


FELIX KIHIMA LIVAHA
felixkihima63@gmail.com

1)Steps to Install Visual Studio Code on Windows

Step 1: Visit the [Official Website](#) of the **Visual Studio Code** using any web browser like [Google Chrome](#), [Microsoft Edge](#),

Download Visual Studio Code


Free and built on open source. Integrated Git, debugging and extensions.



↓ **Windows**

Windows 7, 8, 10, 11

User Installer	64 bit	32 bit	ARM
System Installer	64 bit	32 bit	ARM
.zip	64 bit	32 bit	ARM



↓ **.deb**


Debian, Ubuntu

↓ **.rpm**

Red Hat, Fedora, SUSE

.deb	64 bit	ARM	ARM 64
.rpm	64 bit	ARM	ARM 64
.tar.gz	64 bit	ARM	ARM 64

Snap Store



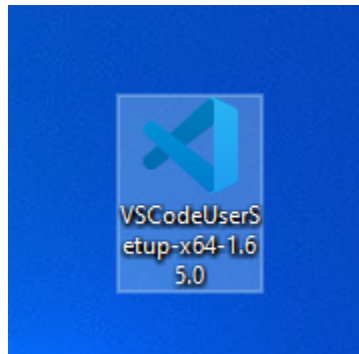
↓ **Mac**

macOS 10.11+

.zip **Universal** **Intel Chip** **Apple Silicon**

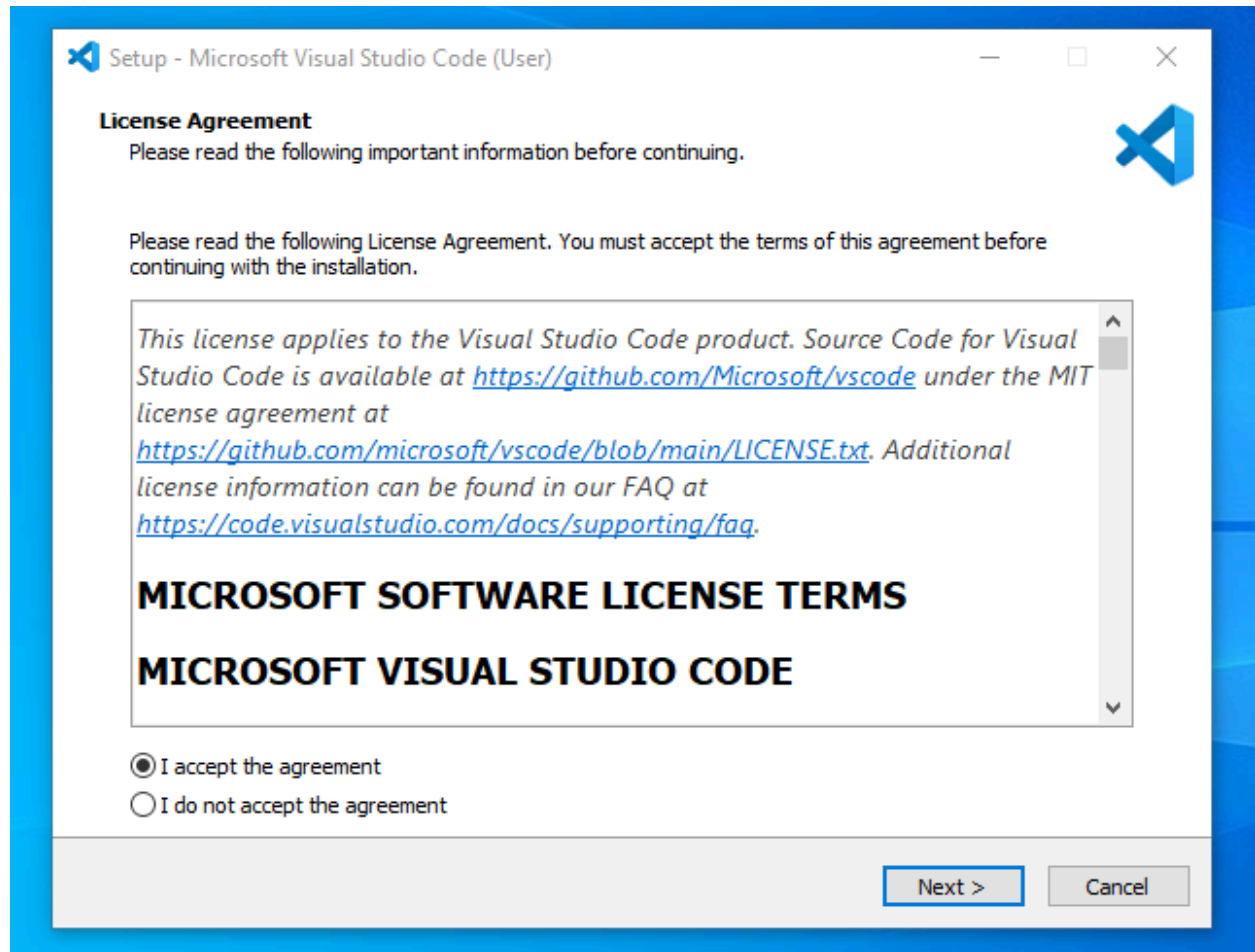
Step 2: Press the “**Download for Windows**” button on the website to start the download of the Visual Studio Code Application.

Step 3: When the download finishes, then the **Visual Studio Code Icon** appears in the downloads folder.

