1. Installation of VS Code

This guide will walk you through the steps to download and install Visual Studio Code (VS Code) on a Windows 11 operating system, including any prerequisites.

Before you start, ensure that your system meets the following prerequisites:

1. Windows 11 Operating System: Ensure you have Windows 11 installed on your machine.

2. Good Internet Connection: An active internet connection to download VS Code and its components.

Steps to Download and Install Visual Studio Code

Step 1: Download Visual Studio Code

1. Visit the Official Website:

Open your web browser and go to the [official Visual Studio Code download page](https://code.visualstudio.com/).

2. Choose Your OS:

On the download page, you will see options for different operating systems. Click on the "Windows" button to download the installer for Windows.

Step 2: Run the Installer

1. Locate the Downloaded File:

Once the download is complete, navigate to the folder where the installer file (usually named `VSCodeSetup-x64-<version>.exe`) is saved.

2. Run the Installer:

Double-click the installer file to start the installation process. If prompted by the User Account Control (UAC), click "Yes" to allow the installer to make changes to your device.

Step 3: Install Visual Studio Code

1. Setup Wizard:

The setup wizard will open. Click "Next" to continue.

2. Accept the License Agreement:

Read the license agreement. If you agree, select "I accept the agreement" and click "Next".

3. Select Installation Location:

Choose the destination folder where you want to install VS Code. The default location is usually fine. Click "Next".

4. Select Additional Tasks:

Choose additional tasks you want the installer to perform, such as:

- Creating a desktop icon.

- Adding "Open with Code" actions to the Windows Explorer context menu.

- Registering VS Code as an editor for supported file types.

- Adding VS Code to the PATH environment variable.

Select the tasks you want and click "Next".

5. Ready to Install:

Review your setup choices. If everything looks correct, click "Install" to start the installation.

6. Finish Installation:

Once the installation is complete, you will see a final screen. You can choose to launch Visual Studio Code immediately by checking the "Launch Visual Studio Code" box. Click "Finish" to exit the installer.

Step 4: Launch Visual Studio Code

1. Open Visual Studio Code:

- If you chose to launch VS Code immediately during the installation, it will open automatically.

- Otherwise, you can open it from the Start Menu by searching for "Visual Studio Code" or using the desktop icon if you created one.

Step 5: Initial Setup

1. Welcome Screen:

The first time you launch VS Code, you will see a Welcome screen. From here, you can start a new file, open a folder, or take a tour of the editor.

2 First-Time Setup

After installing Visual Studio Code (VS Code), there are several initial configurations and settings that you should adjust to create an optimal coding environment. Here are the key settings and recommended extensions:

Initial Configurations

1. User and Workspace Settings

1. Open Settings:

- Go to `File` > `Preferences` > `Settings` or use the keyboard shortcut `Ctrl+,` (Windows/Linux) or `Cmd+,` (macOS).

2. Common Settings to Adjust:

- Font Size: Increase or decrease the font size for better readability.
- Tab Size: Set the number of spaces for a tab.
- Word Wrap: Enable or disable word wrap.
- Line Numbers: Show or hide line numbers.
- Auto Save: Automatically save files after a delay or when the editor loses focus.
- Format on Save: Automatically format your code when you save a file.

2. Theme and Icon Pack

1. Theme:

- Change the color theme to your preference by going to `File` > `Preferences` > `Color Theme` or use the shortcut `Ctrl+K Ctrl+T`.
- Popular themes: "Dark+ (default dark)", "Light+ (default light)", or install new themes like "One Dark Pro", "Dracula", etc.

2. File Icon Theme:

- Change the file icon theme by going to `File` > `Preferences` > `File Icon Theme`.
- Recommended: "Material Icon Theme".

3. Keyboard Shortcuts

1. Customize Shortcuts:

- Go to `File` > `Preferences` > `Keyboard Shortcuts` or use the shortcut `Ctrl+K Ctrl+S`.
- Customize shortcuts to match your workflow.

