# Shortcuts

|  |  |  |
| --- | --- | --- |
| **Groups** | **Functions** | **Shortcuts** |
| **Edit** | Indent | Tab |
| Re-indent | Shift + Tab |
| Toggle single comment | Ctrl + / |
| Toggle block comment | Shift + Alt + A 🡪 change to Ctrl + Shift + / |
| Set multiple cursors | **Each time**: Hold Ctrl while clicking at different positions to add different cursors simultaneously  **All-in-one**:  Way 1: Black out the text, then press Shift + Alt + I. To move left / right / beginning / ending, press Left / Right / Home / End (to blackout while moving, combine these keys with Shift).  Way 2: Black out the text the mouse wheel. |
| Delete a whole word | Ctrl + Delete / Backspace |
| Delete line(s) | Ctrl + Shift + K |
| Duplicate line(s) | Shift + Alt + ↑ / ↓ |
| Copy / cut line(s) | Ctrl + C / X |
| Move line(s) up / down | Alt + ↑ / ↓ |
| Show completion | Ctrl + Space |
| **Navigation** | Quick jump | **Jump to file**: Ctrl + P  **Jump to line**: Ctrl + P, then type : (or Ctrl + G)  **Jump to method**: Ctrl + P, then type @  **Jump back**: Alt + ←  **Jump forward**: Alt + →  **Jump to last edit**: Ctrl + K + Q 🡪 change to Alt + -  **Jump to bracket**: Ctrl + Shift + \ 🡪 change to Alt + ]  **Jump to Definition**: F12  **Jump to Declaration**: Ctrl + F12  **Open Peek Definition**: Alt + F12  **Open Peek Declaration**: Ctrl + Alt + F12 |
| Select occurrences of current word | **Each time**: Ctrl + D  **All-in-one**: Ctrl + F2  **Redo**: Ctrl + U |
| Go to:   * Beginning of a line * Ending of a line * Beginning of a file * Ending of a file | * Home * End * Ctrl + Home * Ctrl + End |
| Switch between files | Ctrl + Tab |
| **View** | Toggle Sidebar | Ctrl + B﻿ |
| Open error and warning window | Ctrl + Shift + M |
| Toggle word wrapping | Alt + Z |
| **Others** | Save all | Ctrl + K + S 🡪 change to Ctrl + Shift + S |
| Show Preference / Setting | Ctrl + , |
| Scroll fast with mouse wheel | Alt + Scroll |

# Extensions

## C/C++

### [C/C++](https://code.visualstudio.com/docs/languages/cpp)

Adds language support for C/C++, including:

Language services:

* Syntax highlight
* Code formatting (clang-format)
* Snippet
* Go to definition/declaration
* Peek definition/declaration
* Class/method/symbol navigation
* Signature help
* Quick info (hover)
* Error squiggles
* IntelliSense

Debugging:

* Windows (PDB, MinGW/Cygwin), Linux and macOS applications
* Line by line code stepping
* Breakpoints (including conditional and function breakpoints)
* Variable inspection
* Multi-threaded debugging support
* Core dump debugging support
* Executing GDB or MI commands directly when using 'C++ (GDB/LLDB)' debugging environment

#### Configure IntelliSense

Without any configuration, extension will attempt to locate headers by searching your workspace folder and by emulating a compiler it finds on your computer.

If if this is insufficient, you can modify the defaults by running the *C/C++: Edit Configurations (UI)* command. This will create a c\_cpp\_properties.json file for you to modify.

