Agenda

1. Editor to IDE - Evolution
2. Installation & Walkthrough
3. Run "Hello World" page in VS Code
4. Debugging In VS Code

What is an IDE?

In simple words, an IDE is a handy piece of software that acts as a text editor, debugger, and compiler all in one. IDEs are designed to make coding easier for developers.

So basically, an IDE is an application that facilitates application development, and gives you a central interface featuring all the tools you'll need like:

-A **code editor** that's designed to help you write and edit your code. It also helps you make it more readable and clean.

-A **compiler** that transforms code written by a human into machine-readable form.

-A **debugger** that helps you eliminate errors from your programs so that your code executes and performs the way it should. This feature provides tools to help you examine your code.

Compiler vs Debugger

Compiler	Debugger
The compiler converts the source code to equivalent machine code for the computer to understand and execute the tasks defined in the program.	Debugger helps to identify the errors in a program and to fix them correctly.
Converts high-level programming language to low-level programming language.	Finds errors in the high-level programming language.
It is a software or set of software.	It is a computer program
Compilers convert the code at once.	The debugger allows you to run your code step by step and it can halt when it crashes.
Languages like C, C++ have compilers	GNU Debugger (GDB), Microsoft Visual Studio Debugger- popular debuggers

Evolution of Editors :

Back in the day, you could say that coding was "text-only". Developers used to write code in a text editor (like Notepad, Emacs, and others). They would write and save applications in the editors with various extensions like .java, and then later run the compiler, note the errors, and go back and fix them until the code worked. Over time, these various activities started to get baked into coding environments and became more automated with just the click of a few buttons. Microsoft's Visual Basic was the first real IDE, and later many companies developed different IDEs for different languages.

- **Command Line Editors** : There is no GUI, you use a keyboard to do all the CRUD operations on files and folders.

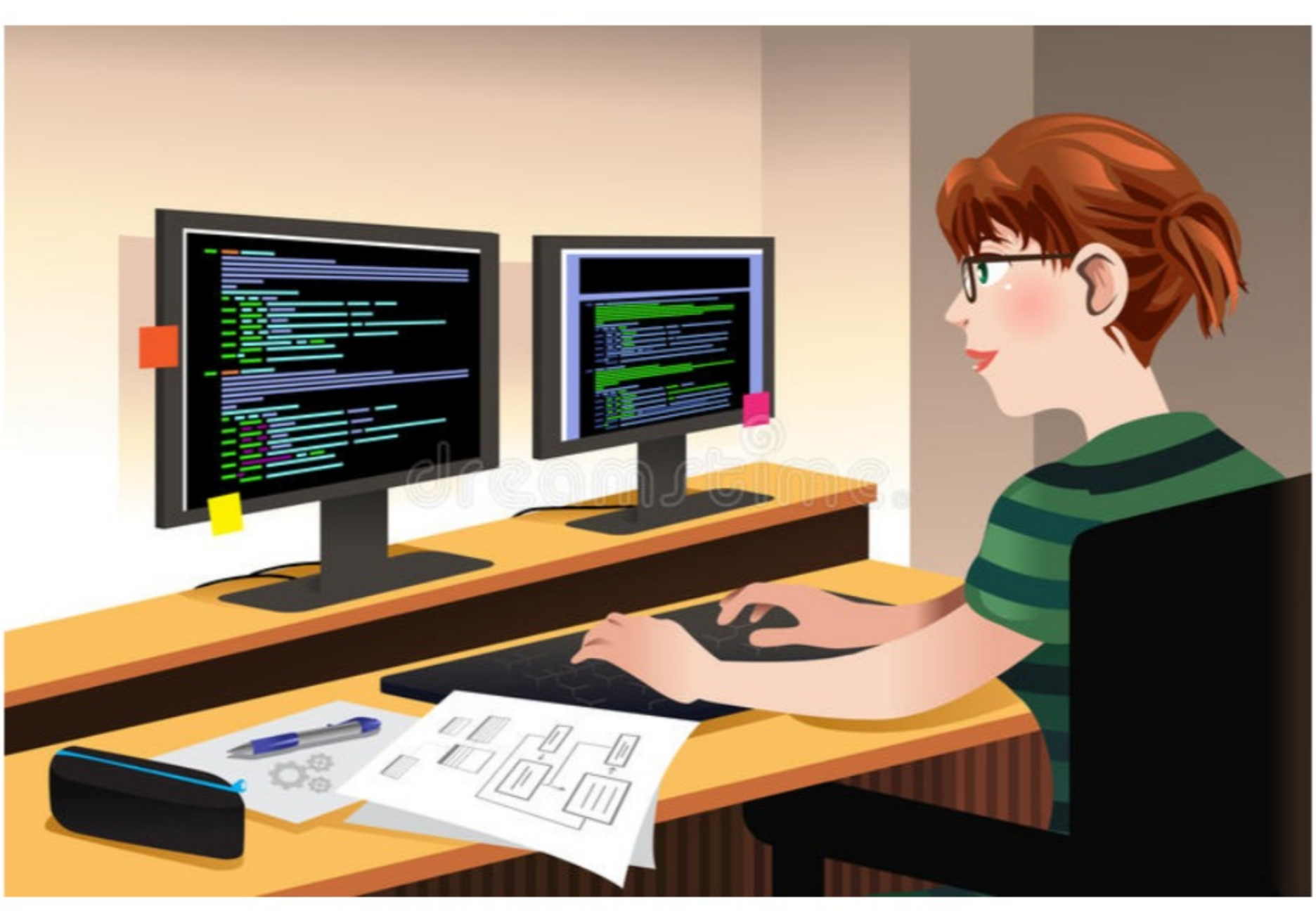
Examples: MS DOS Editor, Ref2 & VI Editor (Unix / Mac OS)

