOver users can google for solutions, for example, several people have posted Apple scripts which voice indentations.

Orca on Linux can also indicate indentation levels, a setting found under its preferences which by default is off. Linux users developing locally may wish to investigate Emacspeak at:

 http://emacspeak.sourceforge.net/

VSCode also has a marketplace extension “Indent Report” which issues sound cues for indentation levels; find it at:

<https://marketplace.visualstudio.com/items?itemName=ogomez92.indent-report>

## Final Thoughts

A few helpful hints will make your experience with CodeSpaces more satisfying. Remember you can move between the editor and the terminal with the F6 or ShiftF6 key but only after you've enable full-screen mode with F11. You can close a tab with CTRL-W and you can use your virtual cursor or browse mode to review everything onscreen.

If you wish to switch between the instructions in an assignment and your CodeSpace, copy, paste and save your assignment instructions in a word processing document or local text editor. This way you can task switch between them so you need not disturb full screen mode to switch between it and the program you are developing. You need the control-tab key to switch between browser tabs, but you also need it to switch between tabs in your CodeSpace and this key conflict will drive you crazy if you try to work with multiple browser windows simultaneously.

Insure you have all punctuation enabled with your screen reader so you won’t inadvertently miss anything when proofreading your code. Unlike using social media, say, precision is paramount, so take your time to read carefully to avoid later frustration.