Interact with Copilot

2 minutes

This unit explores ways that you can maximize your interaction with GitHub Copilot in your development environment. By understanding the service's features and capabilities, you learn how to use it effectively.

The following sections describe the various ways to trigger and use GitHub Copilot, along with examples and shortcuts to help you get the most out of it.

Inline suggestions

Inline suggestions are the most immediate form of assistance in Copilot. As you type, Copilot analyzes your code and context to offer real-time code completions. This feature predicts what you might want to write next and displays suggestions in a subtle, unobtrusive way.

The suggestions that Copilot offers appear as grayed-out text ahead of your cursor.

- To accept a suggestion, select the `Tab` key or the `>` (right arrow) key.
- To reject a suggestion, keep typing or select the `Esc` key.

Inline suggestions are especially useful when you're working on repetitive tasks or you need quick boilerplate code.

Here's an example:

Python

```
def calculate_average(numbers):
    # Start typing here and watch Copilot suggest the function body
```

Command palette

The command palette provides quick access to the various functions in Copilot, so you can perform complex tasks with only a few keystrokes.

1. Open the command palette in Visual Studio Code by selecting `Ctrl+Shift+P` (Windows or Linux) or `Cmd+Shift+P` (Mac).

2. Enter **Copilot** to see available commands.
3. Select actions like **Explain This** or **Generate Unit Tests** to get assistance.

Copilot chat

Copilot chat is an interactive feature that enables you to communicate with Copilot by using natural language. You can ask questions or request code snippets, and Copilot provides responses based on your input.

1. Open the Copilot chat panel in your IDE.
2. Enter questions or requests in natural language, and then evaluate the Copilot response.

For example, you might enter: "How do I implement a binary search in Python?" Copilot chat is ideal for exploring new coding concepts or getting help with unfamiliar syntax.

Copilot might respond with:

Python

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
```

Inline chat

Inline chat enables context-specific conversations with Copilot directly within your code editor. You can use this feature to request code modifications or explanations without switching contexts.

1. Place your cursor where you want assistance.
2. Use the keyboard shortcut `Ctrl+I` (Windows or Linux) or `Cmd+I` (Mac) to open inline chat.
3. Ask questions or request changes specific to that code location.

Inline chat helps you focus on a specific section of your code and receive targeted advice. Additionally, you can utilize slash commands for more efficient interaction.

Slash commands are shortcuts that allow you to quickly perform actions in Copilot. These commands provide a convenient way to interact with Copilot without needing to navigate through menus.

Here are some common slash commands and their usage:

- `/explain` - Provides an explanation of the selected code.
- `/suggest` - Offers code suggestions based on the current context.
- `/tests` - Generates unit tests for the selected function or class.
- `/comment` - Converts comments into code snippets.

To use a slash command, just type the command in your editor and press `Enter`. For example:

Python

```python
# Select the function, use the shortcut to open the inline chat, and type:
/explain
def calculate_average(numbers):
```

Comments to code

Copilot uses natural language processing to convert comments into code. You can describe the functionality that you want in a comment. When you select the `Enter` key, Copilot generates code based on your description.

Here's an example:

Python

```python
# Function to reverse a string
def reverse_string(s):
    # Copilot suggests the function body here
```

Python

```python
## Function to reverse a string
def reverse_string(s):
    return s[::-1]
```

This approach is beneficial for drafting code quickly, especially when your task is straightforward.

Multiple suggestions

For complex code snippets, Copilot can offer multiple alternatives.

1. When Copilot offers a suggestion, look for the light bulb icon.
2. Select the icon or use `Alt+]` (Windows/Linux) or `Option+]` (Mac) to cycle through alternatives.

Multiple suggestions help you explore different coding approaches and select the most appropriate one.

Explanations

Understanding existing code is crucial, especially in large projects. You can use the **Explain This** feature to get explanations for code snippets.

1. Select a block of code.
2. Right-click the code block, and then select **Copilot: Explain This** on the shortcut menu.
3. Read the explanation that Copilot provides for the selected code.

This feature is useful for learning purposes and when you're reviewing code that someone else wrote.

Automated test generation

Unit tests are essential for ensuring code quality and reliability. Copilot can save you time and effort by generating unit tests for your functions or classes.

1. Select a function or class.
2. Use the command palette to select **Copilot: Generate Unit Tests**.
3. Review the test cases that Copilot suggests for your code.

Here's an example:

Python

```python
def add(a, b):
    return a + b

# Copilot might generate a test like this:
def test_add():
    assert add(2, 3) == 5
    assert add(-1, 1) == 0
    assert add(0, 0) == 0
```

Automated test generation helps you maintain code integrity and catch bugs early in the development process.

Keep in mind that Copilot learns from context. Keeping your code well structured and commented helps Copilot provide more accurate and relevant assistance. The more you interact with Copilot, the better it becomes at understanding your coding style and preferences.

Next unit: Set up, configure, and troubleshoot GitHub Copilot

Previous Next