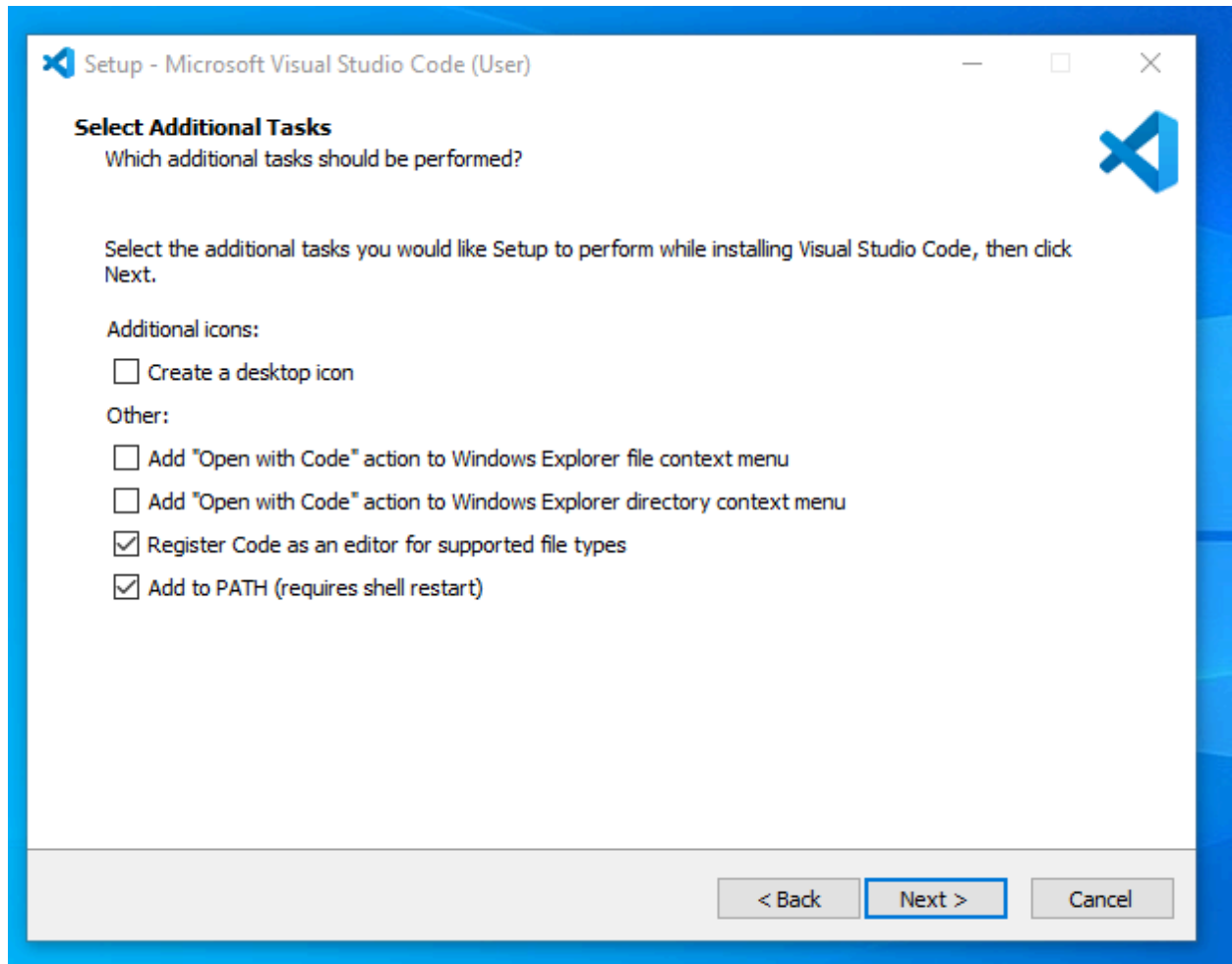


Step 4: Click on the **Installer** icon to start the installation process of the Visual Studio Code.

