GitHub Copilot VS Code Settings Reference

Comprehensive reference for all GitHub Copilot settings in Visual Studio Code, organized by category with practical examples and use cases.

🎯 General Copilot Settings

Core Functionality

```
{
  // Enable or disable Copilot completions globally or per language
  "github.copilot.enable": {
    "*": true,
    "plaintext": false,
    "markdown": false,
    "scminput": false,
    "yaml": false
  },

  // Automatically show inline completions
  "github.copilot.editor.enableAutoCompletions": true,

  // Show Copilot commands as Code Actions when available
  "github.copilot.editor.enableCodeActions": true,

  // Generate suggestions for renaming automatically
  "github.copilot.renameSuggestions.triggerAutomatically": true,

  // Enable semantic search results in the Search view
  "github.copilot.chat.search.semanticTextResults": true
}
```

Suggestion Display

```
{
  // Control inline suggestion appearance
  "github.copilot.advanced": {
    "debug.overrideEngine": "default",
    "debug.testOverrideProxyUrl": "",
    "debug.overrideProxyUrl": "",
```

```
    "length": 500,
    "inlineSuggestEnable": true,
    "listCount": 10
  }
}
```

💬 Chat Settings

Core Chat Configuration

```
{
  // Controls whether Copilot should suggest follow-up questions in chat
  "github.copilot.chat.followUps": "always", // "always", "never",
"afterAction"

  // Specify a locale that Copilot should respond in (e.g., "en", "fr",
"es")
  "github.copilot.chat.localeOverride": "",

  // Enable the /runCommand intent in the Chat view to run VS Code
commands
  "github.copilot.chat.runCommand.enabled": true,

  // Use relevant GitHub projects as starter projects when using /new
  "github.copilot.chat.useProjectTemplates": true,

  // Whether to prompt for a specific symbol scope if using /explain with
no selection
  "github.copilot.chat.scopeSelection": true,

  // Controls where chat queries from the terminal should be opened
  "github.copilot.chat.terminalChatLocation": "quickChat", // "quickChat",
"panel", "editor"

  // Include recently viewed and edited files with Copilot requests in
Inline Chat
  "github.copilot.chat.temporalContext.enabled": true
}
```

Chat Interface Customization

```
{
  // Controls whether the command center shows a menu for chat actions
```

```
  "chat.commandCenter.enabled": true,

  // Controls where the Command Palette should ask chat questions
  "workbench.commandPalette.experimental.askChatLocation": "quickChat",

  // Enable chat participant detection in the Chat view
  "chat.experimental.detectParticipant.enabled": true,

  // Font settings for chat codeblocks
  "chat.editor.fontFamily": "Consolas, 'Courier New', monospace",
  "chat.editor.fontSize": 14,
  "chat.editor.fontWeight": "normal",
  "chat.editor.lineHeight": 19,
  "chat.editor.wordWrap": "on"
}
```

Chat Participants Control

```
{
  // Control which chat participants are enabled
  "github.copilot.chat.participants.enabled": {
    "@workspace": true,
    "@vscode": true,
    "@terminal": true,
    "@github": true
  },

  // Priority ordering for participant suggestions
  "github.copilot.chat.participants.priority": [
    "@workspace",
    "@vscode",
    "@terminal",
    "@github"
  ]
}
```

✏️ Editing Session Settings

Copilot Edits Configuration

```
{
  // Ask for confirmation before undoing an edit
  "chat.editing.confirmEditRequestRemoval": true,
```

```
  // Ask for confirmation before performing a redo of the last edit
  "chat.editing.confirmEditRequestRetry": true,

  // Automatically save generated changes from Copilot Edits to disk
  "chat.editing.alwaysSaveWithGeneratedChanges": false,

  // Maximum number of files in working set
  "chat.editing.workingSet.maxFiles": 20,

  // Enable preview mode for editing sessions
  "chat.editing.enablePreview": true,

  // Auto-apply simple changes without confirmation
  "chat.editing.autoApplySimpleChanges": false
}
```

Working Set Management

```
{
  // Automatically include related files in working set
  "chat.editing.workingSet.autoIncludeRelated": true,

  // Include test files when adding source files to working set
  "chat.editing.workingSet.includeTests": true,

  // Include type definition files automatically
  "chat.editing.workingSet.includeTypes": true
}
```