- **GUI Editors** : Everything in Command line editors + Graphical User Interface. Use keyboard + mouse to do CRUD operations. features like auto complete texts, auto-formatting, theme etc.

Examples : Notepad & Notepad++, Atom & Sublime

- **Integrated Development Environment** : Rich set of plugins to make your development faster & productive, support to build, debug the code. You don't need to navigate to different applications to complete development, IDE has lots of things in built, avail it at just a mouse click.

Examples: IntelliJ, Eclipse, Visual Studio (VS Code).



Common Features of IDEs

Most IDEs have certain features in common that prove very useful when you're coding.

-**Intellisence** – IDEs with this feature can provide code completion, quick info, and member lists on a project.

-**Smart code editing** – IDE can indent code lines, match words and brackets, and also highlight keywords.

-**Debugger** – Most IDEs provide powerful debugging features along with a graphical debugger and breakpoints.

-**Extension/plugin manager** – You can add new extensions/plugins to extend your IDE's functionality.

-**Version Control** – IDEs offer support for various version control systems like Git, Subversion, Mercurial, CVS and so on.

IDE vs General-Purpose Text Editor

IDE vs General-Purpose Text Editor

IDE	Text Editor
An Integrated Development Environment (IDE) (or interactive development environment) is a software application that provides comprehensive facilities to computer programmers for software development. It has a source code editor, build automation tools, and a debugger, and many support lots of additional plugins and extensions.	Text Editors are simpler applications compared to IDEs, they usually correspond to just the code editor segment of an IDE.
IDEs are created to serve the purpose of software development.	Text editors are designed to be used by non-developers.
IDE can detect bugs and naming inconsistencies across classes and modules, and even across files, directly in the editor, before compiling	Text Editors can detect language errors and spelling errors.
Example: VSCode, Pycharm, Eclipse	**Example:** Notepad, WordDoc

Why Should You Use an IDE When You Code?

