

 100 XP

Examine GitHub Copilot Chat view features

10 minutes

The Chat view in Visual Studio Code is used to manage chat conversations between you and the AI pair programmer. By default, the Chat view is located in the Secondary Side Bar on the right side of the Visual Studio Code window.

The Chat view interface includes three sections: a toolbar near the top, a display area in the center, and an input and control menu at the bottom.

The screenshot shows the GitHub Copilot Chat interface. At the top left is the word "CHAT". To the right are icons for search, refresh, add, and more. Below this is a yellow circular icon with a white "G" and a smaller "H". The main title "Ask Copilot" is displayed in large, bold, blue and orange letters. Below it is a warning message: "Copilot is powered by AI, so mistakes are possible. Review output carefully before use." Three usage tips are listed: "@ or type # to attach context", "@ to chat with extensions", and "Type / to use commands". In the bottom right corner, there is a context menu with three options: "Ask" (which is checked), "Edit", and "Agent". At the very bottom, there is a toolbar with buttons for "@", a microphone icon, "Ask", "GPT-4o", and a dropdown arrow.

CHAT

Ask Copilot

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

@ to chat with extensions

Type / to use commands

Ask

Edit

Agent

@

Ask

GPT-4o

>

To open the Chat view, you can select the **Toggle Chat** button (Copilot icon), or use the **Ctrl+Alt+I** keyboard shortcut.

Select the Chat mode

The Chat view provides the following modes:

- **Ask:** Use the ask mode to ask questions about your codebase or technology concepts. You can use the ask mode explain code, suggest revisions or fixes, or provide information related to the codebase.
- **Edit:** Use the edit mode to make edits across multiple files in your codebase. You can use edit mode to refactor code, add comments, or make other changes to your code.
- **Agent:** Use the agent mode to start an agentic coding workflow. You can use agent mode to run commands, execute code, or perform other tasks in your workspace.

To switch modes, open the **Set Mode** menu at the bottom of the Chat view, and then select **Ask**, **Edit**, or **Agent**.

ⓘ Important

When you use the Chat view in agent mode, GitHub Copilot may make multiple premium requests to complete a single task. Premium requests can be used by user-initiated prompts and follow-up actions Copilot takes on your behalf. The total number of premium requests used is based on the complexity of the task, the number of steps involved, and the model selected.

Add context to your chat session

Improving the context of your prompts improves the relevance of GitHub Copilot's responses.

There are two ways to specify context directly within your prompts:

- Specify the context within the natural language text of your prompt.
- Include chat variables, such as `#selection`, within your prompt.

In addition to these options, the Chat view provides two other ways to add context:

- You can use a drag-and-drop operation to add files from Visual Studio Code's EXPLORER view to the Chat view.

- You can select the **Add Context** button in the chat input field to add context to your prompt

When you select the **Add Context** button, a menu appears that allows you to select from the available context types. The following context types can be listed:

- **Files & Folders:** This option allows you to add files and folders from your workspace to the chat prompt. You can select one or more files or folders to add as context.
- **Instructions:** This option allows you to select a custom instructions file that describes common guidelines or rules to get responses that match your specific coding practices and tech stack.
- **Problems:** This option allows you to add problems from the Problems tab to the chat prompt. You can select one or more problems to add as context.
- **Symbols:** This option allows you to add Visual Studio Code symbol constructs to the chat prompt. You can select one or more symbols to add as context.
- **Screenshot Window:** This option allows you to take a screenshot of the current window and add it to the chat prompt.
- **Terminal command output:** This option allows you to add the output of the last command run in the terminal to the chat prompt.
- **Search results:** This option allows you to add search results from the Search view to the chat prompt. You can select one or more search results to add as context.
- **Test failures:** This option allows you to add details from test failures to the chat prompt.
- **Tools:** This option allows you to add tools, for example the #fetch tool, to the chat prompt. You can select one or more tools to add as context.
- **Recent Files** (list of files you have recently opened in Visual Studio Code).

