# INSTALLING VS CODE

**Prerequisites:**

* **Internet connection:** You'll need an internet connection to download the installer.
* **64-bit Windows 11:** VS Code is a 64-bit application and requires a 64-bit version of Windows 11 to run.

**Download and Installation Steps:**

1. **Visit the download page:** Open a web browser and navigate to the official VS Code download page.
2. **Download the installer:** On the download page, locate the download section for Windows. Choose the option for "Windows" (64-bit). Then run the installer; double-click the downloaded ".exe" file to launch the installer. Read the license agreement and click "I Agree" to proceed. **Installation Location:** The installer suggests a default installation location. You can keep this default or choose a different location by clicking "Change" and specifying your preferred directory.
3. **Additional Tasks:** The installer offers options to create a desktop icon and add VS Code to the PATH environment variable. You can choose these options based on your preference.
4. **Install:** Click "Install" to begin the installation process. This might take a few minutes depending on your internet speed. Once the installation finishes, you can choose to launch VS Code directly from the installed application.

# Configurations and settings after installing vs code

**interface Customization:**

* **Theme:** VS Code offers a variety of built-in themes and supports installing custom themes from the extensions marketplace.
* **Font Size and Style:** Adjust the font size and style to your preference for better readability. You can find these settings under **File > Preferences > Settings**
* **Workbench Layout:** VS Code allows customizing the layout of the editor panels. You can experiment with different layouts to find what suits your workflow best. Drag and drop panels or use the layout options under the **View** menu.

**Settings for Efficiency:**

* **Auto Save:** Enable auto-save (**File > Preferences > Settings > Files > Auto Save**).
* **Keyboard Shortcuts:** VS Code comes with a rich set of default keyboard shortcuts. Explore them (**Help > Keyboard Shortcuts**) . You can also install extensions that provide keyboard shortcuts specific to programming languages you use.
* **Code Formatting:** Enable automatic code formatting on save (**Settings > Editor > Formatting > On Save**). This helps maintain consistent code style and improves readability. You can further customize formatting rules based on your preferred coding style.

**Essential Extensions**:

* **Language-specific extensions:** Install extensions for the programming languages you use. These extensions provide syntax highlighting, code completion, debugging tools, and linters (tools that identify potential errors) specific to those languages. Some popular examples include Python (Pylance), C++ (C/C++), and Java (Java Extension Pack).
* **Code Runner:** This extension allows you to execute code snippets directly within VS Code, streamlining your development workflow.
* **GitLens:** This extension provides Git integration features within VS Code, making it easier to manage your code versions and collaborate with others.

# User interface overview

**1. Activity Bar**

* **Purpose:** The Activity Bar provides quick access to different views and functionalities within VS Code.
* **Components:**
  + **Explorer:** Manage your project files and folders.
  + **Search:** Search for files and text across your project.
  + **Source Control (Git):** Manage your code versions using Git (if integrated).
  + **Run and Debug:** Run and debug your code.
  + **Extensions:** Access and manage installed extensions.
  + **Custom views:** Views contributed by extensions you've installed.
* **Customization:** You can reorder the icons in the Activity Bar to prioritize the views you use most frequently.

**2. Side Bar:**

* **Purpose:** It's a dynamic panel that adapts to your needs.
* **Components:** The content of the Side Bar varies based on the context. For example:
  + **Explorer view:** When the Explorer view is active, the Side Bar might show file properties or a preview of the selected file.
  + **Search view:** The Side Bar might display search filters and refinement options.
  + **Git view:** The Side Bar might show details about code changes and commit history.
* **Visibility:** You can show or hide the Side Bar using the **View > Appearance > Toggle Side Bar** menu option.

**3. Editor Group:**

* **Purpose:** This is where you write, edit, and view your code.
* **Components:**
  + **Editor:** Each editor tab represents a single file you're working on. You can open multiple files simultaneously and arrange them in tabs or split the editor area vertically or horizontally.
  + **Line Numbers:** Lines are numbered on the left side of the editor for easy reference.
  + **Scroll Bars:** Vertical and horizontal scroll bars allow you to navigate through long files.
* **Functionality:** The editor area provides features like syntax highlighting, code completion, debugging tools, and integration with various programming languages.

