CHAPTER 1

Get Started with GitHub Copilot

Software development is full of challenges to overcome. For years, it has been known that programming with a partner can help you learn more, produce better work, and gain more satisfaction while accomplishing your tasks. Although the benefits of pair programming are known, it isn't always possible to have a pair programming partner with you—until now.

GitHub Copilot is your artificial intelligence (AI) pair programming partner, always ready to assist and eager to help you learn! This book will walk you through how to best utilize GitHub Copilot to help you write better code and do it faster.

In this chapter, we will focus on the required steps for getting started with GitHub Copilot. Let's begin!

- Learn Why GitHub Copilot Matters to You
- Create a GitHub Account
- Acquire a GitHub Copilot License
- Install an IDE Extension
- First Run: Test Copilot

Learn Why GitHub Copilot Matters

GitHub Copilot is your AI pair programmer that can assist you in every phase of your software development lifecycle. Whether you are defining your next great feature or configuring a complex continuous integration/continuous delivery (CI/CD) pipeline for an enterprise-grade deployment, GitHub Copilot will be by your side every step of the way, giving you bespoke insights into your business needs. Get ready to take your development productivity and joy of programming to the next level.

You will find your favorite new AI-powered pair programmer, GitHub Copilot, in an ever-growing number of places within your integrated development environment (IDE) and beyond. This book will teach you how to use each Copilot feature in the different license options. We will also explore case studies with best practices that will help to extend your use of Copilot into all areas of your development lifecycle.

To prove the effectiveness of Copilot, a team at GitHub has conducted qualitative and quantitative research to test their hypothesis of improved developer productivity and happiness. One large-scale survey resulted in some amazing results: 88% indicated they were more productive, 74% said they were able to focus on more satisfying work, 96% indicated they were faster with repetitive tasks, and 73% of survey participants indicated they had more time in a flow state [1].

In addition to the survey, the GitHub team conducted a qualitative experiment by having developers create a web server in JavaScript. Individuals using Copilot finished the exercise on average 55% faster [1]! The team used GitHub Classroom to score submissions for correctness and completeness automatically.

Create a GitHub Account

Before you can start using Copilot, you need to have a valid GitHub account. Head to the following web page and ensure that you have access to your account before getting started:

```
https://github.com/login
```

Acquire a GitHub Copilot License

With a valid GitHub account, we can now review the available licenses for GitHub Copilot. You will need to pick the license that is best for you. There are three GitHub Copilot plans available.

- Copilot Individual
- Copilot Business
- Copilot Enterprise

There are several factors to consider when choosing the correct plan. If you are a student or a maintainer of a popular open-source project, you might be eligible for a free Copilot Individual license.

You can get more information on licenses on this web page:

```
https://docs.github.com/enterprise-cloud@latest/billing/
managing-billing-for-github-copilot/about-billing-for-github-copilot
```

Install an IDE Extension

GitHub Copilot runs as an extension in the following IDEs:

- Azure Data Studio
- JetBrains IDEs (IntelliJ, PyCharm, Rider, and so on)
- Vim/Neovim
- Visual Studio
- Visual Studio Code

NOTE **In this book, we will be using the Visual Studio Code IDE for most of the examples. If you use one of the other supported IDEs as your preferred development platform, the information shared in these examples will be transferrable. We will be covering these additional IDEs later in the book when we detail how to set up and configure them to work with GitHub Copilot.**

NOTE **Support for the JetBrains IDEs is currently in beta.**

Download Visual Studio Code

You can download Visual Studio Code (VS Code) from the following page:

```
http://code.visualstudio.com
```

Once you have installed VS Code on your computer, you should see a welcome screen (see Figure 1.1).