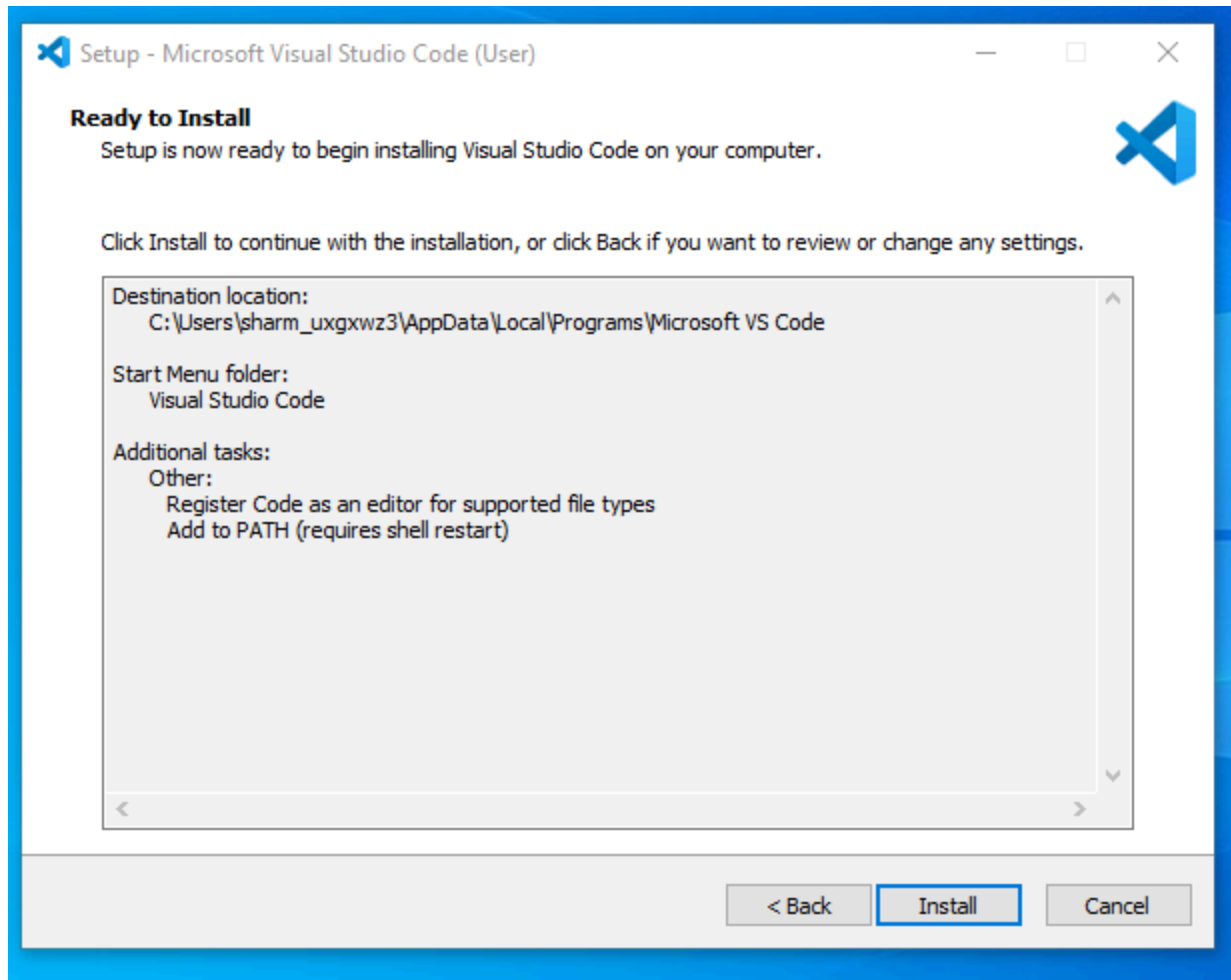
Step 5: After the Installer opens, it will ask you to accept the terms and conditions of the Visual Studio Code. Click on **I accept the agreement** and then click the **Next** button.



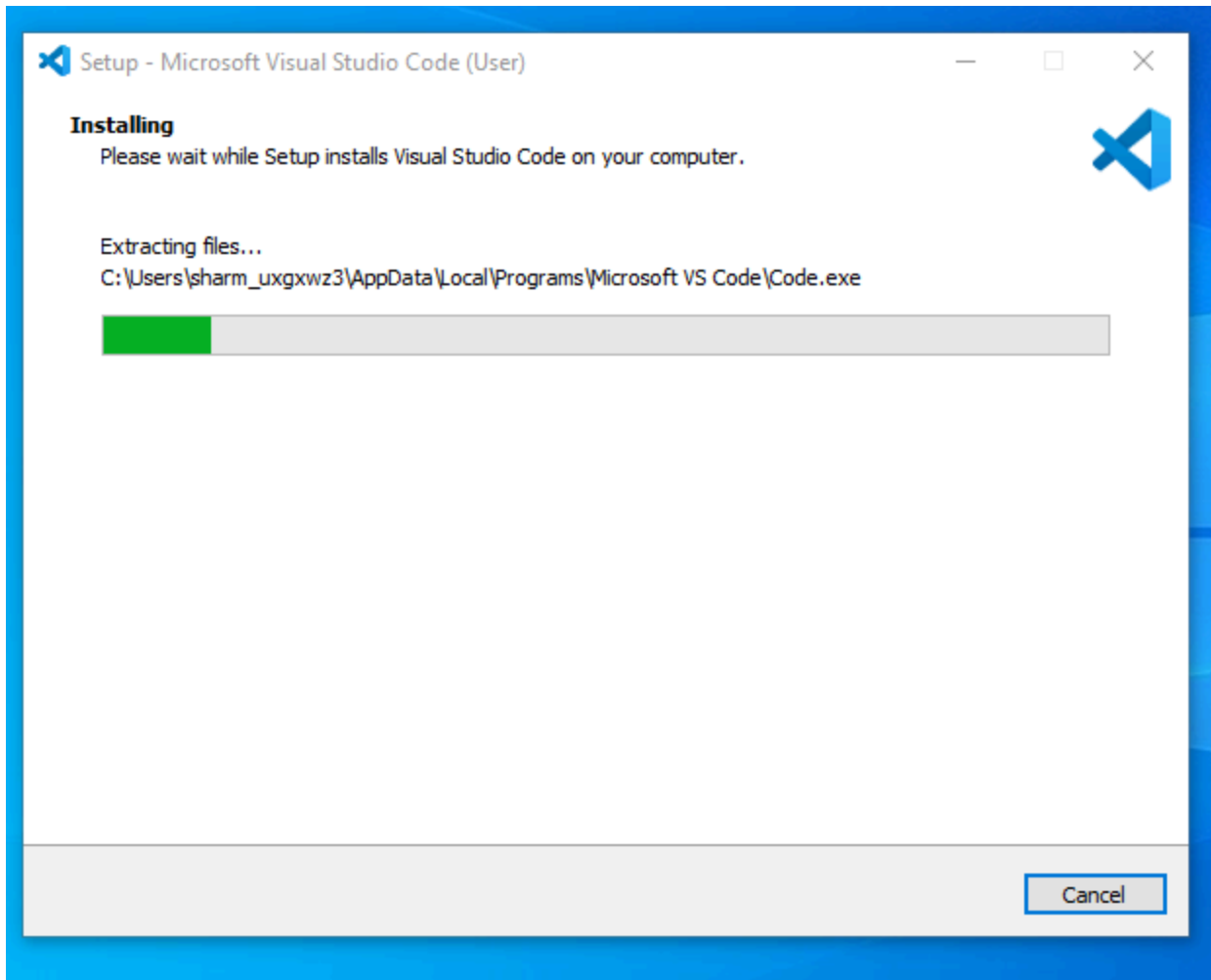
Step 6: Choose the location data for running the Visual Studio Code. It will then ask you to browse the location. Then click on the **Next** button.