Read more: <https://code.visualstudio.com/docs/cpp/faq-cpp#_how-do-i-get-intellisense-to-work-correctly>

#### Compile source files

In tasks.json:

{

    "tasks": [

        {

            "type": "shell",

            "label": "C/C++: g++.exe build active file", // name whatever you like

            "command": "C:/Program Files/mingw-w64/bin/g++.exe", // specify the program to run, in this case, g++

            "args": [ // specify args passed to g++

                "-g",

                "${fileDirname}/\*.cpp", // match all files with extension \*.cpp

// another way: "${workspaceFolder}/\*.cpp"

                "-o",

                "${fileDirname}/${fileBasenameNoExtension}.exe" // bin output file has the same name as active file

            ],

            "options": { // specify additional options for the task execution

                "cwd": "${workspaceFolder}" // set the current working directory to ${workspaceFolder}

            },

            "problemMatcher": [ // selects the output parser to use for finding errors

// and warnings in the compiler output

                "$gcc"

            ],

            "group": {

                "kind": "build", // indicate that the task belongs to a build group

                "isDefault": true // indicate that the task is the default task that

// will run when you initiate a build command in VSCode.

            }

        }

    ],

    "version": "2.0.0"

}

#### Debug

**Full guides:**

* https://code.visualstudio.com/docs/cpp/cpp-debug
* <https://code.visualstudio.com/docs/cpp/launch-json-reference>
* Windows: <https://code.visualstudio.com/docs/cpp/config-mingw>
* Linux: <https://code.visualstudio.com/docs/cpp/config-linux>

**Common issues:**

* To fix error *'Unable to launch debugger (gdb) with root permissions'*, run:

$ sudo sysctl -w kernel.yama.ptrace\_scope=0

#### Attach Debug

**When to use attach debug?**

1. Multi processes. We can debug multiproesses with breakpoints, but it's very difficult.
2. Skip building source code from VSCode. By default, starting debugging from VSCode triggers building the source code first, but if we want to skip the building process and go straight to the debugging process? One simple way is to use attach debug! (If needed, add a sleep for several seconds at the entry point of the program, so we can prevent our program running and finished so fast).

**How to configure attach debug?**

Step 1: Modify launch.json (only for the first time)

{

  "version": "2.0.0",

  "options": {

    // Environment variables

    "env": {

      // The PATH to system executables (and user executables). Each path is separated by a ':' notation (without spaces)

      "PATH": "/bin/:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/other\_paths/to/binary/files",

      // Path(s) to dynamic libs (.so files). Each path is separated by a ':' notation (without spaces)

      "LD\_LIBRARY\_PATH": "/path1/to/dynamic/libs:/path2/to/dynamic/libs"

    }

  },

  "configurations": [

    {

      "name": "myConfig1",                      // Configuration name (can be anything)

      "processId": "${command:pickProcess}",    // Pick a process from drop-down list every time starting debugging

      "program": "/path/to/executable1",        // Specify an executable file's path. Can use ${workspaceFolder}

      "type": "cppdbg",

      "request": "attach",                      // Here we want ATTACH debugging

      "MIMode": "gdb",

      "setupCommands": [

        {

          "description": "Enable pretty-printing for gdb",

          "text": "-enable-pretty-printing",

          "ignoreFailures": false

        }

      ],

      "miDebuggerPath": "/usr/bin/gdb"          // Path to debugger

    },

    {

      "name": "myConfig2",                      // Another configuration name (can be anything)

      "processId": "${command:pickProcess}",    // Pick another process from drop-down list every time starting debugging

      "program": "/path/to/executable2",        // Specify another executable file's path. Can use ${workspaceFolder}

      // ..... Similar settings as above

    }

  ]

}

Step 2: Run the executable

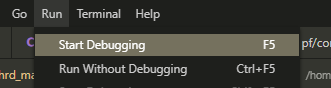
Run the executable which you want to attach debug.

Step 3: Attach the executable

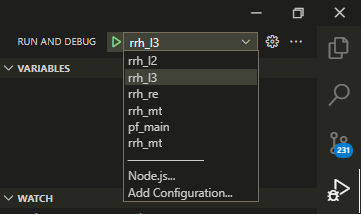
From VSCode, select the executable which you just run at the step 2.

