GitHub Copilot活用（公開版）

サイボウズ 開運研修 2025

開発本部 AIやっていきチーム

加瀬健太（@Kesin11）

講義のコンセプト

- 誰に
  - これから業務でプログラミングを行う開発者
- 何と言ってほしいか
  - 業務でAIを活用する際に気をつけることが分かった
  - GitHub Copilotにどういう機能があるか分かった

目次

自己紹介

- 加瀬健太（@Kesin11）
- 開発本部 AIやっていきチーム
- 普段の業務
  - ChatGAI（社内のChatGPTクローン）運用
  - CursorやDifyなどのAI関連ツールのアカウント管理・運用
  - Kintone AI のR&D協力
- 趣味
  - github.blog/changelogを毎朝見ること
  - VSCodeのリリースノートを毎月見ること
  - 少なくとも2021年から続いている趣味です

@Kesin11

@Kesin11

@kesin11.bsky.social

プログラミングに
AIを活用してますか？

使っているAIツール/サービスを
実況スレに書いてみてください！

（1分待ちます）

サイボウズのGitHub Copilot

- エンジニアには基本的にGitHub Copilot Businessのライセンスが付与されている
  - FreeやProとの違いは、一部の設定がCybozuのGitHub管理者によって強制されていること
  - github.com/settings/copilot から確認できる
- 新機能もなるべく早く利用できるようにAIやっていきチーム、生産性向上チーム、情シスが連携しています

業務でAIを利用する際に
気をつけてほしいこと

業務でAIを利用する際に気をつけてほしいこと

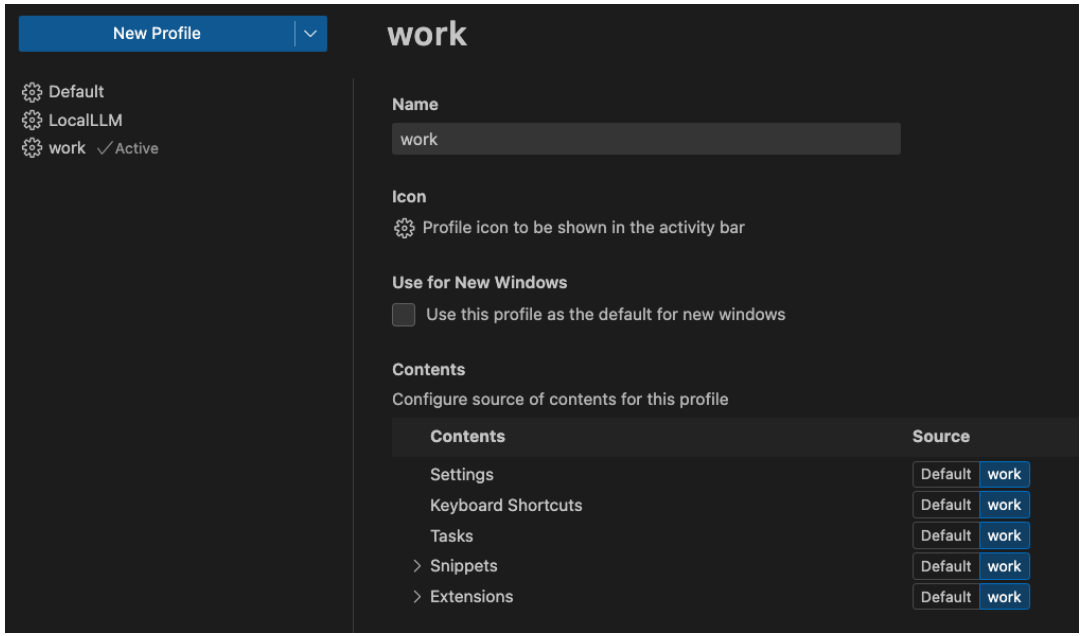
- GitHub Copilotに限らず、AIツール/サービス全般で重要な観点
- 基本的に渡したデータが **AIの学習に利用される** かどうかが大きな争点
  - AIの学習に利用される = そのサービスのユーザーがAIから情報を引き出せる
  - 社外秘のデータがAI学習されてしまうと情報流出に等しい