💭 Inline Chat Settings

Inline Chat Behavior

```
{
  // Controls whether pending Inline Chat sessions prevent saving the file
  "inlineChat.acceptedOrDiscardBeforeSave": "auto", // "auto", "never",
"always"

  // Whether to finish an Inline Chat session when typing outside of
changed regions
  "inlineChat.finishOnType": true,
```

```
  // Whether holding the Inline Chat keybinding will automatically enable
speech recognition
  "inlineChat.holdToSpeech": false,

  // Controls whether to show a hint for Inline Chat on an empty line
  "inlineChat.inlineChat.lineEmptyHint": true,

  // Configure if changes are applied directly or previewed first
  "inlineChat.mode": "preview", // "preview", "live"

  // Experimental suggestion that triggers Inline Chat for natural
language
  "inlineChat.lineNaturalLanguageHint": false
}
```

🎨 Custom Instructions

Code Generation Instructions

```
{
  // Instructions for code generation requests
  "github.copilot.chat.codeGeneration.instructions": "Always use
TypeScript with strict type checking. Include comprehensive error handling
and JSDoc comments. Follow functional programming patterns when possible.
Use descriptive variable names without abbreviations.",

  // Controls whether to use instruction files from .github/copilot-
instructions.md
  "github.copilot.chat.codeGeneration.useInstructionFiles": true
}
```

Test Generation Instructions

```
{
  // Instructions for test generation requests
  "github.copilot.chat.testGeneration.instructions": "Use Jest as the
testing framework. Follow AAA pattern (Arrange, Act, Assert). Create
comprehensive test suites with positive, negative, and edge cases. Use
descriptive test names. Mock external dependencies using jest.mock().",

  // Enable the experimental /setupTests intent and prompting in /tests
generation
```

```
  "github.copilot.chat.setupTests.enabled": true
}
```

Code Review Instructions

```
{
  // Instructions for code review requests
  "github.copilot.chat.reviewSelection.instructions": "Focus on security
vulnerabilities (OWASP Top 10), performance optimization opportunities,
proper error handling, SOLID principles adherence, accessibility
compliance (WCAG 2.1 AA), and maintainability.",

  // Instructions for commit message generation
  "github.copilot.chat.commitMessageGeneration.instructions": "Follow
Conventional Commits format: type(scope): description. Use present tense
and imperative mood. Keep first line under 50 characters. Reference issue
numbers when applicable."
}
```

🧪 Testing Integration

Test-Related Settings

```
{
  // Show Generate tests code lens for symbols not covered by tests
  "github.copilot.chat.generateTests.codeLens": true,

  // Enable the preview /fixTestFailure intent in chat
  "github.copilot.chat.fixTestFailure.enabled": true,

  // Automatically run tests after generating them
  "github.copilot.chat.testing.autoRun": false,

  // Preferred testing framework
  "github.copilot.chat.testing.framework": "jest" // "jest", "mocha",
"vitest", "cypress"
}
```

🛠️ Debugging Settings

Debug Integration

```
{
  // Enable the experimental /startDebugging intent in the Chat view
  "github.copilot.chat.startDebugging.enabled": true,

  // Automatically generate debug configurations
  "github.copilot.chat.debug.autoGenerateConfig": true,

  // Include debug context in chat requests
  "github.copilot.chat.debug.includeContext": true
}
```

📓 Notebook Settings

Jupyter Notebook Integration

```
{
  // Enable the Generate action to create code cells with Inline Chat
  "notebook.experimental.generate": true,

  // Enable Copilot completions in notebook cells
  "github.copilot.notebook.enableCompletions": true,

  // Include notebook context in chat requests
  "github.copilot.notebook.includeContext": true
}
```

♿ Accessibility Settings

Accessibility Features

```
{
  // Whether the Inline Chat also renders an accessible diff viewer
  "inlineChat.accessibleDiffView": "auto", // "auto", "on", "off"

  // Audio signals for chat interactions
```

```
  "accessibility.signals.chatRequestSent": {
    "sound": "on",
    "announcement": "on"
  },
  "accessibility.signals.chatResponseReceived": {
    "sound": "on",
    "announcement": "on"
  },

  // Voice interaction settings
  "accessibility.voice.keywordActivation": true,
  "accessibility.voice.autoSynthesize": "on", // "on", "off", "auto"
  "accessibility.voice.speechTimeout": 1200,

  // Verbosity settings for screen readers
  "accessibility.verbosity.inlineChat": true,
  "accessibility.verbosity.inlineCompletions": true,
  "accessibility.verbosity.panelChat": true
}
```

🏢 Enterprise Settings

Organization-Level Configuration

```
{
  // Enterprise policy compliance
  "github.copilot.enterprise.policies": {
    "enableCodeCompletion": true,
    "enableChatFeatures": true,
    "enableEditingSessions": "restricted", // "enabled", "disabled",
"restricted"
    "allowPublicRepositories": false
  },

  // Audit and compliance
  "github.copilot.enterprise.audit": {
    "enableLogging": true,
    "logLevel": "info", // "debug", "info", "warn", "error"
    "includeCodeSnippets": false,
    "retentionDays": 90
  }
}
```