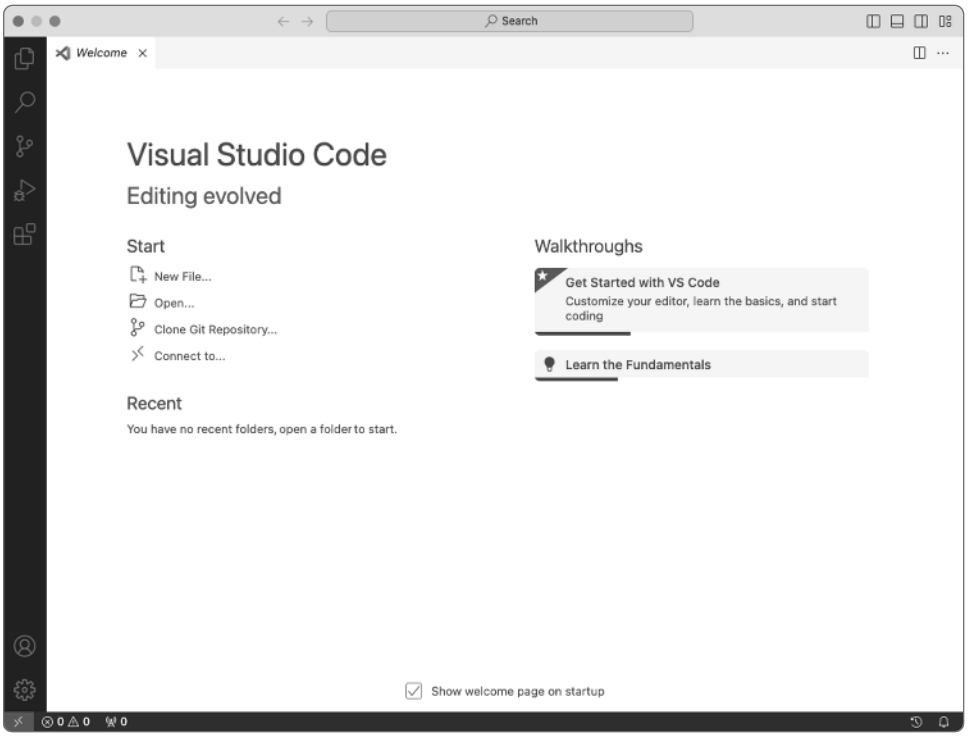

Figure 1.1: VS Code welcome screen

Install the GitHub Copilot Extension

Now that you have the VS Code IDE installed and open, let's navigate to the Extensions panel on the Action Bar. You will find the Extensions panel identified by the "squares" icon.

Now follow these steps:

1. Open the Extensions panel.
2. Search for "GitHub Copilot."
3. Within the GitHub Copilot extension result, click Install (see Figure 1.2).

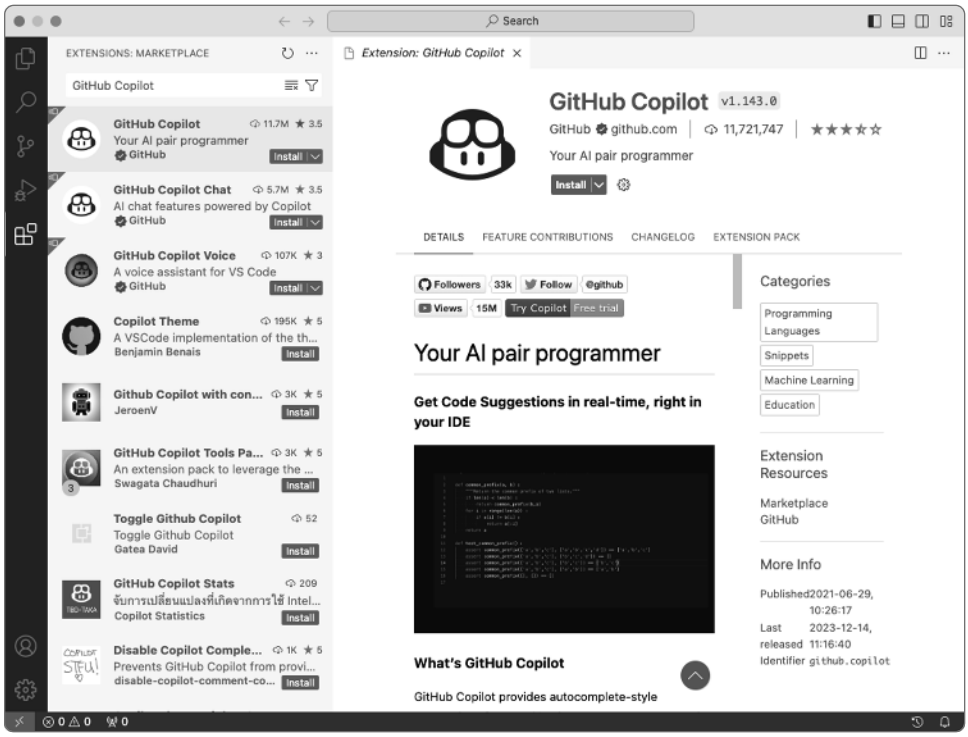

Figure 1.2: VS Code Extensions panel

Configure the IDE Settings for Copilot

After successfully installing the Copilot extension, you are ready to ensure you are authenticated to your GitHub account within VS Code. You should see a pop-up in the lower-right corner of VS Code prompting you to sign in to GitHub (see Figure 1.3). Please use this option to sign in.