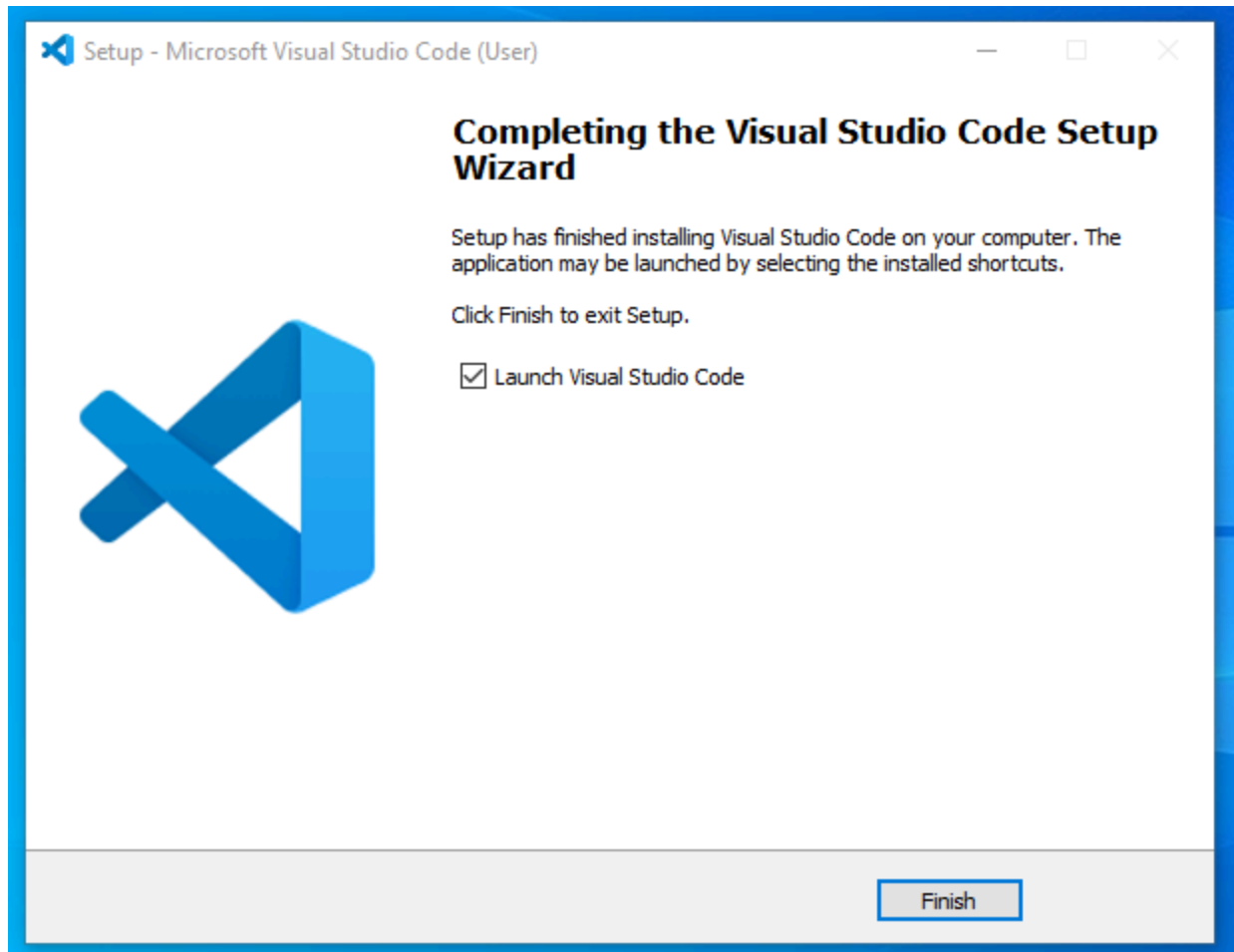
Step 7: Then it will ask to begin the installation setup. Click on the **Install** button.