Recommended Extensions

1. Language Support:

- Python: `ms-python.python`
- JavaScript/TypeScript: `esbenp.prettier-vscode`, `dbaeumer.vscode-eslint`
- HTML/CSS: `ritwickdey.LiveServer`, `ecmel.vscode-html-css`

2. Version Control:

- Git: `eamodio.gitlens` for enhanced Git capabilities.

3. Code Formatting:

- Prettier: `esbenp.prettier-vscode` for consistent code formatting.
- ESLint: `dbaeumer.vscode-eslint` for JavaScript/TypeScript linting.

4. Productivity:

- Live Server: `ritwickdey.LiveServer` to launch a local development server with live reload.
- Path Intellisense: `christian-kohler.path-intellisense` for auto-completing file paths.
- Bracket Pair Colorizer: `CoenraadS.bracket-pair-colorizer-2` for colorizing matching brackets.

5. Snippets:

- JavaScript (ES6) code snippets: `xabikos.JavaScriptSnippets`

- Python Snippets: `njpwerner.autodocstring` for generating docstrings.

Initial Setup of Extensions

Python Extension Setup

1. Install Python Extension:

- Search for `ms-python.python` in the Extensions view (`Ctrl+Shift+X`) and install it.

2. Configure Python Path:

- Open a Python file to activate the extension.

- Set the Python interpreter by clicking on the Python version in the status bar or using the command `Python: Select Interpreter` from the Command Palette (`Ctrl+Shift+P`).

Prettier Setup for JavaScript/TypeScript

1. Install Prettier Extension:

- Search for `esbenp.prettier-vscode` in the Extensions view and install it.

2. Configure Prettier:

- Set Prettier as the default formatter by adding the following to your settings

ESLint Setup for JavaScript/TypeScript

1. Install ESLint Extension:

- Search for `dbaeumer.vscode-eslint` in the Extensions view and install it.

2. Configure ESLint:

- Create an `.eslintrc.json` file in your project root with your configuration.

- Add the following to your settings to enable ESLint

Conclusion

With these configurations and extensions, you should have an optimal coding environment in Visual Studio Code. This setup will help improve productivity, code quality, and ease of use.

3 VS Code User Interface

Visual Studio Code (VS Code) has a user interface (UI) that is designed to be intuitive and efficient for coding. Here are the main components of the VS Code UI, including their purpose and functionality:

Main Components of the VS Code User Interface

1. Activity Bar

The Activity Bar is a vertical bar on the far-left side of the VS Code window. It provides access to different views and functionalities within the editor.

Purpose:

- Navigation: Quickly switch between different views such as Explorer, Search, Source Control, Run & Debug, and Extensions.

- Extensions: Icons for other installed extensions that add additional views can also appear here.

Components:

- Explorer: View and manage files and folders in your workspace.

- Search: Search across files in your workspace.

- Source Control: Manage source control (Git) operations.

- Run & Debug: Access debugging configurations and controls.

- Extensions: Browse and manage installed extensions.

2. Side Bar

The Side Bar is located next to the Activity Bar and displays the contents and functionality of the selected view from the Activity Bar.

Purpose:

- Context-Specific Functions: The Side Bar changes content based on the selected Activity Bar icon. For example, if the Explorer icon is selected, the Side Bar will show the file and folder structure of the workspace.

- Detail Management: Provides detailed views and management options for files, search results, source control changes, and installed extensions.

Examples:

- Explorer View: Displays a tree view of the workspace folders and files.

- Search View: Displays search results and allows for find/replace operations.

- Source Control View: Shows pending changes, commit history, and provides Git operations.

- Extensions View: Lists installed extensions and allows you to search for and install new extensions.

3. Editor Group

Description:

The Editor Group is the central area of the VS Code window where you edit your code. You can have multiple editors open side by side in groups.

Purpose:

- Code Editing: The primary area for writing and editing code.

- Multi-Editor Layouts: Supports multiple editor groups for side-by-side editing.

- Tabs: Each open file is represented by a tab within the editor group.

Features:

- Syntax Highlighting: Color-coding based on the language.

