F-07 - Chat Context Anchors (file, selection, terminal...)

SUMMARY

In this lesson, we explored the concept of **Chat Context Anchors** in GitHub Copilot, which enhance our ability to provide context to Copilot by specifying which files or data it should consider when generating responses. This capability helps in getting more accurate answers by extending the context beyond the currently open file.

1. **File Anchors:** We learned how to use file anchors by appending `#file:filename.py` in our query to Copilot, allowing it to consider multiple files for comprehensive explanations.
2. **Folder Anchors:** We discovered how to right-click and select "add folder to chat" to include an entire folder in the context. This is beneficial when the answer lies across multiple files.
3. **Selection Anchors:** We can select specific code blocks and use `#selection` to narrow down the focus to a specific piece of code.
4. **Terminal Anchors:** By using `#terminal last command` , Copilot can access the last input and output from the terminal, helping resolve issues directly from terminal feedback.
5. **Symbol Anchors:** By typing `#symbol_name` , we can refer to specific functions or symbols and instruct Copilot to compare or explain them.

WHAT WE LEARNED

- Adding file-based context to GitHub Copilot using file anchors.
- Utilizing folder anchors to broaden the context scope.
- Selecting specific code blocks and employing selection anchors.
- Integrating terminal feedback with terminal anchors.
- Comparing specific functions or symbols using symbol anchors.

HOW WE CAN APPLY IT

- **Debugging:** Quickly identify and resolve issues by providing Copilot with wider context.
- **Code Comprehension:** Understand large codebases by summoning detailed explanations of intertwined functions.

- **Optimization:** Compare and analyze functions to optimize performance.
- **Automation:** Automate repetitive tasks or error corrections using terminal anchors.

TIPS AND TRICKS

- Use file anchors when dealing with code that spans across multiple modules.
- Keep the context concise; too many details might overwhelm Copilot, leading to less accurate results.
- Regularly save your work and changes suggested by Copilot for easy rollback if necessary.
- Utilize selection anchors when you need specific insights about a certain piece of code.

EXAMPLES

Using File Anchor

```
# In your query, append:
# file:data_processor.py
```

This instructs Copilot to consult `data_processor.py` alongside the current file.

Utilizing Folder Anchor

Right-click on the desired folder in VS Code and select:

```
Copilot > Add folder to chat
```

This includes the entire folder in the Copilot context.

Applying Terminal Anchor

```
#terminal last command
```

Adds the last terminal input/output to the context, assisting Copilot in diagnosing errors.

Example Function Comparison using Symbol Anchor

```
#symbol:load_data
```

```
# compare load_data with get_total_value_by_category
```

This command compares two specific functions, helping to elucidate their purposes and performance.

These techniques significantly enhance our ability to use GitHub Copilot effectively, turning it into a powerful assistant that can streamline our development processes.