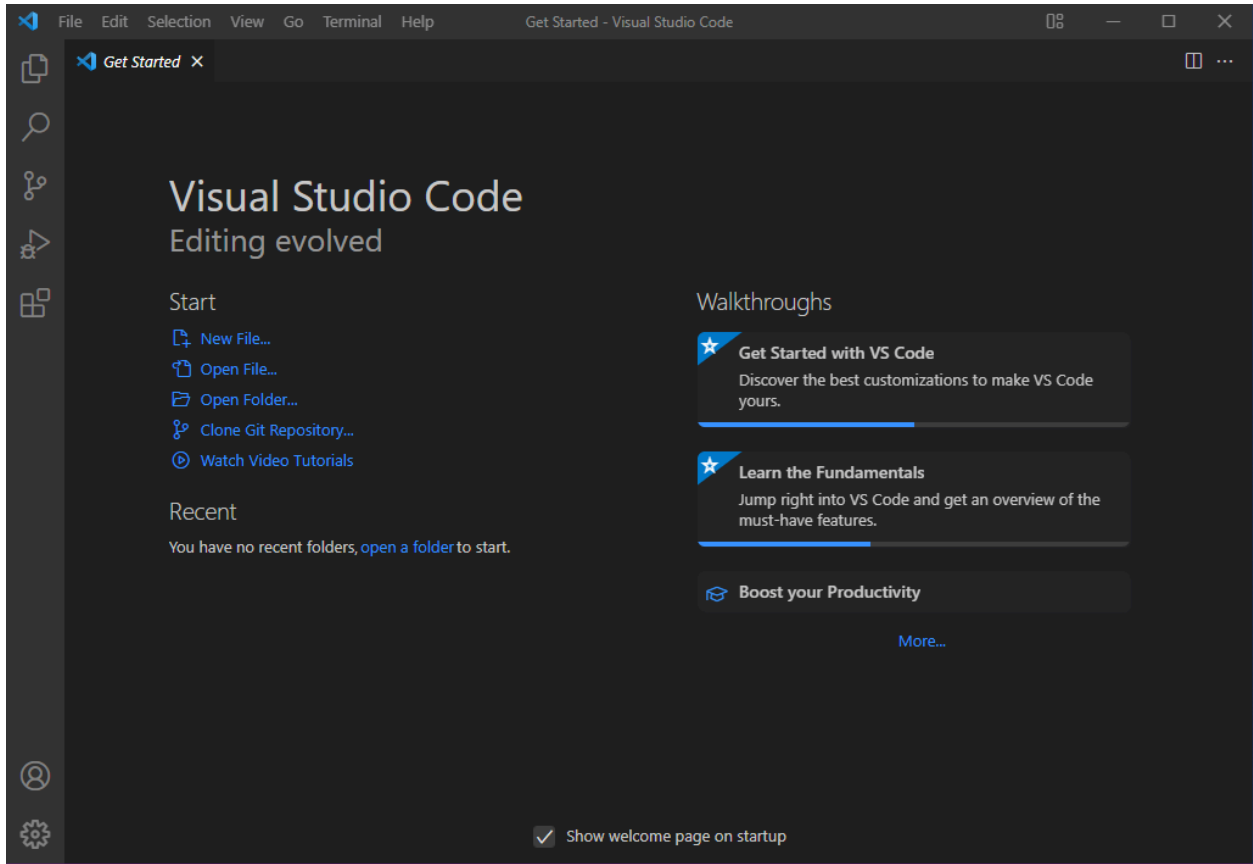
Step 8: After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.



Step 9: After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the “**Launch Visual Studio Code**” checkbox and then click **Next**.



Step 10: After the previous step, the **Visual Studio Code window** opens successfully. Now you can create a new file in the Visual Studio Code window and choose a language of yours to begin your programming journey!



2)

Font & Theme: Set your preferred font and theme (e.g., dark or light) under File > Preferences > Settings (`settings.json`).

json

Copy code

```
"editor.fontFamily": "Consolas, 'Courier New', monospace",  
"workbench.colorTheme": "Default Dark+"
```

•

Indentation & Formatting:

json

Copy code

```
"editor.tabSize": 2,  
"editor.formatOnSave": true
```

•

Line Wrapping:

json

Copy code

```
"editor.wordWrap": "on"
```

-

Cursor Style:

json

Copy code

```
"editor.cursorStyle": "line"
```

-

Auto Save: Choose when files should be automatically saved.

json

Copy code

```
"files.autoSave": "afterDelay"
```

-

Code Lens: Enable/disable Code Lens (inline code actions).

json

Copy code

```
"editor.codeLens": false
```

-

Useful Extensions:

- **Programming Languages:** Install extensions for languages you work with (e.g., Python, JavaScript, Java).
- **Linters & Formatters:** Install extensions like ESLint, Prettier, or Black for automatic linting and code formatting.
- **Version Control:** GitLens for enhanced Git functionality.
- **Debugger:** Debugger for Chrome for debugging JavaScript in the Chrome browser.
- **Theme & Icons:** Install themes and icons that suit your taste (e.g., Material Theme, Material Icon Theme).
- **Productivity:** Bracket Pair Colorizer for visually matching brackets, Bookmarks for quick navigation, and TODO Highlight for highlighting TODOs and FIXMEs in your code.

Keybindings:

- Customize keybindings ([keybindings.json](#)) for your preferred shortcuts and workflow.

Integrated Terminal:

Configure the integrated terminal ([settings.json](#)):

json

Copy code

```
"terminal.integrated.shell.osx": "/bin/bash",  
"terminal.integrated.fontSize": 14
```



Workspace Settings:

Save workspace-specific settings ([settings.json](#)):

json

Copy code

```
"[javascript]": {  
  "editor.tabSize": 4  
}
```



User Interface Tweaks:

- Adjust minimap settings, breadcrumbs visibility, and activity bar visibility based on your preference ([settings.json](#)).

Other Recommendations:

- Explore and configure Git integration.
- Learn and use VS Code's advanced search and replace features.
- Stay updated with VS Code's release notes for new features and improvements.

These configurations and settings provide a solid foundation for a comfortable and efficient coding environment in VS Code. Adjust them further as per your specific needs and coding style preferences.

3)

Activity Bar:

- Purpose: The Activity Bar is located on the far left side of the window and provides quick access to various views and panels within VS Code.

- Components:
 - Explorer: Allows navigation through files and folders in your project.
 - Search: Provides search functionality across your project.
 - Source Control: Integrates with version control systems like Git for managing changes.
 - Run and Debug: Facilitates running and debugging applications.
 - Extensions: Manages VS Code extensions for additional features and functionality.

2. Side Bar:

- Purpose: The Side Bar is located next to the Activity Bar and typically displays contextual information and actions related to the currently selected view or file.
- Components:
 - Explorer: Shows the file structure of your project for easy navigation.
 - Search: Displays search results and allows further filtering and actions.
 - Source Control (Git): Provides Git integration features like staging changes and viewing commit history.
 - Extensions: Lists installed extensions and allows management (enable/disable/configure).