- IntelliSense: Code completion and suggestions.

- Code Linting: Error and warning indicators in the code.

- Split View: Allows splitting the editor into multiple groups horizontally or vertically for comparing or working on multiple files simultaneously.

4. Status Bar

Description:

The Status Bar is a horizontal bar at the bottom of the VS Code window. It provides information about the current workspace, file, and editor status.

Purpose:

- Status Indicators: Displays the current state of the editor and workspace, such as line number, column number, language mode, and encoding.

- Quick Access: Provides quick access to commonly used features and settings.

- Extension Integrations: Extensions can add their own status indicators and controls here.

Components:

- Position Info: Current line and column number.

- Language Mode: Shows the language of the currently active file and allows you to change it.

- Encoding: Displays the file encoding.

- EOL (End of Line): Shows the current end-of-line character (CRLF or LF).

- Git Info: Branch name and status of the repository if the workspace is under source control.

- Problems: Quick access to the problems panel showing errors and warnings.

Summary

- Activity Bar: Provides quick access to different views and functionalities.

- Side Bar: Displays the content and functionalities of the selected view from the Activity Bar.

- Editor Group: The central area for writing and editing code, supporting multiple open files and split views.

- Status Bar: Shows status information about the current workspace and file, and provides quick access to commonly used features and settings.

Understanding these components and how to use them effectively can significantly enhance your productivity and experience while using Visual Studio Code.

4 Command Palette

The Command Palette in Visual Studio Code (VS Code) is a powerful feature that provides quick access to a wide range of commands and settings. It allows users to perform various tasks without having to navigate through menus or remember keyboard shortcuts.

Accessing the Command Palette

The Command Palette can be accessed in two ways:

1. Keyboard Shortcut: Press `Ctrl+Shift+P`

2. Menu: Go to `View` > `Command Palette...`.

When the Command Palette is open, you can start typing to search for and execute commands.

Common Tasks Performed Using the Command Palette

Here are some examples of common tasks that can be performed using the Command Palette:

1. Opening Files

- Command: `Open File...`

-Description: Quickly open a file from your workspace.

- Usage: Type `Open File` and select the file you want to open from the list.

2. Searching for Files

- Command: `Go to File...` or `Quick Open`

- Description: Search for and open files quickly by their name.

- Usage: Type `>` followed by the name of the file.

3. Running and Debugging Code

- Command: `Run Without Debugging`, `Start Debugging`

- Description: Run your code without debugging or start a debugging session.

- Usage: Type `Run Without Debugging` or `Start Debugging`.

4. Git Commands

- Command: `Git: Commit`, `Git: Push`, `Git: Pull`

- Description: Perform Git operations such as committing changes, pushing to a remote repository, and pulling updates.

- Usage: Type `Git: Commit` to commit changes, `Git: Push` to push changes, or `Git: Pull` to pull changes.

5. Installing Extensions

- Command: `Extensions: Install Extensions`

- Description: Search for and install new extensions.

- Usage: Type `Extensions: Install Extensions` and search for the extension you want to install.

6. Formatting Code

- Command: `Format Document`

- Description: Format the entire document according to the language-specific formatter.

- Usage: Type `Format Document`.

7. Changing Settings

- Command: `Preferences: Open Settings`

- Description: Open the settings editor to change user or workspace settings.

- Usage: Type `Preferences: Open Settings`.

8. Navigating to Symbols

- Command: `Go to Symbol in File...`, `Go to Symbol in Workspace...`

- Description: Navigate to symbols (functions, variables, etc.) within the current file or workspace.

- Usage: Type `@` for symbols in the current file, `#` for symbols in the workspace.

9. Switching Language Mode

- Command: `Change Language Mode`

- Description: Change the language mode of the current file.

- Usage: Type `Change Language Mode` and select the desired language.

10. Viewing Problems and Output

- Command: `View: Toggle Problems`, `View: Toggle Output`

- Description: Toggle the Problems or Output panel.

- Usage: Type `View: Toggle Problems` or `View: Toggle Output`.