Note:

* Don't start debugging from menu, which cannot pick process:



* But start debugging from sidebar:

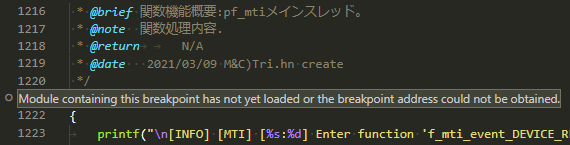


Common issues:

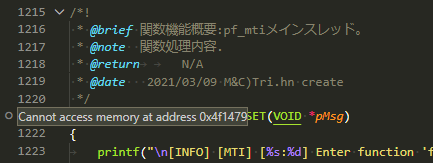
* To fix error *'Superuser access is required to attach to a process. Attaching as superuser can potentially harm your computer. Do you want to continue? [y/N]'*, run:

$ echo 0| sudo tee /proc/sys/kernel/yama/ptrace\_scope

* Following error is caued by setting wrong process in launch.son:



* Unknow:



### [C++ Helper](https://marketplace.visualstudio.com/items?itemName=amiralizadeh9480.cpp-helper)

Generating:

* Implementation for declarations.
* Header guard for headers

### [C++ Algorithm Mnemonics](https://marketplace.visualstudio.com/items?itemName=davidbroetje.algorithm-mnemonics-vscode)

Provides code templates for C++ STL algorithms.

## C#

### [C#](https://marketplace.visualstudio.com/items?itemName=ms-vscode.csharp)

Adds language support for C#, including:

Language services:

* Code formatting
* Snippet
* Go to definition
* Peek definition
* Find all references
* Class/method/symbol navigation
* Quick info (hover)
* Error squiggles/Quick fixes/Suggestions
* Add and show/hide using
* IntelliSense
* CodeLens
* Support for project.json and csproj projects on Windows, macOS and Linux

Debugging:

* Support for .NET Core (CoreCLR). NOTE: Mono debugging is not supported. Desktop CLR debugging has limited support.

### [C# XML Documentation Comments](https://marketplace.visualstudio.com/items?itemName=k--kato.docomment)

Generate XML documentation comments for VSCode (built-in feature in VS IDE).

## Python

### [Python](https://marketplace.visualstudio.com/items?itemName=ms-python.python)

Adds language support for C#, including: IntelliSense, linting, debugging, code navigation, code formatting, Jupyter notebook support, refactoring, variable explorer, test explorer, snippets, and more!

### [Pylance](https://marketplace.visualstudio.com/items?itemName=ms-python.vscode-pylance)

The default performant feature-rich language support for above Python extension – shipped as part of that extension as an optional dependency.

### [autoDocstring](https://marketplace.visualstudio.com/items?itemName=njpwerner.autodocstring)

Quickly generate a docstring snippet (with briefs, parameters, return values, decorators, errors, etc.) that can be tabbed through.

## HTML & CSS

### [HTML CSS Support](https://marketplace.visualstudio.com/items?itemName=ecmel.vscode-html-css)

Add CSS supports for HTML documents:

* Class attribute completion
* Id attribute completion.
* Supports Zen Coding completion for class and id attributes.
* Scans workspace folder for CSS and CSS files.
* Supports remote CSS files.

Tip: Specify remote CSS files to cache. For example, add the following setting to cache Bootstrap 4 remotely:

"css.remoteStyleSheets": [

"https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css"

]

### [CSS Peek](https://marketplace.visualstudio.com/items?itemName=pranaygp.vscode-css-peek)

Allow *Go To Definition*, *Peek Definition*, *Go To Symbol in Workspace*, *Hover* for CSS classes.

**NOTE:**

* Don't use [HTML Snippets](https://marketplace.visualstudio.com/items?itemName=abusaidm.html-snippets): Its features are supported by the default VSCode HTML extension.

### [Quick HTML Previewer](https://marketplace.visualstudio.com/items?itemName=daiyy.quick-html-previewer)

Side and full preview HTML and CSS files.

## PHP

### [PHP Intelephense](https://marketplace.visualstudio.com/items?itemName=bmewburn.vscode-intelephense-client)

Adds language support for PHP, including: IntelliSense, auto completion, signature help, code navigation, hover, error reporting, refactoring and more!

**Warning:**

There is another IntelliSense extension called [**PHP IntelliSense**](https://marketplace.visualstudio.com/items?itemName=felixfbecker.php-intellisense). Don't understand why, but it does not work for me!

### [PHP Debug](https://marketplace.visualstudio.com/items?itemName=felixfbecker.php-debug)

Debug adapter between VSCode and XDebug. We need this extension to debug PHP code in VSCode.

### [phpcs](https://marketplace.visualstudio.com/items?itemName=ikappas.phpcs)

Code snipper for PHP. It can detect violations of a defined coding standard and automatically correct them.

### [Beautify](https://marketplace.visualstudio.com/items?itemName=HookyQR.beautify)

Format code for JavaScript, JSON, CSS, Sass, and HTML.

Config to work with PHP: <https://stackoverflow.com/a/46855721>

### For Laravel

#### [Laravel Snippets](https://marketplace.visualstudio.com/items?itemName=onecentlin.laravel5-snippets)

Collect almost all snippets for Laravel framework.

**Tip**: We can copy snippets (in json files) from this extension to make our own snippet file. On Windows, extensions are stored in %USERPROFILE%\.vscode\extensions.

#### [Laravel Blade Snippets](https://marketplace.visualstudio.com/items?itemName=onecentlin.laravel-blade)

Collect almost all snippets for Laravel Blade. Plus, syntax highlight for .blade.php files.

#### [laravel-goto-controller](https://marketplace.visualstudio.com/items?itemName=stef-k.laravel-goto-controller)

Alt + click to navigate from a route to a respective controller file. PHP IntelliSense doesn't support this feature.

#### [Laravel goto view](https://marketplace.visualstudio.com/items?itemName=codingyu.laravel-goto-view)

Ctrl + click to navigate from a route to a respective view file. PHP IntelliSense doesn't support this feature.

## Markup Language

### [reStructuredText](https://docs.restructuredtext.net/)

This plugin provides rich reStructuredText language support for VSCode, including:

* Syntax highlighting
* Code snippets
* Live preview
* Section builder
* Linter
* IntelliSense

### [Text Tables](https://marketplace.visualstudio.com/items?itemName=RomanPeshkov.vscode-text-tables)

Help create and format table easily.

### [PlantUML](https://marketplace.visualstudio.com/items?itemName=jebbs.plantuml)

This plugin provides rich support for PlantUML in VSCode, including:

* Preview Diagram (Alt-D)
  + Instant preview
  + Auto update
  + Zoom & scroll
* Export Diagrams
  + Export to images
  + Generate URLs.
* Editing Supports
  + Format PlantUML code
  + Syntax highlight
  + Snippets
  + Auto Include
  + Symbol List support

Supported formats: \*.wsd, \*.pu, \*.puml, \*.plantuml, \*.iuml

### Markdown

#### [All In One](https://marketplace.visualstudio.com/items?itemName=yzhang.markdown-all-in-one)

Not tried, but seem to be great.

## CSV

### [Rainbow CSV](https://marketplace.visualstudio.com/items?itemName=mechatroner.rainbow-csv)

* Highlight columns in comma (.csv), tab (.tsv), semicolon and pipe in different colors
* Provide info about column on hover
* Automatic consistency check for csv files (CSVLint)
* Align columns with spaces and shrink (trim spaces from fields)
* Run queries in SQL-like language
* Multi-cursor column editing

### [Preview & Edit CSV](https://marketplace.visualstudio.com/items?itemName=janisdd.vscode-edit-csv)

View and edit csv files like in Excel. Very powerful!

## SQL

### [SQLTool](https://marketplace.visualstudio.com/items?itemName=mtxr.sqltools)

Support various DMSs, including: MySQL, MS SQL Server/Azure, PostgreSQL, SQLite, AWS Redshift

Provide following features:

* Connect to server
* Execute SQL and view results in a grid
* IntelliSense
* Query history
* Bookmarks

Guide: [SQLTools (mteixeira.dev)](https://vscode-sqltools.mteixeira.dev/?umd_source=repository&utm_medium=readme&utm_campaign=mysql)

### [MySQL](https://marketplace.visualstudio.com/items?itemName=cweijan.vscode-mysql-client2)

Provide following features:

* Connect to MySQL server
* Execute SQL and view results in a grid
* Generate mock data
* Query history
* Backup/Import
* Filter table by names

### [SQL Server (mssql)](https://marketplace.visualstudio.com/items?itemName=ms-mssql.mssql)

Provide a rich set of functionalities, including:

* Connect to Microsoft SQL Server, Azure SQL Database and SQL Data Warehouses.
* Write T-SQL script with IntelliSense, Go to Definition, snippets, syntax colorizations, error validations and GO batch separator.
* Execute SQL and view results in a grid.
* Save the result to JSON or CSV file format.

Guide: [Transact-SQL with Visual Studio Code](https://code.visualstudio.com/docs/languages/tsql)

## Yocto

### Yocto BitBake

Language support for BitBake with following features:

* Syntax highlighting
* Context-based suggestions
* Go to definition
* Get recipe variable value (right click and select *BitBake: Get Variable Value*)
* Show recipe reference (when hover the pointer on top of the variable)

Note: For the extension to work correctly, set path to the BitBake executable and the init build environment script.

## Shell Script

### [Shellman Snippet](https://marketplace.visualstudio.com/items?itemName=Remisa.shellman)

Learn how to write professional shell scripts easily.

### [Shell Script Command Completion](https://marketplace.visualstudio.com/items?itemName=tetradresearch.vscode-h2o)

Support shell commands with following features:

* Autocomplete command-line options, flags, and subcommands in shell scripts
* Introspection with mouse hover (to get descriptions for options, flags and subcommands)
* Extra Command Specs for Bioinformatics

### [Bash Debug](https://marketplace.visualstudio.com/items?itemName=rogalmic.bash-debug)

GUI for bashdb debugger.

### [ShellCheck](https://marketplace.visualstudio.com/items?itemName=timonwong.shellcheck)

Integrates ShellCheck – a static analysis toolfor Shell scripts – into VSCode.

## [PowerShell](https://marketplace.visualstudio.com/items?itemName=ms-vscode.PowerShell&ssr=false#overview)

Language support for PowerShell with following features:

* Syntax highlighting
* Code snippets
* IntelliSense
* Problems reported by PowerShell Script Analyzer
* Go to definition
* Find references
* Symbol-based outline view
* Document and Workspace Symbol Navigation
* Run selected PowerShell code in current terminal using F8
* Launch online help for the symbol under the cursor using Ctrl+F1
* Debugger integration
* An Extension Terminal that can interact with the debugger (try Set-PSBreakpoint!)
* PowerShell ISE theme findable in the theme picker
* Ty ISE Mode with the Toggle ISE Mode command

## Visual Enhancements

### [Monokai](https://marketplace.visualstudio.com/items?itemName=AndreyVolosovich.monokai-st3) ST3

This theme is exactly the same Monokai's theme of Sublime Text but for Visual Studio Code instead.

### [Text Marker (Highlighter)](https://marketplace.visualstudio.com/items?itemName=ryu1kn.text-marker)

Highlight selected text for better view.

### [TODO Highlight](https://marketplace.visualstudio.com/items?itemName=wayou.vscode-todo-highlight)

Highlight TODO, FIXME and your own defined annotations within your code.

### [Bracket Pair Colorizer 2](https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-colorizer-2)

Match brackets with colors. The user can define which tokens to match and which colors to use.

## Remote Development

### [Remote – SSH](https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh)

This extension helps you open project workspaces from any **remote machine**, **VM**, or **container** via **SSH**.

So, you can work on the remote machine just as you do on your own machine. Leverage all of the great features of VSCode such as **IntelliSense**, **code navigation**, **debugging**, etc. as if you were working locally.

Refs:

* <https://code.visualstudio.com/docs/remote/remote-overview>
* <https://code.visualstudio.com/blogs/2019/05/02/remote-development>

## Take Screenshot

### [CodeSnap](https://marketplace.visualstudio.com/items?itemName=adpyke.codesnap)

Take beautiful screenshot of code.

Note:

"codesnap.showWindowTitle": true,

"codesnap.showWindowControls": false

## Git

### [GitLens](https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens)

Supercharges the capabilities of the built-in Git extension in VSCode.

### [Gerrit](https://marketplace.visualstudio.com/items?itemName=SanderRonde.vscode--gerrit)

Integrating Gerrit into VSCode.

### [Diff Viewer](https://marketplace.visualstudio.com/items?itemName=caponetto.vscode-diff-viewer)

Visualize git patch file (side by side).

## Others

### [Path Autocomplete](https://marketplace.visualstudio.com/items?itemName=ionutvmi.path-autocomplete)

Provides path completion. To make it work, start a path with './' or '/' or 'C:' or '~'.

### [Compare Folders](https://marketplace.visualstudio.com/items?itemName=moshfeu.compare-folders)

Compare folders and show the diffs.

Note: Before using the extension, configure some of its settings such as ignoring file extension, choosing left/right folder, etc.

### [Doxygen Documentation Generator](https://marketplace.visualstudio.com/items?itemName=cschlosser.doxdocgen)

Creates and updates Doxygen-style documentation comments in code.

### [Comment Translate](https://marketplace.visualstudio.com/items?itemName=intellsmi.comment-translate)

Uses the Google Translate API to translate comments among different languages.

### [Code Spell Checker](https://marketplace.visualstudio.com/items?itemName=streetsidesoftware.code-spell-checker)

Check spelling mistakes for both code and comments.

### [Bookmarks](https://marketplace.visualstudio.com/items?itemName=alefragnani.Bookmarks)

Mark any line of code. Support Side Bar badge, jumping between bookmarks, naming bookmarks, etc.

### [Save As Snippet](https://marketplace.visualstudio.com/items?itemName=artydeveloperduck.save-as-snippet)

Convert code blocks to snippet on the fly.

Online alternative: <https://snippet-generator.app/>

### [Box Code](https://marketplace.visualstudio.com/items?itemName=PabloAlmonte.boxcode)

Save fragments of your code to use it at another time.

### [Better Align](https://marketplace.visualstudio.com/items?itemName=wwm.better-align)

Align code by colon (:), assignment(=, +=, -=, \*=, /=) and arrow (=>). It has additional support for comma-first coding style and trailing comment.

### [Favorites](https://marketplace.visualstudio.com/items?itemName=kdcro101.favorites)

Add files and directories to workspace favorites. Create groups of directories and files.

### [AutoScroll](https://marketplace.visualstudio.com/items?itemName=pejmannikram.vscode-auto-scroll)

Provide auto scroll to the end of files when files have been changes outside of VSCode. Something can be helpful in watching log files.

# Tips

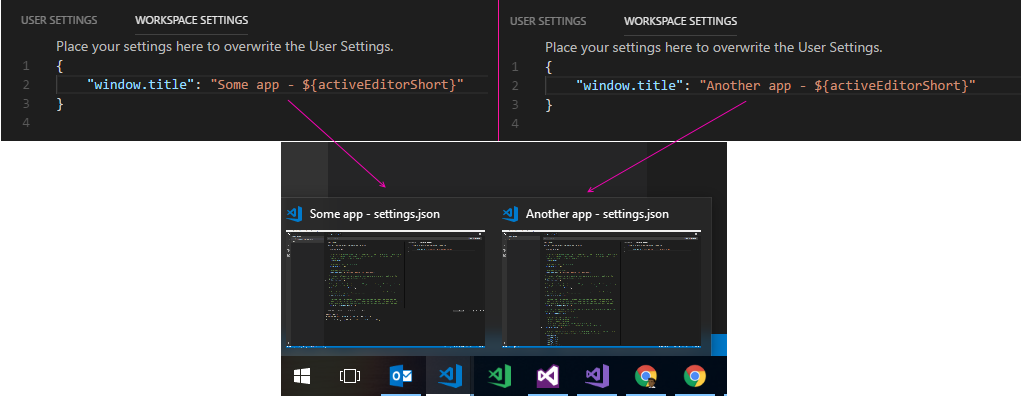
## Create Snippets

<https://code.visualstudio.com/docs/editor/userdefinedsnippets>

## Add Windows Title

Sometimes, you have many VSCode windows (from different workspaces) opening at the same time. And it could be hard to identify all of them.

Solution: Adding "window.title" in workspace settings.json.



Below are supported values you can use in the title template:

"window.title": "${dirty}${activeEditorShort}${separator}${rootName}${separator}${appName}"

// ${activeEditorShort}: the file name (e.g. myFile.txt)  
 // ${activeEditorMedium}: the path of the file relative to the workspace folder (e.g. myFolder/myFile.txt)  
 // ${activeEditorLong}: the full path of the file (e.g. /Users/Development/myProject/myFolder/myFile.txt)  
 // ${folderName}: name of the workspace folder the file is contained in (e.g. myFolder)  
 // ${folderPath}: file path of the workspace folder the file is contained in (e.g. /Users/Development/myFolder)  
 // ${rootName}: name of the workspace (e.g. myFolder or myWorkspace)  
 // ${rootPath}: file path of the workspace (e.g. /Users/Development/myWorkspace)  
 // ${appName}: e.g. VSCode  
 // ${dirty}: a dirty indicator if the active editor is dirty  
 // ${separator}: a conditional separator (" - ") that only shows when surrounded by variables with values or static text

Ref: <https://code.visualstudio.com/docs/getstarted/settings>

# My settings.json

{

  "workbench.colorTheme": "Monokai",

  "editor.minimap.enabled": false,                // Disable mini map

  "workbench.sideBar.location": "right",          // Set side bar to the right side

  "editor.rename.enablePreview": false,           // Prevent closing current tab when opening new tab

  "workbench.editor.enablePreview": false,        // Prevent closing current tab when opening new tab

  "editor.renderWhitespace": "all",               // Show space or tab indentation

  "editor.insertSpaces": false,                   // Prevent converting Tab into Spaces

  "files.trimTrailingWhitespace": false,          // Must use when working on existing code base

  "files.exclude": {                              // Dir / files which are not shown in the Explorer

      "\*.bak": true,

      "Debug": true,

      "Release": true

  },

  "editor.multiCursorModifier": "ctrlCmd",        // Enable multi cursors in VSCode

  "editor.mouseWheelZoom": true,                  // Enable zooming out-in with mouse wheel

  "window.zoomLevel": 0,                          // Zoom out level

  "editor.fontSize": 14,                          // Set font size

"editor.fastScrollSensitivity": 8, // Speed up scrolling

  "workbench.colorCustomizations": {              // Customize color for text highlight

      "editor.selectionHighlightBackground": "#af7979",

      "editor.selectionBackground": "#cc9191",

  },

"workbench.editor.decorations.colors": true, // Set color for tab (error, new, etc.)

"workbench.editor.decorations.badges": true,

"workbench.editor.wrapTabs": true, // Wrap tabs over multile lines

"editor.bracketPairColorization.enabled": true, // Enable bracket colorization

"workbench.editor.openPositioning": "last", // Put latest-opened file at the last tab

// Gitlens

// Disable inline blame

"gitlens.currentLine.enabled": false,

"gitlens.codeLens.authors.enabled": false,

"gitlens.codeLens.recentChange.enabled": false,

"diffEditor.ignoreTrimWhitespace": true, // Ignore white space in diff tool

}