If you don't see this prompt after installing the extension, you can also authenticate using the profile menu on the Action Bar (see Figure 1.4).

After completing the sign-in process via the GitHub authentication pages, you can verify your authentication status within VS Code via the bottom-right Copilot icon. Click this icon to bring up the Copilot status menu (see Figure 1.5).

Within the status menu you have access to your status, chat, settings, logs, documentation, and forums.

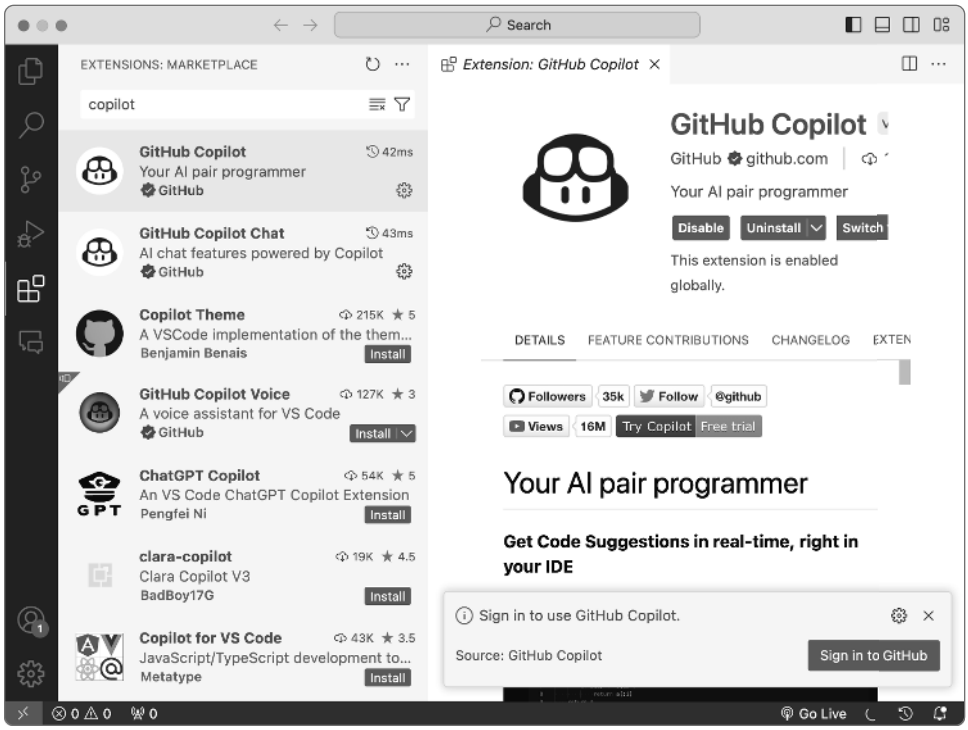

Figure 1.3: Sign-in prompt

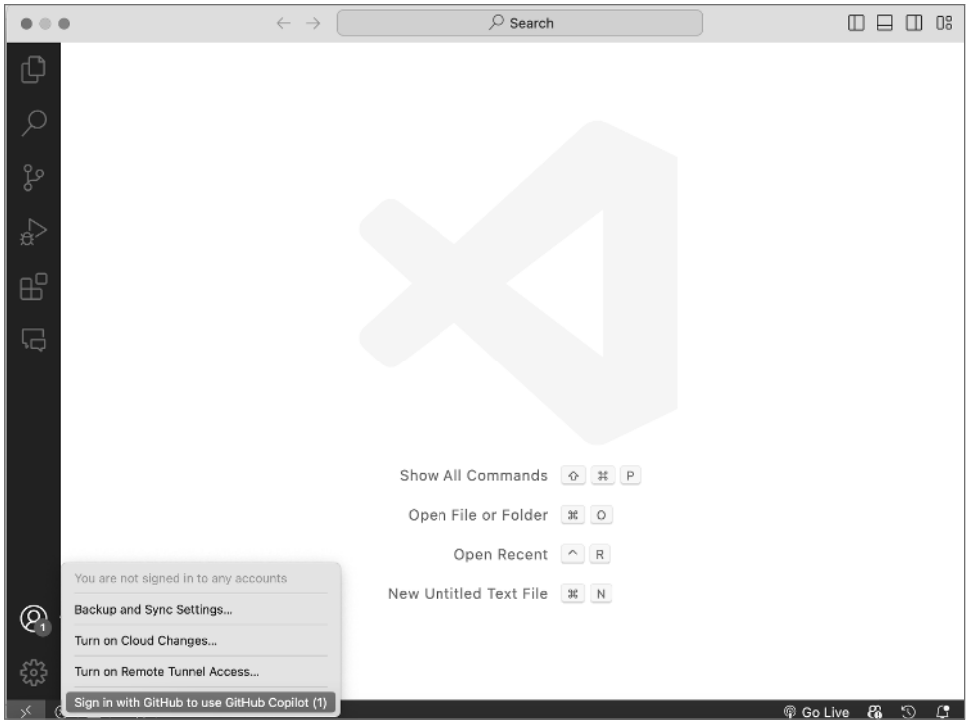

Figure 1.4: Signing in via the Action Bar

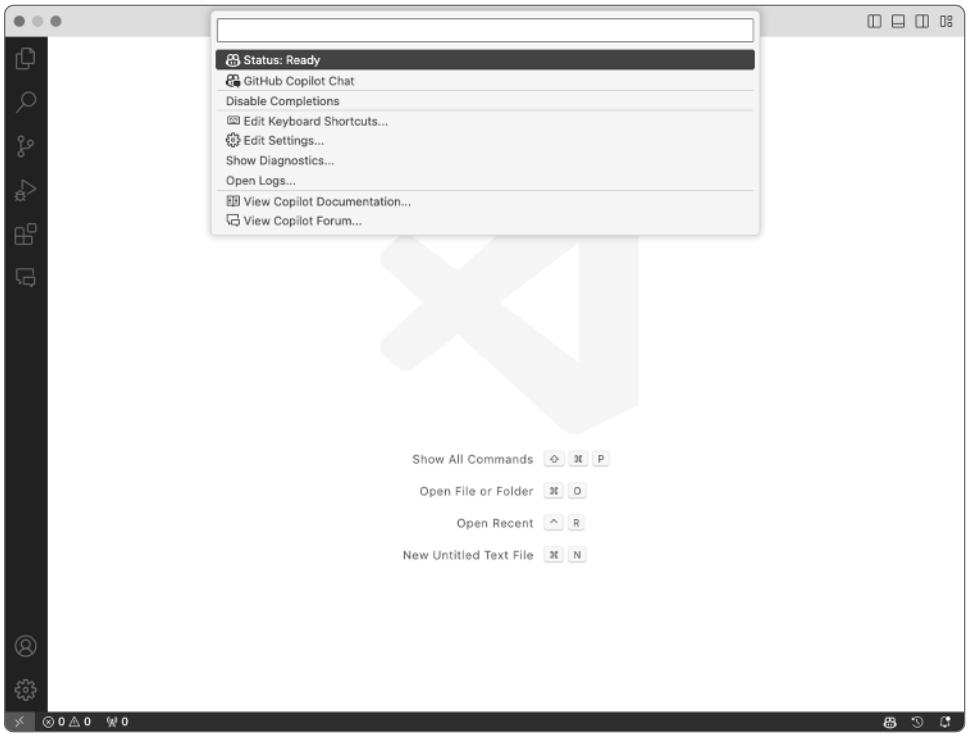

Figure 1.5: Copilot status menu

Install Node.js

Lastly, Node.js will need to be installed to run the example. Node.js is an open-source, cross-platform, back-end JavaScript runtime environment. It allows us to execute JavaScript outside of the web browser.

The easiest way to install Node.js is to go to the website:

`https://nodejs.org/en/download`.

Based on your operating system and computer hardware, select the appropriate download, and follow the installation steps.

After installing Node.js, run the following command in your terminal to confirm that you have installed it successfully.

```
node -v
```

This command will output the node version you have installed.

First Run: Test Copilot

As mentioned, this book will be showcasing the features of GitHub Copilot primarily in Visual Studio Code. There are dedicated chapters later in the book to detail all the other GitHub Copilot IDE experiences.

While most of the code completion features are universal between IDEs, there are differences in the menus, the keyboard shortcuts, and the availability of Copilot Chat (which is available only in Visual Studio and VS Code).