Example Usage

To format a document:

1. Press `Ctrl+Shift+P`.

2. Type `Format Document`.

3. Select the `Format Document` command from the list.

To commit changes using Git:

1. Press `Ctrl+Shift+P`

2. Type `Git: Commit`.

3. Follow the prompts to commit your changes.

5 Extensions in VS Code

Extensions play a crucial role in enhancing the functionality and productivity of Visual Studio Code (VS Code). They provide additional features, integrations, and tools that cater to various development needs. Here's an overview of how users can find, install, and manage extensions in VS Code, along with examples of essential extensions for web development.

Finding, Installing, and Managing Extensions

Finding Extensions

1. Extensions View:

- Open the Extensions view by clicking the Extensions icon in the Activity Bar on the side of the window or pressing `Ctrl+Shift+X`.

2. Search for Extensions:

- In the Extensions view, use the search bar at the top to find extensions by name, keyword, or category.

3. Browse the Marketplace:

- You can also visit the [Visual Studio Code Marketplace](https://marketplace.visualstudio.com/VSCode) to browse and search for extensions.

Installing Extensions

1. Install via Extensions View:

- Once you find an extension, click the `Install` button. The extension will be downloaded and installed automatically.

2. Install via Command Palette:

- Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS) to open the Command Palette.

- Type `Extensions: Install Extensions` and select it to open the Extensions view, where you can search and install extensions.

Managing Extensions

1. Enable/Disable Extensions:

- In the Extensions view, installed extensions are listed. You can enable or disable an extension by clicking the gear icon next to it and selecting `Enable` or `Disable`.

2. Update Extensions:

- Extensions are automatically updated, but you can manually check for updates by clicking the gear icon next to the extension and selecting `Check for Updates`.

3. Uninstall Extensions:

- To uninstall an extension, click the gear icon next to the extension and select `Uninstall`.

4. View Extension Settings:

- Some extensions have customizable settings. Click the gear icon and select `Extension Settings` to view and configure these settings.

Essential Extensions for Web Development

1. Prettier - Code Formatter

2. ESLint

3. Live Server

4. Debugger for Chrome

5. Visual Studio IntelliCode

6. GitLens

7. HTML Snippets

8. CSS Peek

9. Path Intellisense

10. REST Client

6 Integrated Terminal

Using the Integrated Terminal in VS Code

Opening the Integrated Terminal

There are several ways to open the integrated terminal in Visual Studio Code:

1. Keyboard Shortcut:

   - Press `Ctrl+` (Windows/Linux) or `Cmd+` (macOS).

2. Menu:

   - Go to `View` > `Terminal`.

3. Command Palette:

   - Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS) to open the Command Palette, then type `Terminal: Create New Integrated Terminal` and select it.

Using the Integrated Terminal

Once the terminal is open, you can perform various tasks just as you would in an external terminal. Here are some common actions:

1. Create a New Terminal:

   - Click the plus icon (`+`) in the terminal panel or press `Ctrl+Shift+` (Windows/Linux) or `Cmd+Shift+` (macOS).

2. Switch Between Terminals:

   - Click the dropdown arrow next to the terminal tabs to select a different terminal session.

   - Use the keyboard shortcut `Ctrl+Page Down` and `Ctrl+Page Up` (Windows/Linux) or `Cmd+Page Down` and `Cmd+Page Up` (macOS) to navigate between terminals.

3. Split Terminal:

   - Click the split terminal icon to create a side-by-side terminal session within the same panel.

   - Use the command `Terminal: Split Terminal` from the Command Palette.

4. Resize Terminal Panel:

   - Drag the top edge of the terminal panel to adjust its height.

5. Run Commands:

- Type your commands as you would in any terminal. For example, you can navigate directories with `cd`, list files with `ls` (or `dir` on Windows), run scripts, manage version control with Git, etc.

Advantages of Using the Integrated Terminal

1. Convenience and Efficiency:

- The integrated terminal is embedded within VS Code, reducing the need to switch between the editor and an external terminal. This streamlines the workflow, allowing for faster and more efficient coding and debugging.