3. Editor Group:

- Purpose: The Editor Group is where you view and edit your code files. It occupies the central area of the VS Code window.
- Functionality:
 - You can have multiple Editor Groups open simultaneously, each containing different files or views (e.g., split vertically or horizontally).
 - Tabs within each Editor Group allow quick switching between open files.

4. Status Bar:

- Purpose: The Status Bar is located at the bottom of the VS Code window and provides information and actions related to the current workspace and file.
- Information Displayed:
 - Language Mode: Displays the programming language of the current file.
 - Git Branch: Shows the current Git branch and status (e.g., clean, changes pending).
 - Line Endings: Indicates the line endings used in the file (e.g., LF, CRLF).
 - Encoding: Displays the character encoding of the file.

- Extension Status: Some extensions may display additional information or actions in the Status Bar.

Additional Components:

- Minimap: A small preview of the entire file's contents, allowing for quick navigation.
- Breadcrumbs: Located above the Editor Group, provides navigational breadcrumbs based on the file's structure.

Customization and Extensions:

- VS Code's interface can be customized extensively through themes, settings adjustments (in `settings.json`), and the installation of extensions that add new functionality or customize existing features.

4) Accessing the Command Palette:

To access the Command Palette in VS Code, you can use the following methods:

- Keyboard Shortcut: Press **Ctrl+Shift+P** (Windows, Linux) or **Cmd+Shift+P** (Mac).
- Menu: Go to **View > Command Palette...**

Once opened, the Command Palette displays a text input field where you can start typing to search for commands. As you type, it dynamically filters and shows relevant commands and options.

Examples of Common Tasks Using the Command Palette:

1. File and Folder Operations:
 - Create New File: Type "New File" and select "File: New File" to create a new file.
 - Open File: Type part of the file name and select "File: Open File" to open a specific file.
 - Open Folder: Type "Open Folder" to add a new folder to your workspace.
2. Editing and Navigation:
 - Find and Replace: Type "Replace" or "Find" to access options like "Replace in Files" or "Find in Files".
 - Go to Line: Type "Go to Line" and enter a line number to navigate directly to that line in the current file.
3. Version Control (Git):
 - Git Operations: Type "Git" to see options like "Git: Pull", "Git: Commit", "Git: Push", etc., for managing Git repositories.

4. Extensions:
 - Install Extensions: Type "Extensions: Install Extensions" to search for and install new extensions from the VS Code Marketplace.
 - Manage Extensions: Type "Extensions: Show Installed Extensions" to manage installed extensions.
5. Settings and Configuration:
 - Open Settings: Type "Preferences" or "Settings" to access and modify VS Code settings (`settings.json`).
 - Change Color Theme: Type "Color Theme" to switch between installed color themes.
6. Run and Debug:
 - Run Commands: Type "Run" or "Debug" followed by your application name to run or debug your code.
 - Start Debugging: Type "Debug: Start Debugging" to begin debugging your application.
7. Tasks and Build:
 - Run Task: Type "Tasks: Run Task" to execute a predefined task (e.g., build task) configured in `tasks.json`.
8. Miscellaneous:
 - Toggle Sidebar Visibility: Type "View: Toggle Sidebar" to show or hide the Sidebar.
 - Toggle Integrated Terminal: Type "View: Toggle Integrated Terminal" to open or close the integrated terminal.

Advantages of Using the Command Palette:

- Efficiency: Quickly execute commands without leaving your keyboard.
- Discoverability: Easily explore and discover available commands and options.
- Flexibility: Supports a wide range of tasks, from file management to debugging and customization.

5)

Role of Extensions:

Extensions in VS Code serve several purposes:

1. **Enhanced Functionality:** They add features like language support, debugging capabilities, code snippets, and integrations with external tools and services.

2. **Customization:** Extensions enable customization of themes, icons, key bindings, and UI components to match personal preferences or specific project requirements.
3. **Productivity Tools:** Many extensions automate repetitive tasks, improve code quality with linters and formatters, and provide shortcuts for common workflows.

Finding and Installing Extensions:

To find and install extensions in VS Code:

1. **Using the Command Palette:**
 - Open the Command Palette (**Ctrl+Shift+P** or **Cmd+Shift+P**).
 - Type "Extensions: Install Extensions" and press Enter.
 - Search for extensions by name or category.
2. **Using the Extensions View:**
 - Click on the Extensions icon in the Activity Bar.
 - Search for extensions directly in the search bar.
 - Click on an extension to view details and install it.
3. **VS Code Marketplace:**
 - Visit the [Visual Studio Code Marketplace](https://openmarketplace.visualstudio.com/) in a web browser.
 - Search for extensions by name, category, or tag.
 - Click on an extension to view details and install it directly into VS Code.

Managing Extensions:

Once installed, extensions can be managed in several ways:

- **Disabling and Enabling:** Toggle extensions on or off to activate or deactivate their functionality.
- **Configuring Settings:** Some extensions provide customizable settings through the VS Code settings interface (**settings.json**).
- **Updating Extensions:** VS Code notifies you when updates are available, and you can update extensions directly from the Extensions view.

Examples of Essential Extensions for Web Development:

1. **ESLint:** Provides integration with ESLint for JavaScript and TypeScript linting.
2. **Prettier:** Code formatter for JavaScript, TypeScript, CSS, and other languages.
3. **Live Server:** Launches a local development server with live reload capability for HTML, CSS, and JavaScript files.
4. **Debugger for Chrome:** Enables debugging of JavaScript applications in the Chrome browser directly from VS Code.