**4. Status Bar (Bottom Bar):**

* **Purpose:** The Status Bar displays contextual information about your current workspace and project.
* **Components:** It typically shows:
  + **Current File Information:** The name of the currently open file and its position within your project directory.
  + **Selection Information:** Details about the selected text in the editor, such as line and column number.
  + **Language Mode:** The programming language the currently open file is recognized as.
  + **Git Status:** If Git is integrated, the Status Bar might display information about uncommitted changes or the current branch.
  + **Encoding:** The character encoding used by the current file.
* **Functionality:** The Status Bar can also contain buttons for specific actions depending on the context. For example, it might offer options to format code or manage indentation.

# Command pallete

The Command Palette in VS Code is a powerful tool that acts as a central hub for accessing all functionalities and commands available within the editor. It allows you to quickly execute actions without needing to navigate through menus or memorize keyboard shortcuts.

**Accessing the Command Palette:**

There are three ways to access the Command Palette:

* **Keyboard Shortcut:** The most common way is by pressing Ctrl+Shift+P
* **Menu:** Navigate to the **View > Command Palette** menu option.
* **Search Bar:** Click on the magnifying glass icon in the top right corner of the VS Code window. This opens a search bar where you can start typing the command you want to execute.

**Using the Command Palette:**

Once you open the Command Palette, a search bar appears. Start typing the name of the command you want to use, and VS Code will provide suggestions based on what you've typed.

**Examples of Common Tasks using the Command Palette:**

* **Open Files:** Quickly open any file in your project by typing its name or path.
* **Navigate through Code:** Jump to specific lines, symbols, or recently opened files.
* **Format Code:** Format your code automatically based on your preferred style.
* **Run and Debug:** Start or stop your code execution, set breakpoints for debugging.
* **Install Extensions:** Search for and install extensions to add new functionalities to VS Code.
* **Change Settings:** Access and modify various editor settings and configurations.

# EXTENSIONS IN VS CODE

-These are small software add-ons that extend VS Code's functionality, catering to specific programming languages, frameworks, and development tasks.

**The Role of Extensions:**

* **Enhanced Functionality:** Extensions add features and tools that aren't built-in to VS Code by default. This can include syntax highlighting for new languages, code completion for specific frameworks, linters for code quality checks, debuggers for tailored debugging experiences, and version control integration for streamlined collaboration.
* **Increased Productivity:** Extensions can automate repetitive tasks, improve code navigation, and provide valuable insights into your code, saving you time and effort.
* **Customization:** With a vast library of extensions, you can personalize your VS Code environment to match your workflow and development needs.

**Finding, Installing, and Managing Extensions:**

1. **Extensions Marketplace:** The built-in Extensions marketplace is your one-stop shop for discovering and installing extensions. Access it by clicking on the Extensions icon (puzzle piece icon) in the Activity Bar.
2. **Search and Browse:** Search for extensions by name or functionality. You can also browse through curated collections based on categories like programming languages, themes, and debuggers.
3. **Install and Manage:** Once you find an extension, click the "Install" button. Installed extensions are listed in the Extensions view, where you can manage.

**Essential Extensions for Web Development:**

Here are some popular and valuable extensions for web development:

* **Language-specific extensions:**
  + **HTML (HTML Tools):** Provides intelligent code completion, emmet shortcuts for faster HTML coding, and live previews for real-time feedback on your HTML structure.
  + **CSS (CSS Peek):** Offers peek functionality to view CSS styles associated with HTML elements, improving code navigation and understanding.
  + **JavaScript (ESLint, Prettier):** ESLint helps identify potential errors and enforce code style guidelines, while Prettier automatically formats your code for consistency and readability.
* **Debuggers:** Extensions like the Chrome Debugger or Debugger for Chrome allow you to step through your code, set breakpoints, and inspect variables during development, simplifying debugging processes.
* **Version Control Integration:** GitLens provides a visual representation of your Git repository, making it easier to understand code changes and collaborate with others.
* **Live Server:** This extension allows you to launch a development server directly within VS Code, letting you preview your web pages in a browser with automatic reloading upon code changes, streamlining development workflow.

# INTEGRATED TERMINAL

## Utilizing the Integrated Terminal in VS Code

The integrated terminal in VS Code provides a convenient way to execute commands and interact with your operating system directly from your code editor. Here's how to access and use it effectively:

**Opening the Integrated Terminal:**

There are three ways to open the integrated terminal:

* **Menu:** Navigate to the **Terminal > New Terminal** menu option.
* **Keyboard Shortcut:** Press Ctrl+ (backtick)
* **Panel:** Click on the "+" icon in the bottom panel and select "Terminal" from the options.

This will open a new terminal window within the VS Code interface, typically at the bottom of the window.

**Using the Integrated Terminal:**

Once the terminal opens, you can use it just like any regular terminal application. Type your desired commands and press Enter to execute them. The terminal window displays the output of your commands, allowing you to interact with your system, manage files, run build scripts, and perform other tasks related to your development process.

**Advantages of the Integrated Terminal:**

* **Convenience:** The biggest advantage is having the terminal readily available within your code editor.
* **Context Awareness:** The integrated terminal can be context-aware. For example, if you right-click on a file or folder in the editor, you can choose options to open a terminal in that specific directory, eliminating the need to navigate manually using command-line constructs.
* **Integration with VS Code Features:** The integrated terminal can leverage other VS Code features. For instance, you can copy and paste code snippets directly between the editor and the terminal.
* **Multiple Terminals:** You can open multiple terminal instances within VS Code, allowing you to work in parallel sessions or manage different tasks simultaneously.

# FILE AND FOLDER MANAGEMENT

VS Code empowers you to organize and interact with your project files and folders seamlessly. Here's a breakdown of how to create, open, and manage them effectively:

**Creating Files and Folders:**

* **New File:**
  + **Menu:** Navigate to **File > New File**.
  + **Shortcut:** Press Ctrl+N
  + **Explorer Context Menu:** Right-click inside a folder in the Explorer sidebar and select "New File".
* **New Folder:**
  + **Menu:** Navigate to **File > New Folder**.
  + **Shortcut:** Press Ctrl+Shift+N
  + **Explorer Context Menu:** Right-click inside a folder in the Explorer sidebar and select "New Folder".
* **Explorer Drag and Drop:** Drag and drop files or folders from your operating system's file explorer directly into the VS Code Explorer to create copies within your project.

**Opening Files:**

* **Double-click:** Double-click on a file name in the Explorer sidebar to open it in the editor.
* **Search Bar:** Use the search bar at the top of the Explorer sidebar to quickly locate files by name.
* **Go to File (Quick Open):**
  + **Search within Editor:** Start typing the name of the file you want to open. VS Code will suggest matching files as you type.
  + **Select:** Choose the desired file from the suggestions list to open it.

**Managing Files and Folders:**

* **Renaming:** Right-click on a file or folder and select "Rename" or simply click once on the name to edit it directly.
* **Moving:** Drag and drop files or folders within the Explorer sidebar to reorganize them. You can also use cut, copy, and paste options from the context menu.
* **Deleting:** Right-click on a file or folder and select "Delete".

**Navigating Efficiently:**

* **Explorer Panel:** Navigate through your project directory structure using the Explorer sidebar. You can expand and collapse folders to view their contents.
* **Breadcrumbs:** The breadcrumbs bar at the top of the editor area indicates your current location within the project directory. Click on any directory name in the breadcrumbs to jump to that location.
* **Go to Definition/Symbol:** These functionalities allow you to quickly jump to the definition of a function, variable, or symbol used within your code, aiding navigation in large projects.
* **Recently Used Files:** VS Code maintains a list of recently opened files. Access this list from the **File** menu to quickly switch back to previously used files.

# SETTINGS AND PREFERENCES

There are two primary ways to access settings in VS Code:

* **Menu:** Navigate to **File > Preferences > Settings** This opens the Settings editor within VS Code.
* **Keyboard Shortcut:** Press Ctrl+, This is the quickest way to access settings.

**Customizing Settings:**

The Settings editor displays a search bar and a categorized list of settings. You can explore the list or use the search bar to find specific settings you want to modify.

**Examples:**

* **Changing Theme:**
  + Search for "Theme" in the settings search bar.
  + A list of available themes will appear. Click on the desired theme to apply it. You can also install additional themes from the VS Code Marketplace.
* **Adjusting Font Size:**
  + Search for "Font Size" in the settings search bar.
  + You'll find a setting named "Editor > Font Size". Use the slider or enter a desired font size value.
* **Modifying Key bindings:**
  + Search for "Keyboard Shortcuts" in the settings search bar.
  + You can browse through the existing keybindings or use the search bar to find a specific command.
  + Click on the desired command and then click on the keybinding value on the right. This allows you to define a new keyboard shortcut for that command.

# 

# DEBUGGING IN VS CODE

VS Code offers a powerful debugging experience that helps you identify and fix errors in your code.

**Prerequisites:**

* **Installed Debugger Extension:** Depending on your programming language, you might need to install a specific debugger extension from the VS Code Marketplace. For example, for Python, you might need the "Python" extension.
* **Launch Configuration:** A launch configuration defines how your program is launched for debugging. VS Code often provides default configurations or allows you to create custom ones.

**Steps:**

1. **Open your program file:** Ensure the file you want to debug is open in the VS Code editor.
2. **Set Breakpoints:** Click on the line numbers where you want the program to pause during execution.
3. **Open the Run and Debug view:** Click on the "Run and Debug" icon in the Activity Bar,
4. **Select the Launch Configuration:** If you have multiple configurations, choose the appropriate one that matches your program's setup. VS Code might suggest a default configuration.
5. **Start Debugging:** Click on the green play button in the Run and Debug view. This will launch your program in debug mode, pausing at any breakpoints you set.

**Key Debugging Features in VS Code:**

* **Breakpoints:** Pause program execution at specific lines.
* **Stepping:** Step through your code line by line, inspecting variables and the call stack at each step. This helps you understand the program's flow and identify where errors occur.
  + **Step Over:** Executes the current line and moves to the next line
  + **Step Into:** Executes the current line and steps into any function calls that are made.
  + **Step Out:** Executes the current line of code and exits the current function call, returning to the calling function.
* **Call Stack:** View the sequence of function calls that led to the current point in the execution. This helps you understand the context of an error.
* **Variables:** Inspect the values of variables at any point during execution. This allows you to verify if variables are assigned and hold the expected values.
* **Console:** View the output of your program while debugging.
* **Conditional Breakpoints:** Set breakpoints that only trigger under specific conditions, such as when a variable reaches a certain value.

# USING SOURCE CONTROL

VS Code offers seamless integration with Git, allowing you to manage your code versions directly from within the editor. Here's how to set up Git in VS Code, make commits, and push your changes to a remote repository like GitHub:

### Initializing a Git repository:

1. **Open your project folder:** Ensure you have your project files and folders open in VS Code.
2. **Open the Source Control view:** Click on the Source Control icon (looks like a git branch) in the Activity Bar,
3. **Initialize Repository:** If you don't see any Git functionality within the Source Control view, click on the "+" icon and select "Initialize Git Repository". This creates a new Git repository within your project folder.

### Making Commits:

1. **Stage Changes:** Changes to your code files are not automatically tracked by Git. You need to stage them explicitly before making a commit. In the Source Control view, you'll see a list of modified files. Click on the checkbox next to each file you want to include in the commit.
2. **Write a Commit Message:** Click on the message box at the bottom of the Source Control view. Enter a clear and concise message describing the changes you've made in this commit.
3. **Commit Changes:** Click on the green checkmark button to commit the staged changes with your message.

### Pushing Changes to GitHub:

1. **Connect to GitHub:** You'll need to connect your VS Code to your GitHub account for pushing changes. Follow the instructions within VS Code to establish this connection.
2. **Create a Remote Repository (Optional):** If you haven't already, create a new repository on GitHub.com to store your code.
3. **Add Remote:** In the Source Control view, click on the "..." menu and select "Add remote". Specify a name for the remote (e.g., "origin") and provide the URL of your GitHub repository.
4. **Push Changes:** Once you've made a commit and connected your account, click on the "..." menu again and select "Push to origin" (or "Push branch" if you're working on a specific branch). This pushes your local commits to the remote repository on GitHub.