2. Context Awareness:

- The integrated terminal opens in the context of your workspace or project directory by default. This means commands you run in the terminal are relative to the workspace, making it easier to manage project-specific tasks without additional navigation.

3. Synchronization with Editor:

- The terminal is synchronized with the VS Code workspace, so tasks like running tests, compiling code, and committing changes can be directly tied to the files you are working on. You can easily view errors or output in the terminal while making changes in the editor.

4. Customization:

- VS Code allows extensive customization of the terminal, such as changing the shell (e.g., Bash, PowerShell, Command Prompt), adjusting appearance settings (e.g., font size, color scheme), and configuring startup scripts.

5. Multiple Terminals:

- You can have multiple terminal sessions open simultaneously, each with its own shell and environment. This is useful for running different tasks in parallel, such as running a development server in one terminal and a build process in another.

6. Integration with Extension:

- Many extensions in VS Code can interact with the terminal, automating tasks like running tests, linting code, or deploying applications. This integration further enhances productivity and streamlines workflows.

Example Usage

1. Running a Development Server:

- Open the integrated terminal.

- Navigate to your project directory using `cd`.

- Run the command to start your development server (e.g., `npm start`, `python manage.py runserver`, etc.).

- The server output will be displayed in the terminal, and you can continue to code in the editor.

2. Version Control with Git:

- Open the integrated terminal.

- Use Git commands like `git status`, `git add`, `git commit`, and `git push` to manage your repository.

- The terminal allows you to handle version control without leaving the VS Code environment.

Conclusion

The integrated terminal in VS Code offers significant advantages over using an external terminal, including convenience, context awareness, synchronization with the editor, customization, multiple terminal sessions, and integration with extensions. These features help streamline development workflows and enhance productivity.

7 File and Folder Managaement

Managing files and folders in Visual Studio Code (VS Code) is essential for efficient development. Here's a comprehensive guide on how to create, open, and manage files and folders in VS Code, as well as how to navigate between different files and directories efficiently.

Creating Files and Folders

Creating a New File

1. From the Explorer Sidebar:

- Open the Explorer sidebar by clicking on the Explorer icon in the Activity Bar or pressing `Ctrl+Shift+E`.

- Click on the "New File" icon (`+`) at the top of the Explorer view or right-click in the Explorer and select `New File`.

- Enter the file name and press `Enter`.

2. Using the Command Palette:

- Press `Ctrl+Shift+P` to open the Command Palette.

- Type `New File` and select `File: New File`.

- Save the new file by giving it a name and choosing the location.

3. Using Keyboard Shortcuts:

- Press `Ctrl+N` to create a new untitled file.

- Save it by pressing `Ctrl+S`, giving it a name, and choosing the location.

Creating a New Folder

1. From the Explorer Sidebar:

- Open the Explorer sidebar by clicking on the Explorer icon in the Activity Bar or pressing `Ctrl+Shift+E`.

- Click on the "New Folder" icon (`+`) at the top of the Explorer view or right-click in the Explorer and select `New Folder`.

- Enter the folder name and press `Enter`.

Opening Files and Folders

Opening a Single File

1. From the Explorer Sidebar:

- Click on the file in the Explorer sidebar to open it in the editor.

2. Using the Command Palette:

- Press `Ctrl+Shift+P` to open the Command Palette.

- Type `Open File` and select `File: Open File...`.

- Browse to the file and open it.

3. Using the File Menu:

- Go to `File` > `Open File...` and browse to the file.

4. Using Quick Open:

- Press `Ctrl+P` (Windows/Linux) or `Cmd+P` (macOS) to open Quick Open.

- Start typing the name of the file and select it from the list.

Opening a Folder or Workspace

1. From the File Menu:

   - Go to `File` > `Open Folder...` and browse to the folder you want to open.

   - To open a workspace, go to `File` > `Open Workspace...`.

2. Using the Command Palette:

   - Press `Ctrl+Shift+P` to open the Command Palette.