5. **GitLens:** Enhances Git integration with advanced features like inline blame annotations, repository file history, and more.
6. **HTML CSS Support:** Provides autocompletion, syntax highlighting, and other features for HTML and CSS development.
7. **Bracket Pair Colorizer:** Colorizes matching brackets to improve code readability.
8. **Auto Close Tag / Auto Rename Tag:** Automatically closes HTML tags or renames them when the opening tag is edited.

Benefits of Using Extensions:

- **Increased Efficiency:** Extensions automate tasks and streamline workflows.
- **Enhanced Code Quality:** Linters and formatters help maintain consistent coding standards.
- **Extended Language Support:** Adds support for various programming languages and frameworks.
- **Customization:** Tailor VS Code to meet specific project requirements and personal preferences.

6)

Opening the Integrated Terminal:

1. **Opening the Terminal:**
 - Press **Ctrl+`** (Backtick) on Windows and Linux, or **Cmd+`** on macOS.
 - Alternatively, you can use the menu: **View > Terminal**.
2. **Navigating to a Specific Folder:**
 - By default, the integrated terminal opens in the root directory of your workspace.
To open it in a specific folder:
 - Right-click on the folder in the Explorer sidebar.
 - Select **Open in Terminal** from the context menu.

Using the Integrated Terminal:

Once the integrated terminal is open, you can use it just like any other terminal:

- **Navigation:** Use commands like **cd** to navigate between folders.
- **Running Commands:** Execute commands like **npm install**, **git pull**, **python script.py**, etc.
- **Running Scripts:** Start development servers (**npm start**), build scripts, or any command-line tool.

Advantages of Using the Integrated Terminal:

1. **Seamless Integration:**

- The terminal is directly integrated into the VS Code interface, eliminating the need to switch between multiple applications.
- This integration enhances workflow efficiency as you can code and execute commands in the same environment.

2. **Context Awareness:**

- The terminal automatically opens in the context of your current workspace or folder, making it easier to execute commands relevant to your project.

3. **Customizability:**

- You can customize the terminal's appearance, behavior, and settings (e.g., shell type, font size) through VS Code's settings ([settings.json](#)).

4. **Direct Access to Tools:**

- You can run development tools and scripts (e.g., compilers, linters, test runners) directly from the integrated terminal, leveraging VS Code's capabilities.

5. **Output Interactivity:**

- Some VS Code extensions (e.g., Debugger for Chrome) integrate with the terminal, allowing interactive debugging sessions and output interaction.

6. **Workspace Persistence:**

- The terminal retains its state (e.g., command history) across VS Code sessions, providing continuity in your workflow.

Comparison with External Terminals:

- **Efficiency:** Avoids context-switching and saves time by keeping everything within one interface.
- **Integration:** Works seamlessly with VS Code's editor and extensions.
- **Customization:** VS Code's terminal can be customized to match personal preferences and project requirements.
- **Productivity:** Streamlines development tasks and improves overall workflow management.

7)

Creating and Opening Files and Folders:

1. **Creating a New File or Folder:**

- To create a new file, use one of the following methods:
 - **Command Palette:** Open the Command Palette (**Ctrl+Shift+P** or **Cmd+Shift+P**) and type "New File". Select "File: New File" to create a new file in the current workspace.

- **Explorer Sidebar:** Right-click on the parent folder where you want to create the file, then select "New File".
 - To create a new folder, follow similar steps but choose "New Folder" instead.
- 2. **Opening Files and Folders:**
 - Double-click on a file in the Explorer sidebar to open it.
 - Use the Command Palette (**Ctrl+P** or **Cmd+P**) to quickly open a file by name. Type **Ctrl+P** and start typing the file name.
- 3. **Opening a Folder in VS Code:**
 - Use the Command Palette (**Ctrl+Shift+P** or **Cmd+Shift+P**) and type "Open Folder" to add an existing folder to your workspace.
 - Alternatively, drag and drop a folder into the VS Code window to open it.

Managing Files and Folders:

1. **Renaming and Deleting:**
 - **Renaming:** Right-click on a file or folder in the Explorer sidebar and choose "Rename".
 - **Deleting:** Right-click on a file or folder and select "Delete" (or press **Delete** on the keyboard).
2. **Moving and Copying:**
 - Use the Explorer sidebar to drag and drop files or folders to move them within the workspace.
 - To copy, hold down **Ctrl** (Windows/Linux) or **Cmd** (Mac) while dragging the file/folder to the destination.
3. **Searching and Filtering:**
 - Use the search bar at the top of the Explorer sidebar to quickly find files by name within the current workspace.
 - You can also filter files by type (e.g., All, Open Editors, Recent Files) using the dropdown menu in the Explorer sidebar.

Navigating Between Files and Directories Efficiently:

1. **Switching Between Open Files:**
 - Use **Ctrl+Tab** (Windows/Linux) or **Cmd+Tab** (Mac) to cycle through open files in the editor.
 - **Ctrl+P** (Windows/Linux) or **Cmd+P** (Mac) allows you to quickly open files by typing part of their name.
2. **Navigating Between Directories:**
 - In the Explorer sidebar, click on folders to expand or collapse them to navigate through the directory structure.


- Use **Alt+Left** (Windows/Linux) or **Cmd+[** (Mac) to navigate back to the previous directory you were viewing.
 - **Alt+Right** (Windows/Linux) or **Cmd+]** (Mac) navigates forward if you've navigated back.
3. **Switching Between Editor Groups:**
- If you have split your editor into multiple groups (**View > Editor Layout**), use **Ctrl+1**, **Ctrl+2**, etc. (Windows/Linux) or **Cmd+1**, **Cmd+2**, etc. (Mac) to switch between them.
4. **Using Bookmarks and Breadcrumbs:**
- Utilize VS Code's bookmarks feature (**Ctrl+F2** to toggle) to mark important locations in your codebase for quick navigation.
 - Enable breadcrumbs (**View > Show Breadcrumbs**) to navigate through files and symbols in a hierarchical view.