Get the Prerequisites

As mentioned, the following are the prerequisites to testing Copilot:

- VS Code
- GitHub account
- GitHub Copilot license
- GitHub Copilot extension
- Node.js

Explore Copilot

Let's make sure that Copilot is working by writing a quick example function. In this section, you will create a palindrome checker to showcase some of the basic interactions you will have with Copilot within your editor.

Start by opening a folder in VS Code. You can do this via the Explorer menu (see Figure 1.6) or the keyboard shortcut (Cmd+O/Ctrl+O).

NOTE **Throughout this book, keyboard shortcuts will be displayed for both macOS and Windows OS.**

Create a new folder called **copilot-test** and click Open within your Finder/Explorer window.

Add a new file to your open folder called `palindrome-checker.js` (see Figure 1.7).

Now you are ready to start writing the Node.js script. Let's start by typing a top-level comment in the `palindrome-checker.js` file, as shown here:

```
// node.js application that checks if a string is a palindrome
```

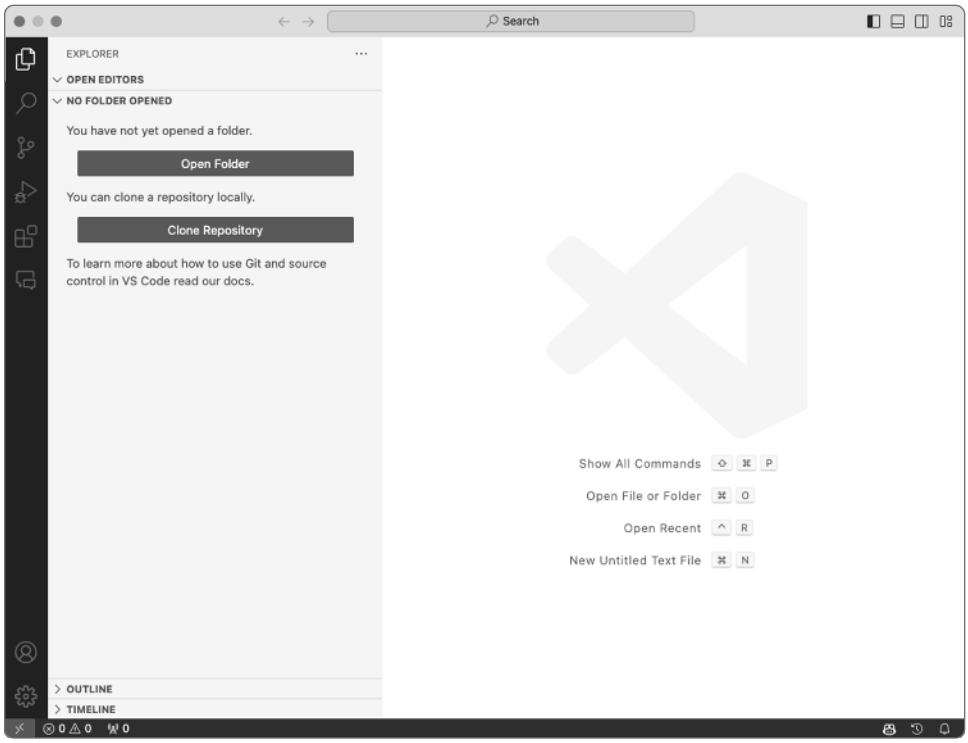