   - Type `Open Folder` or `Open Workspace` and select the appropriate command.

Managing Files and Folders

Renaming Files and Folders

1. From the Explorer Sidebar:

   - Right-click on the file or folder and select `Rename`.

   - Enter the new name and press `Enter`.

Moving Files and Folders

1. Drag and Drop:

   - Drag the file or folder to the new location within the Explorer sidebar.

2. Cut and Paste:

   - Right-click on the file or folder, select `Cut`, navigate to the new location, right-click, and select `Paste`.

Deleting Files and Folders

1. From the Explorer Sidebar:

- Right-click on the file or folder and select `Delete`.

- Confirm the deletion if prompted.

Navigating Between Files and Directories Efficiently

Quick Open

1. Using Quick Open:

- Press `Ctrl+P` (Windows/Linux) or `Cmd+P` (macOS).

- Start typing the name of the file, and select it from the list.

File Explorer

1. Explorer Sidebar:

- Use the Explorer sidebar to navigate through your project's directory structure.

- Collapse and expand folders by clicking the arrow icons.

Go to Definition and References

1. Using Go to Definition:

- Place the cursor on a symbol (function, variable, etc.) and press `F12` or right-click and select `Go to Definition`.

2. Using Peek Definition:

- Place the cursor on a symbol and press `Alt+F12` or right-click and select `Peek Definition`.

3. Find All References:

- Place the cursor on a symbol and press `Shift+F12` or right-click and select `Find All References`.

Navigation History

1. Back and Forward:

- Navigate back and forward through your editing history with `Alt+Left Arrow` and `Alt+Right Arrow`.

Breadcrumbs

1. Enable Breadcrumbs:

- Click the breadcrumb icon at the top of the editor or use the keyboard shortcut `Ctrl+Shift+.`.

- Navigate through symbols and files using the breadcrumbs bar.

Split Editor

1. Split Editor:

- To open a file in a new editor group, right-click on the file and select `Split Right` or use the keyboard shortcut `Ctrl+\`.

2. Move Between Split Editors:

- Use `Ctrl+1`, `Ctrl+2`, etc., to switch focus between editor groups.

Terminal and File Navigation

1. Integrated Terminal:

- Use the integrated terminal to navigate and manage files with terminal commands.

- Open the terminal with `Ctrl+` and use commands like `cd`, `ls`, `mkdir`, `touch`, etc.

Extensions

1. File Navigation Extensions:

- Install extensions like `Path Intellisense`, `File Utils`, or `Project Manager` to enhance file and directory navigation.

By utilizing these features and shortcuts, you can efficiently manage and navigate through files and folders in VS Code, significantly enhancing your productivity and workflow.

8 Settings and Preferences

Users can find and customize settings in Visual Studio Code (VS Code) through various interfaces: the Settings UI, the settings.json file, and the Keybindings Editor. Here's a detailed guide on how to change the theme, font size, and keybindings.

Accessing Settings

Settings UI

1. Open Settings UI:

   - Click on the gear icon in the lower left corner of the VS Code window and select `Settings`.

   - Alternatively, go to `File` > `Preferences` > `Settings` or press `Ctrl+,` (Windows/Linux) or `Cmd+,` (macOS).

Settings JSON

1. Open settings.json:

   - In the Settings UI, click the `{}` icon in the top right corner to open the `settings.json` file directly.

   - Alternatively, open the Command Palette with `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS), then type `Preferences: Open Settings (JSON)` and select it.

Keybindings Editor

1. Open Keybindings Editor:

   - Go to `File` > `Preferences` > `Keyboard Shortcuts` or press `Ctrl+K Ctrl+S`.

Changing the Theme

Using Settings UI

1. Open the Command Palette:

   - Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS).

2. Select Theme:

   - Type `Color Theme` and select `Preferences: Color Theme`.

   - Browse and select the desired theme from the list.

Using settings.json

1. Open settings.json:

   - Follow the steps mentioned above to open the `settings.json` file.