Tips for Efficiency:

- **Custom Keybindings:** Customize keybindings (**Preferences > Keyboard Shortcuts** or **Ctrl+K Ctrl+S**) to streamline navigation and file management tasks.
- **Useful Extensions:** Consider extensions like Project Manager for managing multiple projects, or Bookmarks for enhanced navigation within files.

8)

Accessing Settings:

1. **Using the Settings UI:**
- Click on the gear icon () in the lower-left corner of the Activity Bar to open the Settings.
 - Alternatively, press **Ctrl+,** (Windows/Linux) or **Cmd+,** (Mac) to open the Settings directly.
2. **Editing settings.json:**
- For more advanced customizations or to directly edit settings not available in the UI, you can edit the **settings.json** file.
 - Open the Command Palette (**Ctrl+Shift+P** or **Cmd+Shift+P**) and type "Preferences: Open Settings (JSON)" to open **settings.json**.

Examples of Customizations:

Changing the Theme:

1. **Using the Settings UI:**
- Go to **File > Preferences > Color Theme**.

- Select your preferred theme from the list (e.g., Dark+, Light+, Material Theme).

2. Editing **settings.json**:

Add or modify the following line in **settings.json**:

json

Copy code

```
"workbench.colorTheme": "Dark+"
```

○

Adjusting Font Size:

1. Using the Settings UI:

- Go to **File > Preferences > Settings**.
- Search for "Font Size" and adjust the "Editor: Font Size" setting.

2. Editing **settings.json**:

Add or modify the following line in **settings.json**:

json

Copy code

```
"editor.fontSize": 14
```

○

Customizing Keybindings:

1. Using the Settings UI:

- Go to **File > Preferences > Keyboard Shortcuts**.
- Search for the command you want to customize, then click on the pencil icon to edit the keybinding.

2. Editing **keybindings.json**:

- Open the Command Palette (**Ctrl+Shift+P** or **Cmd+Shift+P**) and type "Preferences: Open Keyboard Shortcuts (JSON)" to open **keybindings.json**.

Example of adding a custom keybinding:

json

Copy code

```
[
```

```
{
```

```
  "key": "ctrl+shift+f",
```

```

    "command": "editor.action.formatDocument",

    "when": "editorTextFocus"
  }
]

```

-

Additional Settings Customizations:

- **Editor Preferences:** Adjust various editor settings like indentation, word wrap, line height, etc.
- **File Associations:** Configure file associations and settings specific to file types.
- **Extensions:** Some extensions also provide settings that can be customized through the VS Code settings interface.

Tips for Effective Customization:

- **Use Search:** Utilize the search bar in the Settings UI (**Ctrl+F** or **Cmd+F**) to quickly find specific settings.
- **Compare with Default Settings:** Click on the **{ }** icon in the top-right corner of the Settings UI to compare your custom settings with the default ones.

9)

Setting Up and Starting Debugging:

1. Install Required Extensions:
 - Ensure you have the necessary debugging extensions installed for your programming language or framework. For example, for Node.js debugging, you would need the "Debugger for Chrome" or "Debugger for Node.js" extension.
2. Open Your Project in VS Code:
 - Open VS Code and navigate to the folder containing your project files (**File > Open Folder...**).
3. Configure Debugging:
 - Click on the Debugging icon in the Activity Bar (usually represented by a bug icon), or use the keyboard shortcut **Ctrl+Shift+D** (**Cmd+Shift+D** on Mac) to open the Debug view.
 - Click on the gear icon (⚙️) to create a **launch.json** file if it doesn't exist already. This file contains configurations for debugging.
4. Create or Select a Configuration:

- Click on the "create a launch.json file" link, then select your environment or runtime. For example, choose "Node.js" if you're debugging a Node.js application.
- 5. Edit `launch.json`:
 - VS Code generates a basic `launch.json` with default configurations. Customize it as needed based on your project setup (e.g., set the program path, arguments, environment variables).
- 6. Set Breakpoints:
 - In your code editor, click in the gutter next to the line numbers to set breakpoints. A red dot indicates a breakpoint where program execution will pause for inspection.
- 7. Start Debugging:
 - Press **F5** or click the green play button in the Debug view to start debugging. VS Code will launch your program in debugging mode, and execution will pause at the first breakpoint encountered.

Key Debugging Features in VS Code:

1. Step Through Code:
 - Use controls like **F10** (Step Over), **F11** (Step Into), and **Shift+F11** (Step Out) to navigate through code line-by-line, stepping over or into function calls.
2. Watch and Variables Panel:
 - View and monitor the values of variables and expressions in real-time using the Variables panel. Add variables to the Watch panel to track specific values.
3. Call Stack:
 - Visualize the call stack to understand the hierarchy of function calls and where in the program execution you currently are.
4. Debug Console:
 - Use the Debug Console to interactively execute code snippets and evaluate expressions during debugging sessions.
5. Conditional Breakpoints:
 - Set breakpoints with conditions (**Right-click > Add Conditional Breakpoint**) that cause the program to pause only when certain conditions are met.
6. Exception Handling:
 - Configure how VS Code handles exceptions (**uncaughtException**, **caughtException**) by modifying settings in `launch.json`.
7. Debugging Tasks:
 - Execute tasks (e.g., build, test) in debugging sessions by defining them in `tasks.json` and configuring them in `launch.json`.
8. Debugging Across Environments:

- Debug applications running locally, remotely (via SSH), or in containers, leveraging VS Code's remote debugging capabilities.