Figure 1.6: Open Folder button

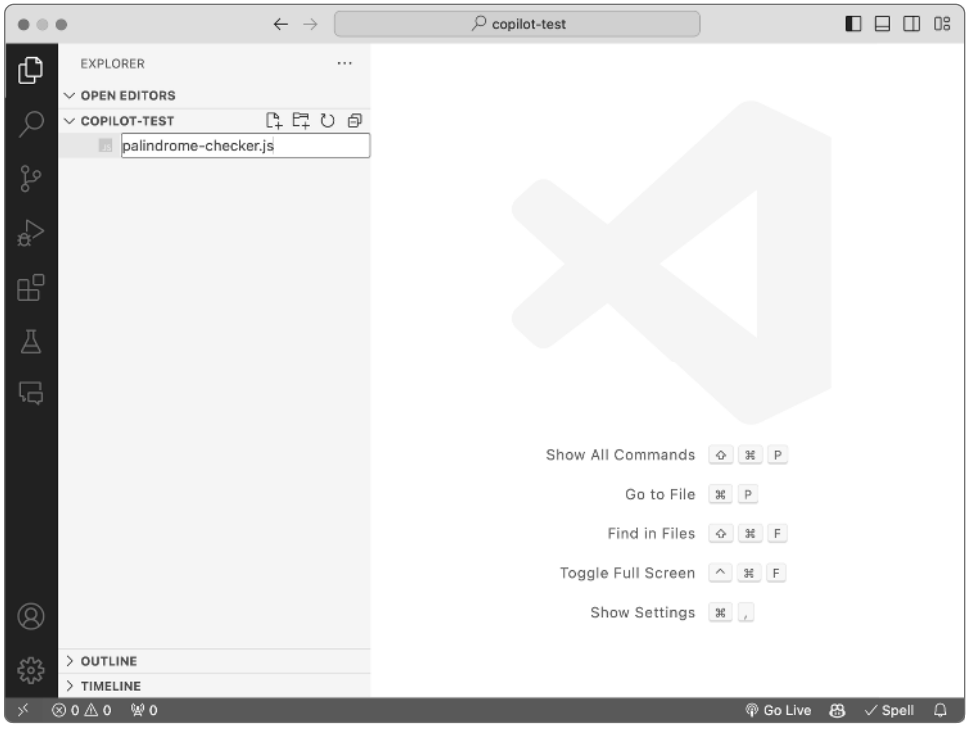

Figure 1.7: Creating the `palindrome-checker.js` file

As you start writing this comment, Copilot should start suggesting some text to complete it (see Figure 1.8).

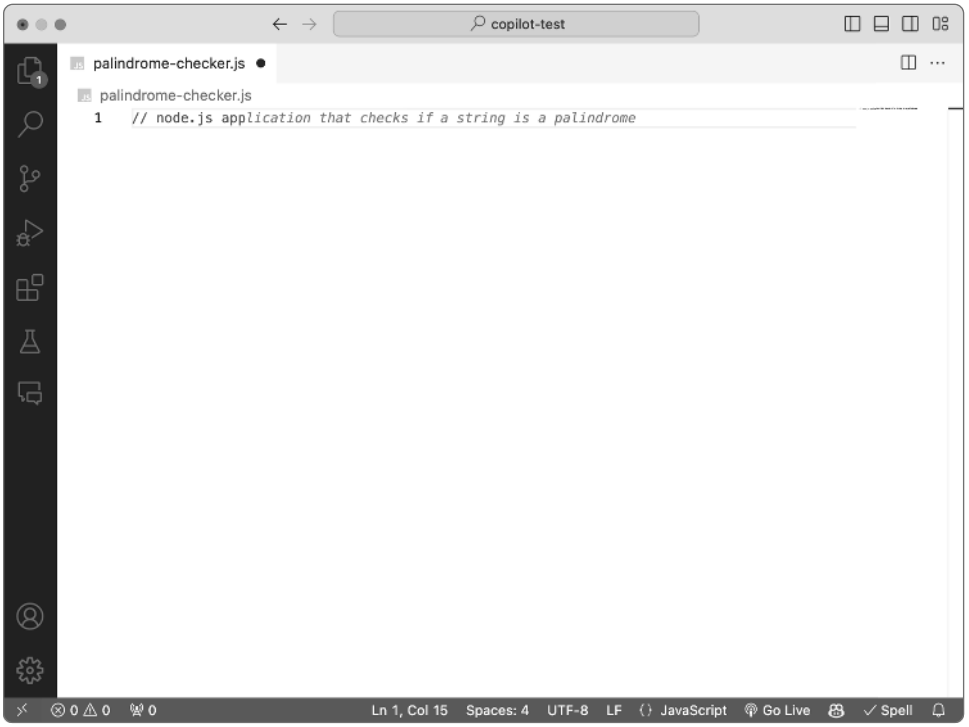

Figure 1.8: Copilot completion suggestion for top-level comment

You can accept the Copilot text completion by hitting the Tab key.

NOTE **You can provide context to Copilot in several ways, such as descriptive file comments, inline comments, examples, filenames, imports, method names, variables, and open file tabs. You will see all these context types in action throughout this book.**

With the comment at the top of the file, continue to the next line. You can provide additional context by adding some examples of palindromes you would like the checker to expect to receive.

```
// examples: racecar, taco cat
```

Without this example comment, there is a chance that the palindrome function will not account for phrase inputs. Adding this context is a way for you to continue to refine the context given to Copilot to get better output.

