Tutorial: Get started with Visual Studio Code

In this tutorial, you learn about the key features of Visual Studio Code to help you get started with coding quickly. You learn about the different components of the user interface and how to customize it to your liking. You then write some code and use the built-in code editing features, such as IntelliSense and Code Actions, and you learn about running and debugging your code. By installing a language extension, you add support for a different programming language.

Tip
If you prefer to follow along with a video, you can watch the Getting Started video, which covers the same steps as this tutorial.

Prerequisites

- Download and install Visual Studio Code on your computer

Open a folder in VS Code

You can use VS Code to work on individual files to make quick edits, or you can open a folder, also known as a *workspace*.

Let's start by creating a folder and opening it in VS Code. You'll use this folder throughout the tutorial.

1. Open Visual Studio Code and select **File** > **Open Folder...** from the menu to open a folder.
2. Select **New Folder** and create a new folder named `vscode101`. Then select **Select Folder** (**Open** on macOS) to open the folder in VS Code.

   VS Code now considers the folder you've opened a workspace.
3. On the Workspace Trust dialog, select **Yes, I trust the authors** to enable all features in the workspace.

   Important

Workspace Trust lets you decide whether code in your project folder can be executed by VS Code. When you download code from the internet, you should first review it to make sure it's safe to run. Get more info about Workspace Trust.

4. You should now see the **Explorer** view on the left, showing the name of the folder.

   You'll use the Explorer view to view and manage the files and folders in your workspace.

Tip
When you open a folder in VS Code, VS Code can restore the UI state for that folder, such as the open files, the active view, and the layout of the editor. You can also configure settings that only apply to that folder, or define debug configurations. Get more info about workspaces.

Explore the user interface

Now that you have a folder open in VS Code, let's take a quick tour of the user interface.

Switch between views with the Activity Bar

1. Use the Activity Bar to switch between different views.

   Tip
   Hover over the Activity Bar to see the name of each view and the corresponding keyboard shortcut. You can toggle a view open and closed by selecting the view again or pressing the keyboard shortcut.

2. When you select a view in the Activity Bar, the **Primary Side Bar** opens to show view-specific information.

   For example, the Run and Debug view enables you to configure and start debugging sessions.

View and edit files with the Editor

1. Select the Explorer view in the Activity Bar, and select the **New File...** button to create a new file in your workspace.

2. Enter the name `index.html` and press `Enter`.

   A file is added to your workspace and an Editor opens in the main area of the window.

3. Start typing some HTML code in the `index.html` file.

   As you type, you should see suggestions popping up that help you complete your code (*IntelliSense*). You can use the `Up` and `Down` keys to navigate the suggestions, and `Tab` to insert the selected suggestion.

4. Add more files to your workspace and notice that each file opens a new Editor tab.

   You can open as many editors as you like and view them side by side vertically or horizontally. Learn more about side by side editing.

Access the terminal from the Panel area

1. VS Code has an integrated terminal. Open it by pressing ``Ctrl+` ``. You can also use the **View** > **Terminal** menu item.

   You can choose between different shells, such as PowerShell, Command Prompt, or Bash, depending on your operating system configuration.

2. In the terminal, enter the following command to create a new file in your workspace.

   Bash
   ```
   echo "Hello, VS Code" > greetings.txt
   ```

   The default working folder is the root of your workspace. Notice that the Explorer view automatically picks up and shows the new file.

3. You can open multiple terminals simultaneously. Select the **Launch Profile** dropdown to view the available shells and choose one.

Access commands with the Command Palette

1. Open the **Command Palette** by pressing `Ctrl+Shift+P`. You can also use the **View** > **Command Palette** menu item.

   Many of the commands in VS Code are available through the Command Palette. When you install extensions, they can add extra commands to the Command Palette.

   Tip
   Notice that the Command Palette shows the default keyboard shortcut for commands that have one. You can use the keyboard shortcut to run the command directly.

2. The Command Palette supports different modes of operation:

   1. **Command mode (>)**: after the `>` symbol, start typing to filter the command list. For example, type `move terminal` to find commands to move the terminal to a new window.

   2. **Quick Open mode**: remove the `>` character and start typing to search for files in your workspace. You can use the `Ctrl+P` keyboard shortcut to open the Command Palette and start searching for files directly.

   3. **Symbol search mode (#)**: replace the `>` character by `#` to search for symbols like variables or functions in your code.

Tip
VS Code uses fuzzy matching to find files or commands. For example, typing `odks` returns the `Open Default Keyboard Shortcuts` command.

Configure VS Code settings

You can customize almost every part of VS Code by configuring settings. You can use the **Settings Editor** to modify the settings in VS Code or directly modify the `settings.json` file.

1. Press `Ctrl+,` to open the Settings Editor (or select the **File** > **Preferences** > **Settings** menu item).

Tip
Use the search box to filter the list of settings that are shown.

2. By default, VS Code doesn't automatically save modified files. Select a value from the Auto Save dropdown to change this behavior.

   VS Code automatically applies changes to settings. When you modify a file in your workspace, it should now be automatically saved.

3. To revert a setting to its default value, select the gear icon next to the setting and select **Reset Setting**.

   Tip
   You can quickly find all modified settings by typing `@modified` in the search box or selecting the **Modified** filter.

4. You can use the tabs in the Settings Editor to switch between **User** settings and **Workspace** settings.

   User settings apply across all your workspaces. Workspace settings only apply to the current workspace. Workspace settings override user settings. Get more information about settings in VS Code.

Write some code

VS Code has built-in support for JavaScript, TypeScript, HTML, CSS, and more. In this tutorial, you create a sample JavaScript file and use some of the code editing features that VS Code offers.

VS Code supports many programming languages and in a next step, you'll install a language extension to add support for a different language, namely Python.