Team Collaboration

```
{
  // Shared team settings
  "github.copilot.team.sharedInstructions": true,

  // Enforce consistent coding standards
  "github.copilot.team.enforceStandards": {
    "codeStyle": true,
    "namingConventions": true,
    "documentationRequirements": true
  }
}
```

⚡ Performance Settings

Optimization Configuration

```
{
  // Request timeout settings
  "github.copilot.advanced.timeout": {
    "completion": 5000,
    "chat": 10000,
    "editing": 15000
  },

  // Cache settings
  "github.copilot.advanced.cache": {
    "enabled": true,
    "maxSize": "100MB",
    "ttl": 3600
  },

  // Resource usage limits
  "github.copilot.advanced.resources": {
    "maxConcurrentRequests": 5,
    "maxMemoryUsage": "500MB"
  }
}
```

🔒 Privacy and Security Settings

Data Protection

```
{
  // Control data transmission to Copilot service
  "github.copilot.privacy": {
    "blockSensitiveData": true,
    "excludeFilePatterns": ["*.env", "*.key", "*.pem", "secrets/**"],
    "requireExplicitConsent": true
  },

  // Telemetry and analytics control
  "github.copilot.telemetry": {
    "enabled": false,
    "includeUsageData": false,
    "includePerformanceData": false
  }
}
```

🎛️ Advanced Configuration Examples

Development Environment Setup

```
{
  // React/TypeScript project optimized settings
  "github.copilot.enable": {
    "*": true,
    "javascript": true,
    "typescript": true,
    "javascriptreact": true,
    "typescriptreact": true,
    "json": false,
    "jsonc": false
  },

  "github.copilot.chat.codeGeneration.instructions": "Use React functional
components with hooks. Prefer TypeScript with strict mode. Include
PropTypes or TypeScript interfaces. Use CSS modules or styled-components.
Follow React best practices for performance.",

  "github.copilot.chat.testGeneration.instructions": "Use React Testing
Library and Jest. Test component behavior, not implementation. Include
```

```
accessibility tests. Mock external dependencies."
}
```

Backend API Development

```
{
  // Node.js/Express API optimized settings
  "github.copilot.chat.codeGeneration.instructions": "Use Express.js with
TypeScript. Implement proper error handling middleware. Use async/await
for asynchronous operations. Include input validation with Joi or Zod.
Follow RESTful API design principles. Add comprehensive logging.",

  "github.copilot.chat.reviewSelection.instructions": "Focus on API
security, input validation, error handling, and performance. Check for SQL
injection, XSS, and authentication issues. Verify proper HTTP status codes
and response formats."
}
```

Data Science/ML Projects

```
{
  // Python data science optimized settings
  "github.copilot.enable": {
    "python": true,
    "jupyter": true,
    "markdown": true
  },

  "github.copilot.chat.codeGeneration.instructions": "Use pandas for data
manipulation, matplotlib/seaborn for visualization, and scikit-learn for
ML. Include proper error handling and data validation. Add docstrings
following NumPy style. Use type hints where appropriate."
}
```

🔧 Troubleshooting Settings

Common Issues Resolution

```
{
  // Enable debug logging
  "github.copilot.advanced.debug": {
```

```
    "enabled": true,
    "logLevel": "debug",
    "logToFile": true
  },

  // Network and connectivity
  "github.copilot.advanced.network": {
    "timeout": 10000,
    "retryAttempts": 3,
    "useProxy": false,
    "proxySettings": {
      "host": "",
      "port": 0,
      "authentication": false
    }
  }
}
```

📊 Workspace-Specific Settings

Project-Level Configuration

Create a `.vscode/settings.json` file in your project root:

```
{
  // Project-specific Copilot configuration
  "github.copilot.chat.codeGeneration.instructions": "Follow this
project's specific coding standards defined in CONTRIBUTING.md. Use the
established error handling patterns. Maintain consistency with existing
API design.",

  "github.copilot.enable": {
    "*": true,
    "sql": false  // Disable for sensitive database files
  },

  "chat.editing.workingSet.maxFiles": 15,  // Smaller working sets for
this project

  "github.copilot.chat.testGeneration.instructions": "Use the project's
custom test utilities. Follow the established test file naming convention.
Include integration tests for API endpoints."
}
```

Remember: Settings hierarchy follows VS Code's standard precedence - workspace settings override user settings, which override default settings.