業務でAIを利用する際に気をつけてほしいこと

- 一般的にBusinessとかEnterpriseのような名前が付くサービスは学習に利用されないことが規約に明記されていることが多い
  - GitHub Copilot Businessは学習に利用されないことが規約に明記されている
- 逆にそれ以外のサービスはオプトアウト方式であったり、不明だったり、拒否できないこともある
  - 学習を拒否できないサービスは基本的に業務利用は厳しい
  - 規約に明記されていない「不明」も危険として考えるべき
  - オプトアウト方式もヒューマンエラーが発生しうるので危険として考えるべき
    - BusinessやEnterpriseプランだとオプトアウトを強制できることが多いのでこれは安全

業務でAIを利用する際に気をつけてほしいこと

- プライベートで利用しているAIツールがいつのまにか業務PCにインストールされている、ということも気をつける
  - 例：VSCode Extension、Chrome Extension、dotfiles共有 + Homebrew
  - アカウントで別PCと設定が共有されるものは要注意



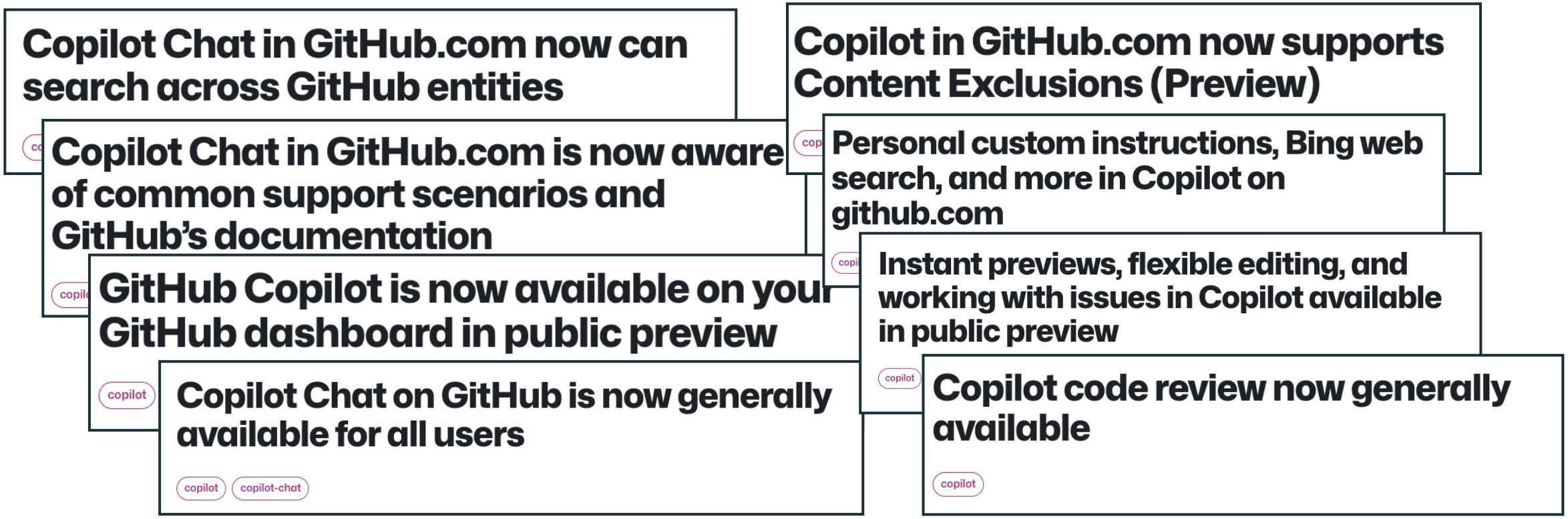
例：VSCodeのプロファイルをDefaultとは分ける
最低でもExtensionsはDefaultと独立させる

例：Chromeの同期設定で拡張機能は外す

GitHub Copilot

github.com 内で使えるGitHub Copilotの機能

- 色々なところでCopilotが使える。ありすぎてもう分からないぐらい
  - https://docs.github.com/en/copilot/using-github-copilot/copilot-chat/asking-github-copilot-questions-in-github