Here are some of the main advantages of using an IDE:

- **IDEs save you time and effort** – the entire purpose of using an IDE is to make development faster and easier. IDEs do this by providing you with many helpful resources, shortcuts, error recognition, and more.
- **They're Easy to Setup** – An IDE brings different capabilities together in one place and therefore reduces the struggle of constantly switching between tools.
- IDEs can **correct syntax**, give warnings, and help you write quality code.

Why have we chosen VS Code for the curriculum?

- Developing an IDE needs lots of engineering work; hence most IDEs come with some cost. VS Code is free and developed by Microsoft.
- VS code is the most popular IDE among frontend, backend and full-stack engineers.
- VS Code has a huge set of plugins that increases your productivity and scope of integration.
- VS Code has a large community of developers for help and support.

Download

Download VS Code - Quickly find the appropriate install for your platform (Windows, macOS and Linux)

Installation Instructions

While we demonstrate installation for Windows here.

You can find and follow the installation instruction specific to your operating system:

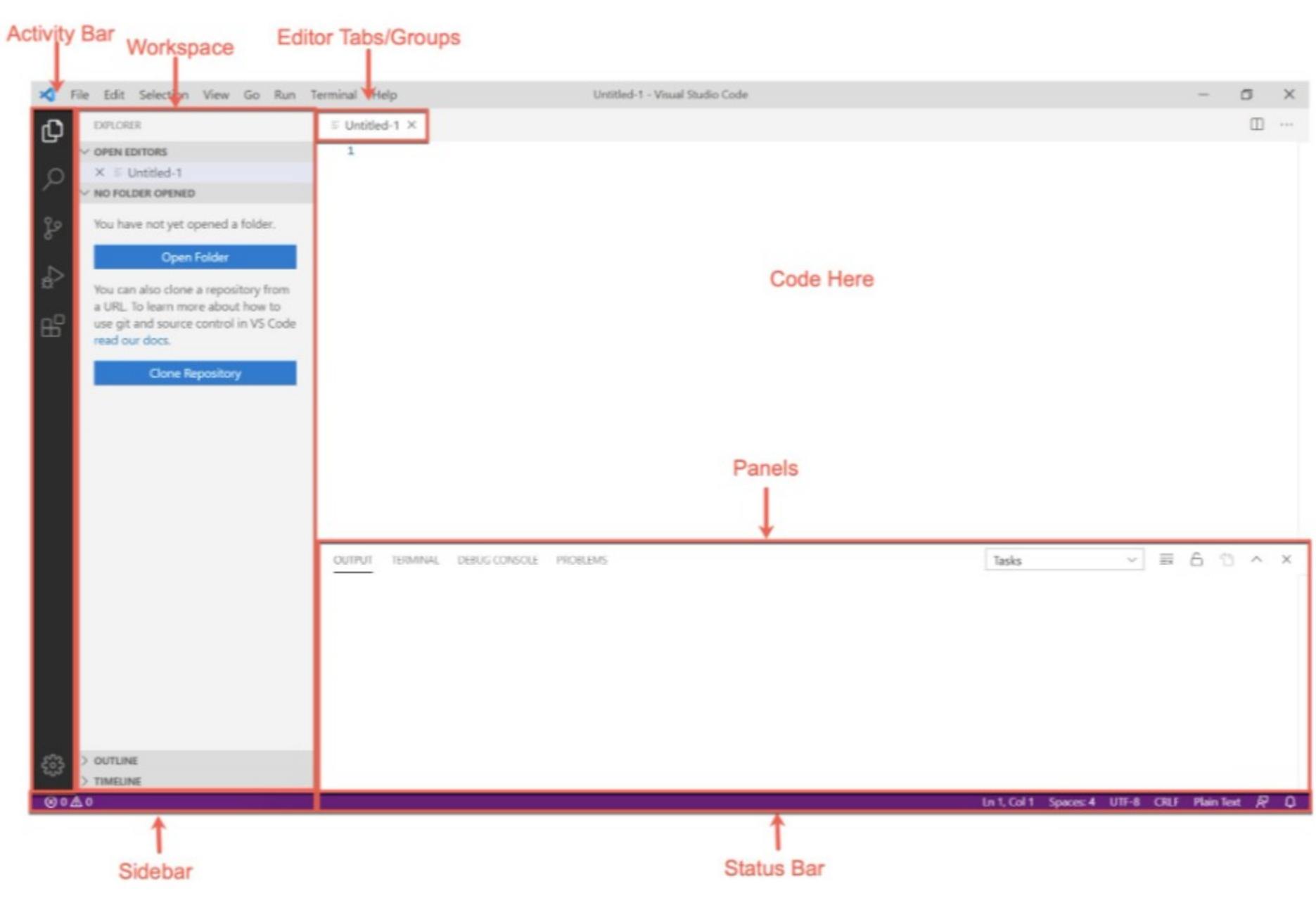
- Windows
- macOS
- Linux
- Raspberry Pi