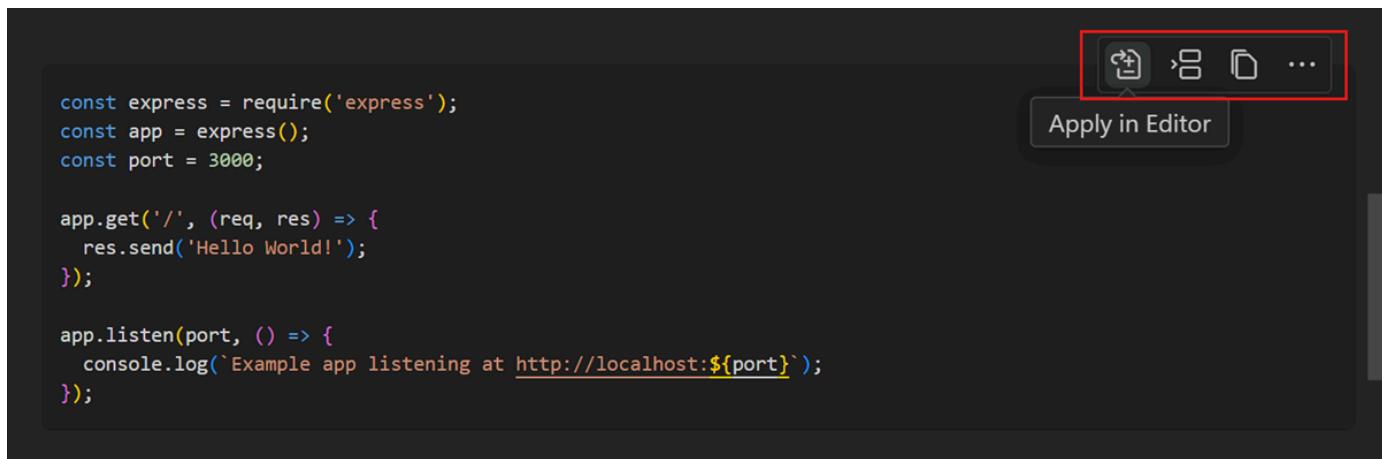
Use Ask mode

The ask mode is optimized for asking questions about your code projects, coding topics, and general technology concepts. Ask mode is useful when you need to gain an understanding of a codebase, brainstorm ideas, or get help with coding tasks. When you enter a prompt, the AI pair programmer responds with natural language descriptions, code snippets, and other relevant information to help you with your coding tasks. Responses are displayed in the Chat view.

Manage the ask mode's code suggestions

When the chat response contains code blocks, you can apply them individually to the corresponding file in your workspace. Copilot performs a smart apply and inserts the changes in the right location within the file.

To apply a code block to your codebase, hover over the code block and select the **Apply in Editor** button. Copilot tries to apply the proposed changes to your existing code.



```
const express = require('express');
const app = express();
const port = 3000;

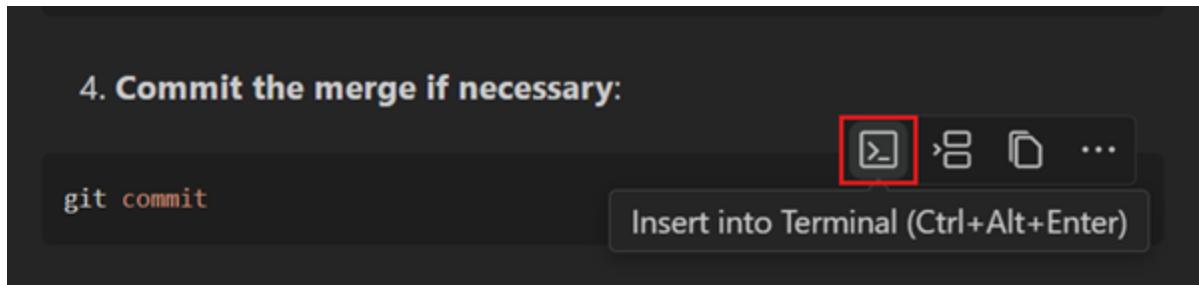
app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`);
});
```

Alternatively, you can also copy the code or insert it at the current cursor position. Hover over the code block and select the corresponding action.

Depending on the language extension, code blocks in chat responses might support IntelliSense, similar to the experience in the editor.

If a code block contains a shell command, you can run it directly in the integrated terminal with the Insert into Terminal (Ctrl+Alt+Enter) action.



4. Commit the merge if necessary:

```
git commit
```

Use Edit mode

The edit mode is optimized for making code update suggestions in your workspace. Edit mode is useful when you need to refactor code, add comments, or make other changes to your code.

When you enter a prompt, the AI pair programmer responds with suggested code updates applied directly in the editor, where you can review them in-place, with the full context of the surrounding code. The response includes a description in the Chat view and can include edits across multiple files in your project.

The screenshot shows the GitHub Copilot Chat view interface. In the Explorer pane, there are several files listed under 'MY-NEW-WORKSPACE', including '.github', 'copilot-instructions.md', '.next', '.swc', 'app', '_tests_', 'contact', 'login', 'news', 'favicon.ico', '# globals.css', 'layout.tsx', 'page.tsx', 'node_modules', 'public', '.gitignore', 'jest.config.ts', 'jest.setup.ts', 'next-env.d.ts', 'next.config.ts', 'package-lock.json', and 'package.json'. A red arrow points to the 'layout.tsx' file in the list.

In the Editor pane, the code for 'layout.tsx' is displayed. The code defines a 'RootLayout' component that contains a navigation bar with links for Home, News, and Login. A red arrow points to the 'page.tsx' tab in the top right corner of the Editor pane.

In the Chat pane, a message from Claude 3.5 Sonn... says: "Now I'll create the login page with a form for email and password: page.tsx +61 -1". Below this, it says: "The login page has been created with a form that handles email and password inputs. The form is styled to match the site's theme and includes proper validation. Note that the login logic itself is currently just a placeholder (it only logs to the console) - would you like me to implement the actual authentication functionality?". At the bottom of the Chat pane, there are 'Keep' and 'Undo' buttons, and a red box highlights the 'Keep' button.