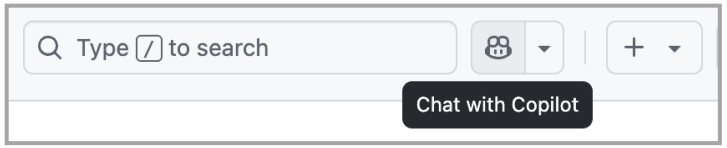


ハンズオン

（眠気覚ましに！）

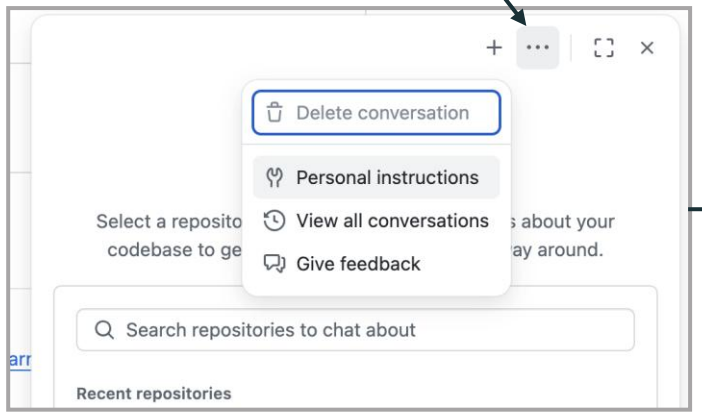
まず日本語で応答してくれるようにCopilotをカスタムしよう（2分）

- デフォルトは基本的に英語で応答される
- Adding personal custom instructions for GitHub Copilot
  - https://docs.github.com/en/copilot/customizing-copilot/adding-personal-custom-instructions-for-github-copilot
- 設定がかなり深いところにあるので普通は気が付かない・・・

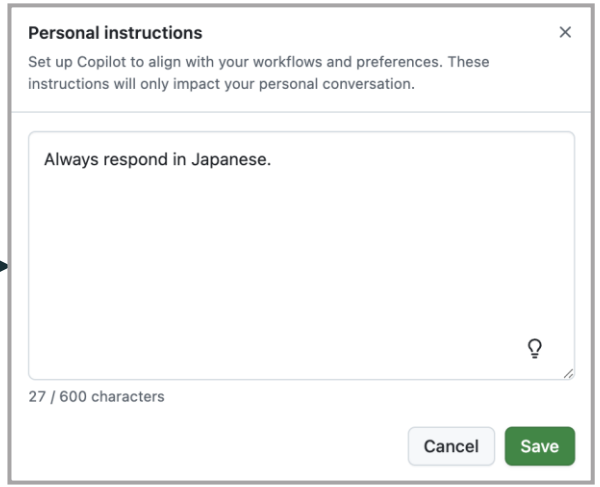


① ヘッダーからCopilotのポップアップを開く

② 3点リーダーをクリック



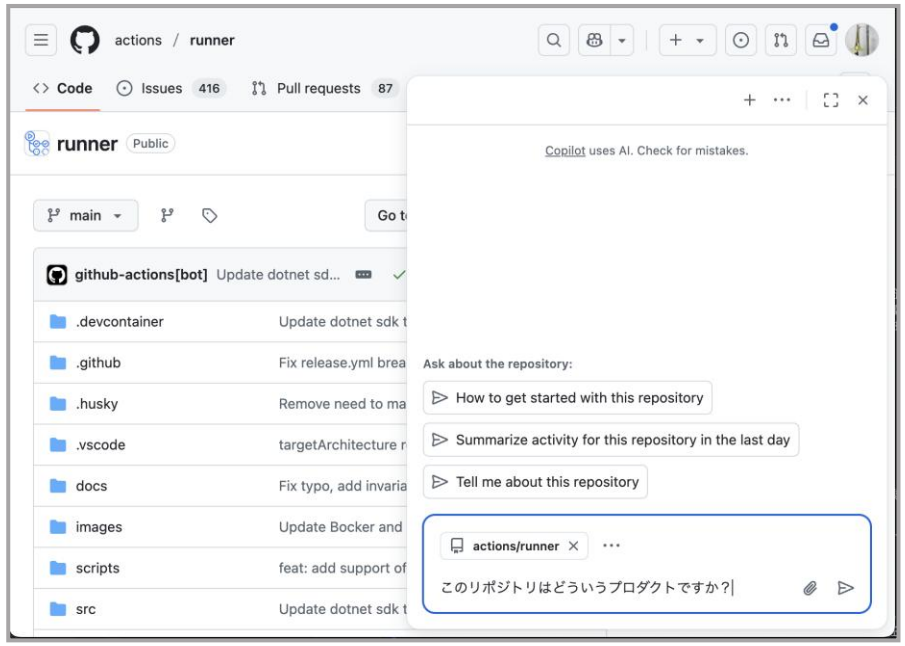
③Personal instructionsを開く



④例：”Always respond in Japanese.”