Installation

1. Download the Visual Studio Code installer for Windows.
2. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). This will only take a minute.
3. By default, VS Code is installed under `C:\\Users\\{Username}\\AppData\\Local\\Programs\\Microsoft VS Code`.

Interface Tour

When you open up VS Code for the first time, you will see a user interface that looks like the following screenshot. You'll see that VS Code has a few main areas you'll be using frequently.



The main VS Code interface can be broken down into five distinct areas:

-**Editor Window (Tabs/Groups)** – The editor window is where you'll be doing most of your work. This pane is where you will view and edit all of the code you're working on. Whenever you open a new file or edit an existing file, the editor window is where you'll the code will show up. VS Code has tabs in this editor pain that allow you to open up multiple files at once and editor groups that group various tabs.

-**Workspace** – The workspace will be the next most common part of the UI you'll be using. The workspace is where any files you have open in tabs will show up. You'll commonly open entire folders too to see all files in a particular folder at once here.

-**Sidebar** – The sidebar is where you'll see information such as the Git repo you have open, the name of a Git branch you're working under, and the ability to push Git changes to a remote repo.

-**Panels** – The panels section is the "output" section. You will find various "tabs" with information returned by VS Code and its extensions under this pane. Here is where you will also find the handy integrated terminal. The integrated terminal is a built-in Bash terminal (with other shells included via extensions) that allows you to run code directly in VS Code without having to open a separate shell.