Now that you have a short, clear, and targeted comment for your file, you can add two line returns to the file to move on to the function implementation. If Copilot doesn't automatically give you the line you would like as a suggestion, you can start typing the statement you would prefer. Copilot will then use this additional context to drive a better completion suggestion for the current line you are working on.

In my case, I will accept a line comment describing a function to check if a string is a palindrome and implement the first line of the function name (see Figure 1.9).

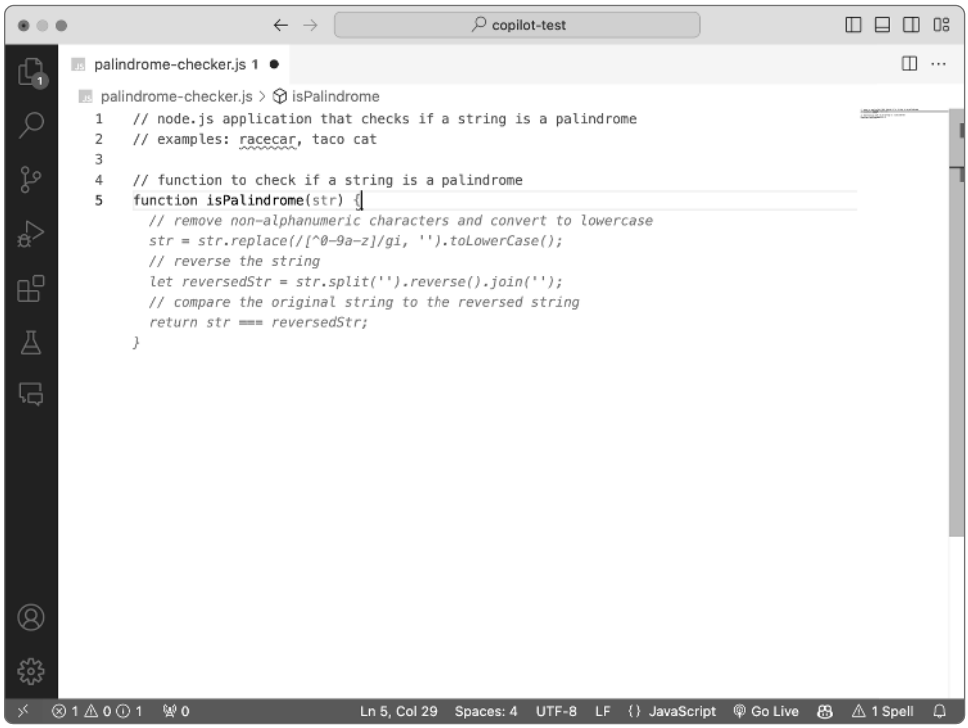

Figure 1.9: Copilot completion suggestion for palindrome function

NOTE **GitHub Copilot is nondeterministic and may return different results for you even though you have the same comment structure within your file. We will cover the AI behind Copilot in depth in Chapter 2, "Decoding GitHub Copilot."**

You can continue to accept and refine code completions provided by Copilot in the file. Accept suggestions with the Tab key or use the menu if available when hovering on the suggestion text.

After coding the `isPalindrome` function, you can continue to add some test cases. Copilot will most likely suggest continuing. But to ensure you get the output you want, add another line comment.

```
// test cases
```

After you add a new line, Copilot should start completing the test cases using the examples from the top-level file comment. Continue implementing these test cases (see Figure 1.10).