1. In the Explorer view, create a new file `app.js`, and start typing the following JavaScript code:

   JavaScript
   ```
   function sayHello(name) {
     console.log('Hello, ' + name);
   }

   sayHello('VS Code');
   ```

As you type, you should see suggestions popping up that help you complete your code (*IntelliSense*). You can use the `Up` and `Down` keys to navigate the suggestions, and `Tab` to insert the selected suggestion.

Notice also the formatting of the code (*syntax highlighting*), to help you distinguish between different parts of the code.

2. When you put the cursor on the string `Hello,`, you should see a lightbulb icon appear to indicate there's a Code Action.

   You can also use the `Ctrl+Space` keyboard shortcut to open the lightbulb menu.

3. Select the lightbulb icon, and then select **Convert to template string**.

   Code Actions are suggestions to apply quick fixes to your code. In this case, the Code Action converts `""Hello, " + name` into a template string `` `Hello, ${name}` ``, which is a special JavaScript construct to embed expressions in strings.

Learn more about code editing features, IntelliSense, code navigation, and refactoring in VS Code.

Use source control

Visual Studio Code has integrated source control management (SCM) and includes Git support out-of-the-box.

Let's use the built-in Git support to commit the changes you've made previously.

1. Select the **Source Control** view in the Activity Bar to open the Source Control view.

2. Make sure you have Git installed on your computer. If you don't have Git installed, you'll see a button in the Source Control view to install it on your machine.

3. Select **Initialize Repository** to create a new Git repository for your workspace.

   After you initialize a repository, the Source Control view shows the changes you've made in your workspace.

4. You can stage individual changes by hovering over a file and selecting + next to a file.

   Tip
   To stage all changes, hover over **Changes** and select the **Stage All Changes** button.

5. Enter a commit message, for example `Add hello function`, and then select the **Commit** to commit the changes to your Git repository.

   Tip
   Select **Graph** in the Source Control view to show a visual representation of the commit history of your Git repository.

There's a lot more to discover about source control in VS Code. Get more info about source control in VS Code.

Install a language extension

VS Code has a rich ecosystem of extensions that let you add languages, debuggers, and tools to your installation to support your specific development workflow. There are thousands of extensions available in the Visual Studio Marketplace.

Let's install a language extension to add support for Python, or any other programming language you are interested in.

1. Select the **Extensions** view in the Activity Bar.

   The Extensions view enables you to browse and install extensions from within VS Code.

2. Enter *Python* in the Extension view search box to browse for Python-related extensions. Select the **Python** extension published by Microsoft, and then select the **Install** button.

3. Now, create a new Python file `hello.py` in your workspace, and start typing the following Python code:

   Python

```
def say_hello(name):
    print("Hello, " + name)

say_hello("VS Code")
```

Notice that you now also get suggestions and IntelliSense for the Python code.

Run and debug your code

VS Code has built-in support for running and debugging Node.js applications. In this tutorial, you use the Python extension you installed in the previous step to debug a Python program.

Let's debug the `hello.py` program that you created in the previous step.

1. Make sure that Python 3 is installed on your computer.

   If there's no Python interpreter installed on your computer, you'll see a notification in the lower right corner of the window. Select **Select Interpreter** to open the **Command Palette** and select the Python interpreter you want to use or install one.

2. In the `hello.py` file, place the cursor on the `print` line and press `F9` to set a breakpoint.

   A red dot appears in the left margin of the editor, indicating that a breakpoint is set. With a breakpoint, you can pause the execution of your program at a specific line of code.

3. Press `F5` to start a debugging session.

   1. Select the Python debugger:

   2. Choose to run the current Python file:

4. Notice that the program starts and that the execution stops at the breakpoint you set.

   Tip

Inspect the value of the `name` variable by hovering over it in the editor while the execution is paused. You can view the value of variables at any time in the **Variables** view in the **Run and Debug** view.

5. Press the **Continue** button in the Debug toolbar or press `F5` to continue the execution.

There are many more debugging features in VS Code, such as watch variables, conditional breakpoints, and launch configurations. Dive into the details of debugging in VS Code.

Enhance your coding with AI and GitHub Copilot

GitHub Copilot is an AI-powered assistant that helps you write code faster, and can help you with a wide range of tasks, such as code completion, code refactoring, and fixing errors.

Let's get started by getting code suggestions from Copilot.

1. Make sure you have set up GitHub Copilot in VS Code. Follow the steps in our Copilot Setup guide.

   Tip
   If you don't have a Copilot subscription yet, you can use Copilot for free by signing up for the Copilot Free plan and get a monthly limit of inline suggestions and chat interactions.

2. In the `hello.py` file, place the cursor at the end of the file and type this function header.

   Python
   ```python
   def say_day_of_week(date)
   ```

   Copilot will automatically suggest the rest of the function. Accept the code suggestion by pressing `Tab`.

3. Next, let's invoke our new function.

   Python
   ```python
   say_day_of_week(date.today())
   ```

   Notice that there's a squiggle on the `date` keyword, indicating that there's an error.

4. Put the cursor, on the `date` keyword, select the *lightbulb* icon, and then select **Fix**.

Copilot will suggest a fix for the error. Select **Accept** if you're happy with the suggestion.

Tip
You can also use the `Ctrl+.` keyboard shortcut to manually trigger a quick fix.

There's a lot more you can do with Copilot in VS Code. Discover more about GitHub Copilot in VS Code with our Copilot Quickstart.

Next steps

Congratulations! You've completed the tutorial and explored some of the key features of Visual Studio Code. Now that you've learned the basics of Visual Studio Code, get more info about how to:

- Use AI to accelerate your coding
- Discover and run unit tests for your code
- Use the integrated terminal
- Set up a remote development environment