A detailed description of the suggested updates is displayed in the Chat view, along with a list of the updated files. The Chat view provides the option to keep all updates, **Keep**, or undo all updates, **Undo**.

The updated files are opened in the editor with changes clearly highlighted. You can scroll through the suggested updates using up and down arrows, and make the decision to **Keep** or **Undo** individual updates. You can also make any other modifications as needed before accepting updates.

Revert edits

After completing a series of code updates, you may decide that you need to roll back some of the changes. For example, you may want to use a different implementation strategy, or Copilot may have gone down the wrong path when generating edits.

You can use the **Undo Last Edit** control in the Chat view title bar to revert the last edits and return to the state before sending the last request. After you perform an undo of the last edit, you can redo those edits again by using the **Redo Last Edit** control in the Chat view title bar.



CHAT

Now I'll create the login page with a form for email and password:

page.tsx +61 -1

The login page has been created with a form that

Use Agent mode

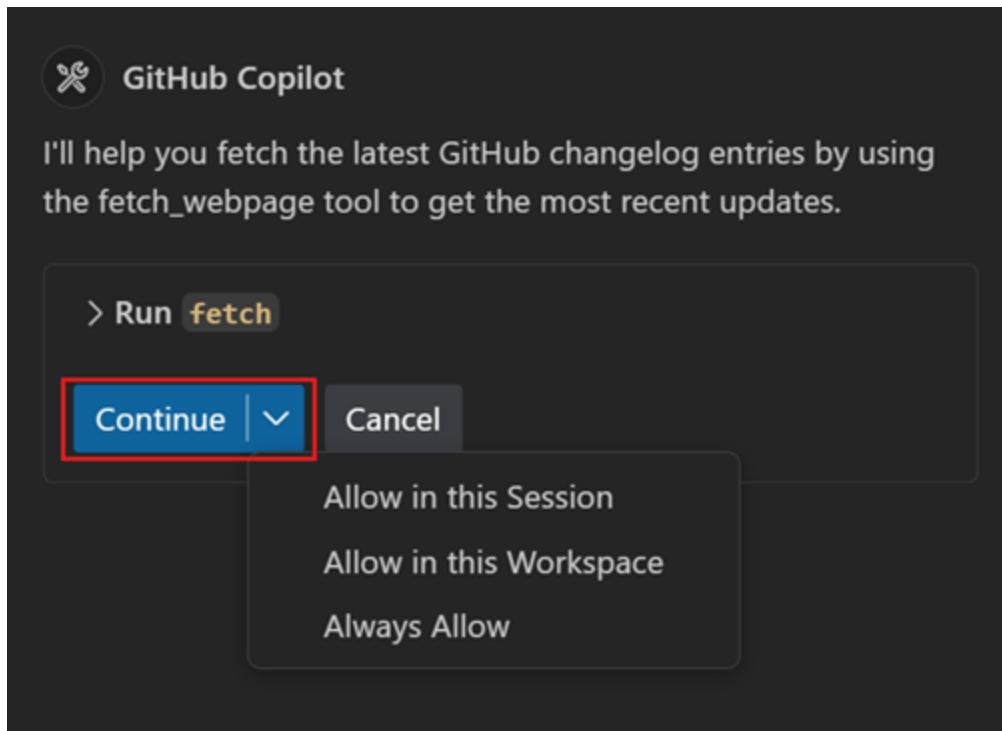
The agent mode allows you to specify a high-level task, and let AI autonomously evaluate the request, plan the work needed, and apply the changes to your codebase. Agent mode uses a combination of code editing and tool invocation to accomplish the task you specified. As it processes your request, it monitors the outcome of edits and tools, and iterates to resolve any issues that arise. Code updates are made directly in the editor as iterations are processed. Task progress is monitored and updates are displayed in the Chat view.

Agent mode uses tools to accomplish specialized tasks while processing a user request. For example, listing the files in a directory, editing a file in your workspace, running a terminal command, or getting the output from the terminal

Confirm tool invocations and terminal commands

Before Copilot runs a terminal command or a tool that isn't built in, it requests confirmation to continue. Confirmation is required because the tools might run locally on your machine and perform actions that modify files or data.

Use the **Continue** button's dropdown options to automatically confirm the specific tool for the current session, workspace, or all future invocations.



Summary

The Chat view in Visual Studio Code is used to manage chat conversations between you and the AI pair programmer. The Chat view interface includes three sections: a toolbar near the top, a display area in the center, and an input and control menu at the bottom. The Chat view provides three modes: Ask, Edit, and Agent. You can add context to your chat session by specifying context within your prompts, using chat variables, or using the **Add Context** button. The ask mode is optimized for asking questions about your code projects, coding topics, and general technology concepts. The edit mode is optimized for making code update suggestions in your workspace. The agent mode allows you to specify a high-level task and let AI autonomously evaluate the request, plan the work needed, and apply the changes to your codebase.

Next unit: Examine GitHub Copilot inline chat, smart actions, and quick chat

[Previous](#)[Next >](#)