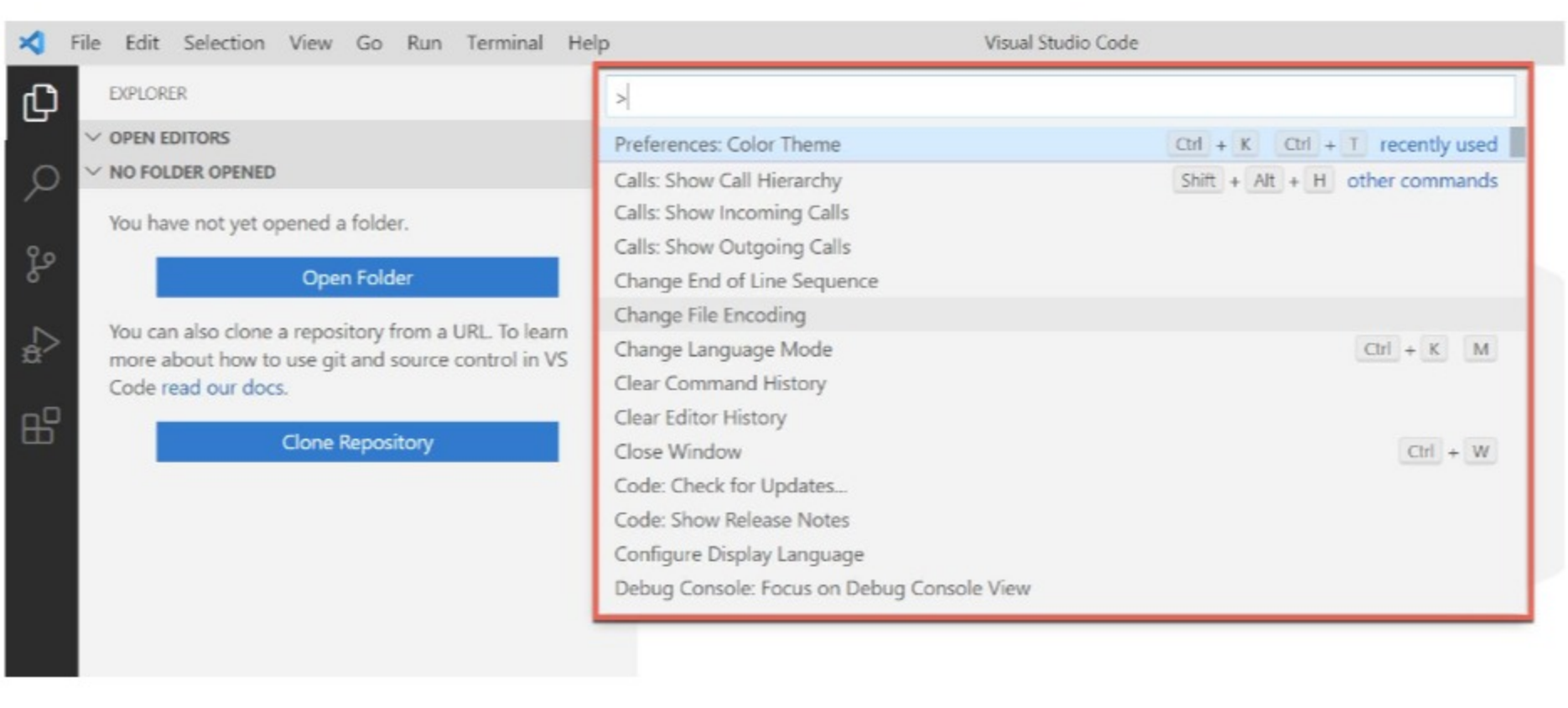
-**Status Bar** – The status bar provides information about the open editor tab. The status bar shows cursor position, encoding, the format VS Code recognizes the file format to be, among other things. The status bar is where VS Code and its extensions will also display information as they run.

Commands and the Command Palette

There's a lot to do in VS Code, especially if you have installed many extensions (covered later). You can control a lot of that functionality via the typical File, Edit, and **View** menus at the top of the window but not everything.

The easiest way to make things happen in VS Code is via commands found in the *command palette*. The command palette is a menu that appears at the top of the screen when you click on the **View** menu and select **Command Palette,** or you hit Ctrl-Shift-P on your keyboard.

In the following screenshot, you can see an example of the command palette.



VS Code command palette

The command palette is where you will find all functionality of VS Code. The command palette is great because all functionality for managing tasks, settings, snippets, and more are located in a single menu (covered later).

The command palette is where you will find everything you can do in both a default VS Code installation and configure any extension you have installed.

First Webpage
We'll now create our first webpage.

Please follow along in the session and use the code snippet given below

```
<!DOCTYPE html>

<html>
    <head>
        <title>
            Hello World!
        </title>
    </head>
</html>
```

Go further with Visual Studio Code's features

If you already feel comfortable with the previous steps, explore the following features to further customize your development environment. You don't need to use these suggestions to complete the projects on Codecademy but they can help make you more efficient when writing code and are what make Visual Studio Code such a useful editor!

- Debugging code in the editor: That's right, you can run and test code from the editor!
- Integrated terminal: You can run command line commands from your editor with Visual Studio Code.
- Version control: You don't need to switch to the terminal on your computer to track changes with Git.