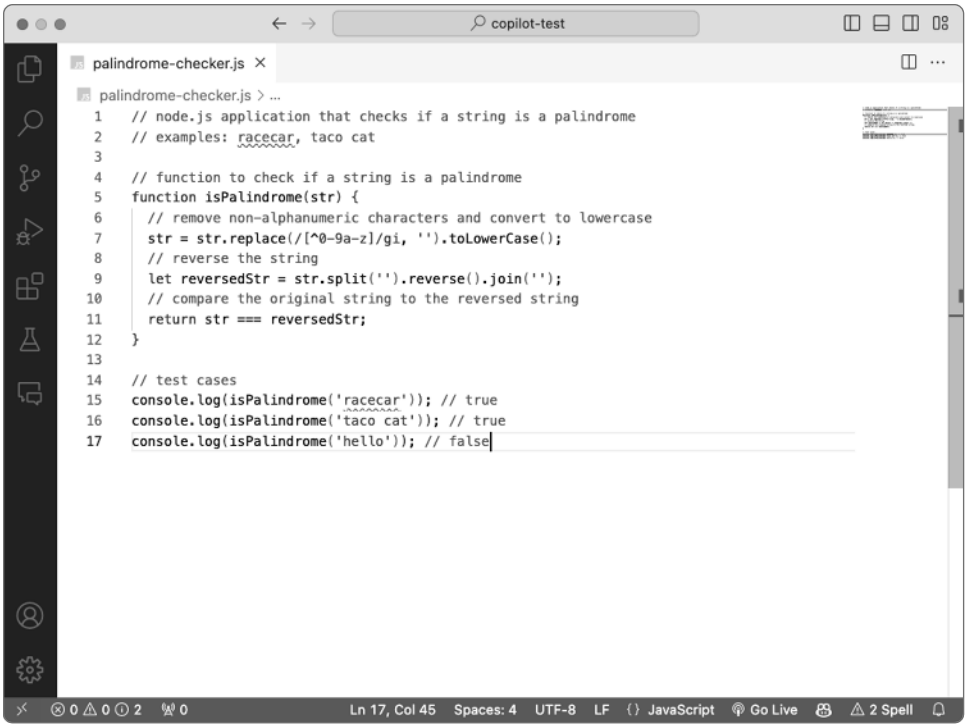

Figure 1.10: Palindrome function with tests

With the function and tests, you can now run this file to see if you get the expected output. If you are following along in VS Code, you can open the integrated terminal to run the function via the keyboard shortcut (Ctrl+`).

Within the integrated terminal that is open to the current location of your Node.js file, type the following to run the function:

```
node palindrome-checker.js
```

You should now see output indicating the expected results from your function (see Figure 1.11).

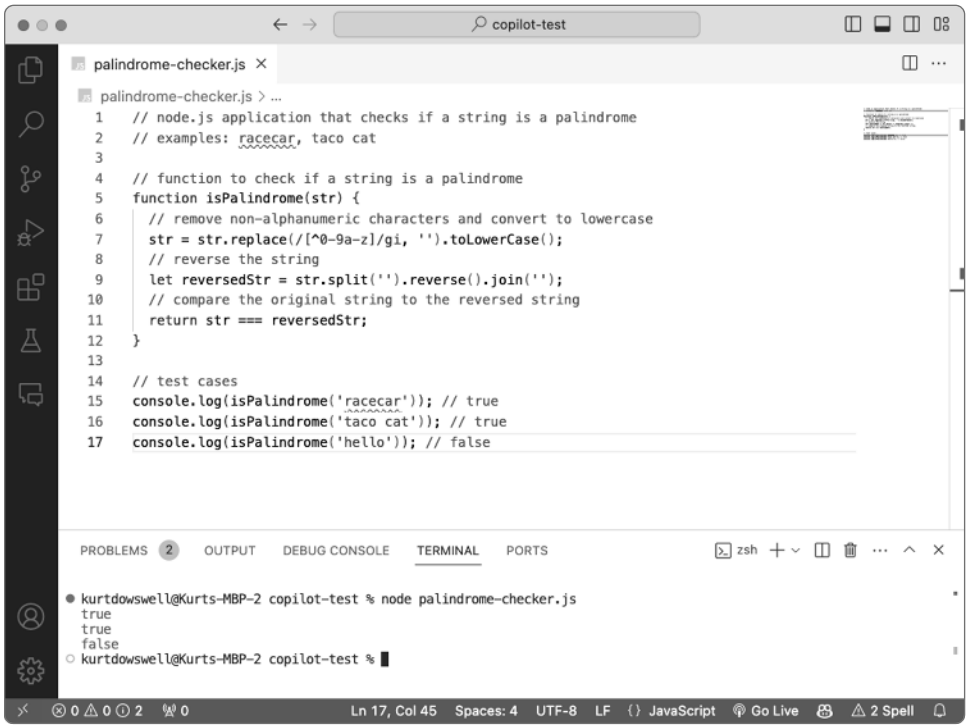

Figure 1.11: Palindrome function terminal run with results

Conclusion

There are a multitude of ways in which GitHub Copilot will assist you in your developer workflow. So far, we have only scratched the surface of the capabilities Copilot provides. Up next, you will learn more about GitHub Copilot code suggestions, what data sources and training it uses, and privacy and security considerations that are important to know.

Reference

[1] E. Kalliamvakou, 2022. "Research: quantifying GitHub Copilot's impact on developer productivity and happiness," `https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness`