Contents

Introduction

Visual studio code (often fondly abbreviated to VSCode) is a free open-source code editor developed by Microsoft. It is highly customizable with a vast array of powerful extensions giving it the complete power of an Integrated Development Environment (IDE). In this document we log quality of life tips and tricks made by the current user to act both as a log and a useful guide for new users. The guide will generally summarize information and include useful links for self learning.

For an in-depth guide to getting started with VSCode it is good to start with introduction.

Installation

VSCode can be downloaded directly from here.

Settings

Settings in VSCode can be opened via the dropdown menu, File->Preferences->Settings, or via the following shortcut `Ctrl+,`. Settings can be opened within two modes, either in UI mode,

or in JSON mode which can can be toggled via the Open Settings button,

Extensions

VSCode extensions can be installed from the marketplace tab,

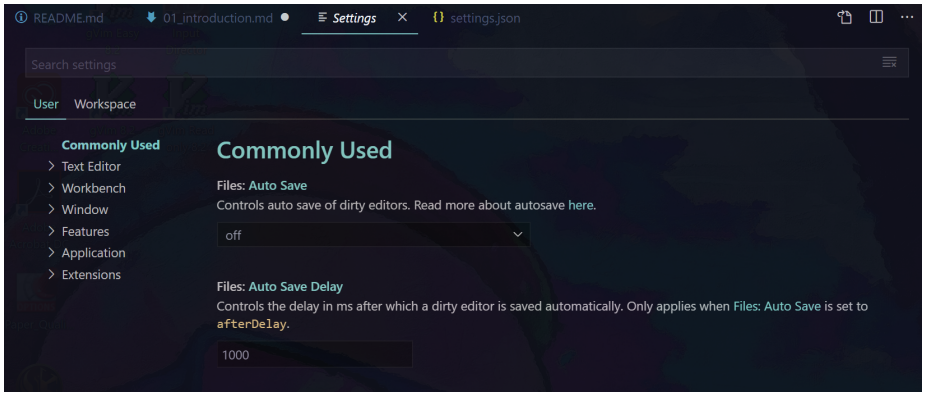

Figure 1: Settings (UI)

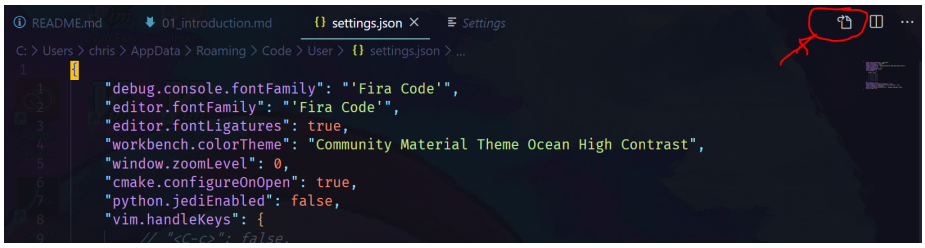

Figure 2: Settings (JSON)

Figure 3: Marketplace

The market place tab can be quickly accessed via the `Ctrl+Shift+x` shortcut. Alternatively the marketplace can be accessed via the web browser

Useful Shortcuts

NOTE: {} means is not literal and means the value of some variable.

Function	Command	Notes
open the command palette	`Ctrl+p`	
open the command palette for commands	`F1`	
run debug	`F5`	
continue debug	`F5`	
stop debug	`Shift+F5`	
enter zen mode	`Ctrl+k, z`	To enable line numbers“zenMode.hideLineNumbers”: false,“zenMode.hideTabs”: false
split screen horizontally	`Ctrl+\`	
move to screed id	`Ctrl+{id}`	
open explorer	`Shift+Ctrl+e`	
open search	`Shift+Ctrl+f`	
open search control	`Shift+Ctrl+g`	
open debugging	`Shift+Ctrl+d`	
open extensions	`Shift+Ctrl+x`	

Version Control

Version control is naturally support by VSCode, just install Git for Windows.

Useful Resources

- Git Cheat Sheet
- Git Fast Version Control Documentation
- Git Versioning in VSCode

To automatically fetch every day set in the `settings.json`

```
"git.autofetch": true,
```

Vim Emulation

Vim is a powerful text editing language,

To allow for relative line numbers for easier jumping, in `settings.json` add,

```
"editor.lineNumbers":"relative"
```

We may wish to allow some shortcut keys to retain their original VSCode bindings. To do so we just disable vim from handling them. For example we wish to retain the use of the 'Ctrl+k' keys, in the `settings.json`,

```
"vim.handleKeys": {
    "<C-k>": false,
    "<C-b>": false,
    "<C-n>": false,
    "<C-s>": false
},
```

Also by default vim does not use the system clip board, we can enable the clipboard,

```
"vim.useSystemClipboard": true,
```

We must note that the vim useSystemClipboard command is system intensive at the moment and should probably not be used.

Explorer

Vim can be used directly in the explorer, to search for files we can employ `/` then just type in the filename.

Useful Resources

- Vim Cheat Sheet
- Vim Adventures
- Code Fu With Vim and VSCode

Some Useful Commands

Change Word Surroundings

To surround some code with new text,

```
hello = "HelloWorld"
```

if we want to convert `"` to `()`, highlight the `"` and type in normal mode,

```
cs"(
```

we see the command is of the form `cs<target><new>`, we can read this as `change surrounding target with new`

Surround Word/Words

let's say we forget to surround some work in parenthesis,

`hello = HelloWorld`

we can surround the word in vim with,

`ysw"`

when we wish to surround two words,

`hello = Hello World`

we can use the following

`ys2w"`

we see this command is of the form `ys<numwords>w<new>`, and can be read as `yank surround <numwords> words with <target>`

Making Things Pretty

To truly experience the joys of programming customizability of the editor is an absolute necessity! It is paramount to your sanity to have both an organized and good looking editor for those inevitable long nights filled with writing or coding. Also, it's a fun way to procrastinate and still feel productive.

Pretty Fonts

A lot of our work is going to be within either the text editor or the terminal. A nice looking font not only improves the looks of you're code but significantly improves readability, reducing the energy necessary for parsing, scanning and joining multiple characters.

I personally recommend installing fira code. The font contains ligatures for common programming multi-character combinations and, in my personal experience, helps to significantly improve the readability of code.

After following the installation instructions we can setup `Fira Code` in windows as the font for our `editor`, `integratedTerminal` and `debugConsole` by adding the following lines to our `settings (JSON)`,

```
"editor.fontFamily": "Fira Code",
"editor.fontLigatures": true,
"terminal.integrated.fontFamily": "Fira Code",
"debug.console.fontFamily": "Fira Code",
```

For more fonts check out Nerd Fonts.

For a nice guide check out Scott Hanselman

Opacity

Sometimes you might have a cool background that you would like to see, set the opacity of the entire editor with GlassIt-VSC, in `settings (JSON)`,

```
"glassit.alpha": 235
```

Coloured Brackets

It can sometimes be difficult to discern which brackets belong to which group, for such cases Rainbow Brackets is a must.

Better Icons

Icons provide visual feedback onto what a filetype is and what a folder contains. vscode-icons provides good looking icons for an improved experience when exploring the icon tree.

Deleting White Spaces

Trailing spaces can cause issues,

- when using macros which bring you to the end of the line
- when directly reading lines from codes

Trailing Spaces provides macros which can be used to delete all trailing white spaces in a file

Markdown

All In One

All in one go to for markdown support in VSCode, Markdown all in one includes preview, autocomplete, shortcuts and more.

Spell Checking

For spell checking can use Spell Right.

Rendering

Markdown has a lot more uses than simple website documentation, it can be compiled with pandoc into multiple formats.

Useful Resources

-Markdown Cheat Sheet

Table Generation

It may become difficult to generate tables in markdown. Luckily there's an online interactive tool which can be used to generate and format tables called the Tables Generator. Ensure to save the generator files when finished for any future changes. This can save hours of time inputting new data into a table.

C/C++

The all in one visual studio code extension for C/C++ provides a set of useful tools to making c++ coding easier.

CMake

CMake can be used for cross platform compilation, the VSCode extension makes it significantly easier to find and compilers and build projects. First install CMake, afterwards we can install the CMake Language Support and finally CMake Tools.

Useful Resources

- The Holy Bible of Numerics
- Configuring VSCode for C/C++ # Latex

VSCode has some nice tools which allows for direct and easy editing of latex files. An all-one extension to use is Latex Workshop. For Latex Workshop to work a LaTeX distribution will be required, the two most popular are,

- Miktex
- Tex Live

Word Wrapping

When writing in LaTeX in VSCode the words may not automatically wrap to a new line. To enable auto word wrap we can use the `Alt+F1` shortcut to we can do so from the command palette by typing `toggle word wrap`. Note however that enabling word wrap way may cause issues with vim.

Reference Handling

When handling references it is common to use a `.bib` file. It can become a hassle copying and pasting references, maintaining keys and checking for integrity. Luckily for us there are a lot of reference handling software available, to name a few,

- Mendeley
- Zotero
- Bookends

- JabRef

I personally use JabRef, but any form of reference handling can make things significantly easier when writing scientific articles.

Spell Checking

Can be performed either using Spell Right or Code Spell Check. I personally use Spell Right.

Grammar

For grammar checking we can use LTeX

Python

A must have tool for VSCode python development is Python.

Once installed for any new projects I recommend creating a new virtual environment.

```
python -m venv pathToVirtualEnvironment
```

An alternative to using a virtual environment would be to use anaconda or miniconda

When VSCode starts when you open a Python file you will automatically be prompted to select a python interpreter. If a virtual environment is installed VSCode will automatically activate it instead.

To select a python interpreter manually press `F1` to open up the command line and type,

```
>python: select interpreter
```

Select the first option and you will be prompted to select from the various environments cached by VSCode.

When python is activated it can be called via the `py` alias

Useful Resources

- Python Course