2. Set the Theme:

   - Add or modify the following line in `settings.json`:

   - Replace `"Theme Name"` with the name of the desired theme (e.g., `"Dark+ (default dark)"`).

Changing the Font Size

Using Settings UI

1. Open Settings UI:

   - Go to `File` > `Preferences` > `Settings` or press `Ctrl+,`.

2. Search for Font Size:

   - Type `Font Size` in the search bar at the top.

3. Set the Font Size:

   - Adjust the `Editor: Font Size` setting to the desired value.

Using settings.json

1. Open settings.json:

   - Follow the steps mentioned above to open the `settings.json` file.

2. Set the Font Size:

   - Add or modify the following line in `settings.json`:

   - Replace `16` with the desired font size.

Changing Keybindings

Using Keybindings UI

1. Open Keybindings Editor:

   - Go to `File` > `Preferences` > `Keyboard Shortcuts` or press `Ctrl+K Ctrl+S`.

2. Search for a Command:

   - In the Keybindings Editor, search for the command you want to change.

3. Change the Keybinding:

- Click the pencil icon next to the command and press the desired key combination.

- To remove a keybinding, click the "Remove Keybinding" icon (trash can).

Using keybindings.json

1. Open keybindings.json:

- In the Keybindings Editor, click the link `keybindings.json` in the top right corner.

2. Set the Keybinding:

- Add a keybinding entry in `keybindings.json`. For example:

- This example binds `Ctrl+Alt+N` to the `New Untitled File` command. Modify the `key` and `command` as needed.

Summary

- Settings UI: User-friendly interface for changing settings.

- settings.json: Direct JSON file for manual edits and advanced configurations.

- Keybindings Editor: GUI for changing keyboard shortcuts.

- keybindings.json: JSON file for manual keybinding configuration.

These methods allow users to customize VS Code to their liking, improving their productivity and coding experience.

9 Debugging in VS Code

Setting up and starting debugging in Visual Studio Code (VS Code) involves a few straightforward steps. Here’s how to do it, along with an overview of key debugging features.

Setting Up and Starting Debugging in VS Code

1. Install Required Extensions

Depending on the programming language you're using, you might need to install a specific extension. For example:

- Python: Install the Python extension.

- JavaScript/Node.js: The JavaScript/TypeScript language support is built-in.

- C/C++: Install the C/C++ extension.

2. Open Your Project

Open the folder containing your project files:

1. Go to `File` > `Open Folder...` and select your project folder.

3. Configure the Debugger

Create a launch configuration file (`launch.json`):

1. Open the Command Palette with `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS).

2. Type `Debug: Open launch.json` and select it.

3. If you don't have a `launch.json` file yet, VS Code will prompt you to select a debug environment. Choose the appropriate environment for your language.

4. A default `launch.json` file will be created in a `.vscode` folder in your project directory. Customize it as needed. Here’s an example configuration for a Node.js application:

4. Set Breakpoints

Set breakpoints in your code where you want the debugger to pause execution:

1. Open the file you want to debug.

2. Click in the gutter to the left of the line numbers or press `F9` to toggle a breakpoint on the current line.

5. Start Debugging

1. Open the Debug view by clicking the Debug icon in the Activity Bar on the side of the window or pressing `Ctrl+Shift+D`.

2. Click the green play button at the top of the Debug panel or press `F5` to start debugging.

Key Debugging Features in VS Code

Breakpoints

- Toggle Breakpoint: Set or remove breakpoints by clicking in the gutter or pressing `F9`.

- Conditional Breakpoints: Right-click on a breakpoint and select "Edit Breakpoint" to add a condition, hit count, or log message.

Step Controls

- Continue (`F5`): Resume program execution until the next breakpoint or end of the program.

- Step Over (`F10`): Execute the next line of code, stepping over any function calls.

- Step Into (`F11`): Step into the function call at the current line.

- Step Out (`Shift+F11`): Step out of the current function and return to the calling function.

Variables

- Watch: Monitor specific variables by adding them to the Watch panel.

- Variables Pane: View local variables, function arguments, and global variables.

- Hover: Hover over variables in the code to see their current value.

Call Stack

- Call Stack Pane: View the call stack of the current execution, which shows the nested function calls.

- Jump to Frame: Click on a stack frame to navigate to that point in the code.

Debug Console

- Evaluate Expressions: Use the Debug Console to run arbitrary expressions and inspect values in the current execution context.

Exception Handling

- Break on Exceptions: Configure the debugger to break when exceptions are thrown. This can be set to all exceptions or uncaught exceptions only.

Integrated Terminal

- Debugging Tasks: Use the integrated terminal to run tasks or additional debugging commands without leaving the editor.

Example: Debugging a Python Program

1. Install the Python Extension: If not already installed, go to the Extensions view (`Ctrl+Shift+X`) and install the Python extension.

2. Open Your Python Project: Go to `File` > `Open Folder...` and select your project folder.

3. Create launch.json:

   - Open the Command Palette with `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS).

   - Type `Debug: Open launch.json` and select it.

   - Choose "Python" when prompted.

   - A default `launch.json` will be created. Customize the `program` attribute to point to your Python script.

4. Set Breakpoints: Open your Python script and click in the gutter to set breakpoints.

5. Start Debugging: Press `F5` to start debugging.

By following these steps and utilizing the key debugging features, you can efficiently debug your code within VS Code, improving your development workflow and productivity.

10 Using Source Control

Here’s a step-by-step guide on how to initialize a repository, make commits, and push changes to GitHub.

Integrating Git with VS Code

Prerequisites

1. Install Git: Ensure Git is installed on your system. You can download it from git-scm.com.

2. GitHub Account: Create a GitHub account if you don’t have one.

Initializing a Repository

1. Open Your Project in VS Code:

- Open VS Code and load your project by going to `File` > `Open Folder...` and selecting your project folder.

2. Open the Source Control View:

- Click on the Source Control icon in the Activity Bar on the side of the window or press `Ctrl+Shift+G`.

3. Initialize Git Repository:

- In the Source Control view, click on the `Initialize Repository` button. This will create a `.git` directory in your project folder, making it a Git repository.

Making Commits

1. Stage Changes:

- Changes to your files will appear in the Source Control view under the `Changes` section.

- To stage changes, click the `+` icon next to each file. Alternatively, you can stage all changes by clicking the `+` icon in the `Changes` section header.

2. Commit Changes:

- Once your changes are staged, enter a commit message in the input box at the top of the Source Control view.

- Click the checkmark icon or press `Ctrl+Enter` to commit the changes.

Pushing Changes to GitHub

1. Create a Repository on GitHub:

- Go to GitHub and create a new repository by clicking the `New` button.

- Fill in the repository details and click `Create repository`.

2. Add GitHub Repository as Remote:

- Copy the repository URL from GitHub.

- In VS Code, open the integrated terminal by pressing `Ctrl+` or going to `View` > `Terminal`.

- Add the remote repository by running the following command, replacing `<URL>` with your GitHub repository URL:

3. Push Changes to GitHub:

- In the terminal, push your commits to GitHub by running:

- If your default branch is not `main`, replace `main` with the name of your default branch (e.g., `master`).

Key Git Features in VS Code

- Source Control View: Provides an overview of your repository status, including changes, commits, and branches.

- Integrated Terminal: Allows you to run Git commands directly within VS Code.

- Git Lens Extension: Enhances Git integration with features like blame annotations, code lens, and repository explorer.

Example Workflow

1. Initialize Repository:

- Open your project in VS Code.

- Click the Source Control icon and click `Initialize Repository`.

2. Make Changes and Commit:

- Edit your files as needed.

- Stage changes by clicking the `+` icon next to the files.

- Enter a commit message and press `Ctrl+Enter`.

3. Push to GitHub:

- Create a repository on GitHub.

- Add the GitHub repository as a remote using the terminal

- Push your changes

By following these steps, you can efficiently use Git for version control in VS Code and manage your project's source code with ease.

RESEARCH APPROACH

Google

W3Schools

StackOverflow