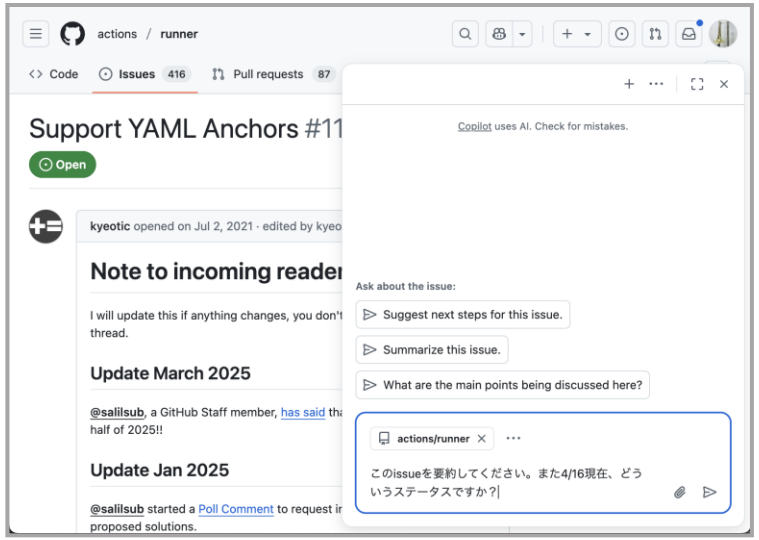
リポジトリについて質問してみよう（1分）

- 好きな社内 or OSSのリポジトリにアクセス
- ヘッダーのGitHub Copilotのマークからチャットを呼び出し
- 「このリポジトリはどういうプロダクトですか？」と質問
- やってみよう！



長いissueを要約してもらおう（2分）

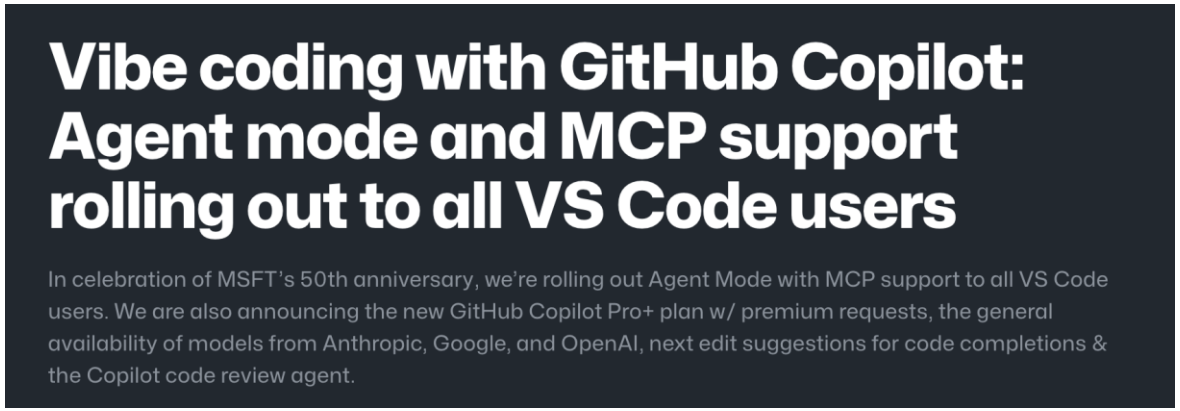
- https://github.com/actions/runner/issues/1182
  - 2021年に作られたissueで120コメント以上付いており、まだOpen
  - 完全に放置されているのか、今後実装される予定があるのかを把握したい
- ヘッダーのGitHub Copilotマークからチャットを呼び出し
- 「このissueを要約してください。また2025/04/16現在、どういうステータスですか？」
- **Copilotが嘘を言う可能性はゼロではないので、コメントを検索するなど裏取りは必ず行うこと**



VSCode内で使える
GitHub Copilotの機能

その前に：他のエディタについて

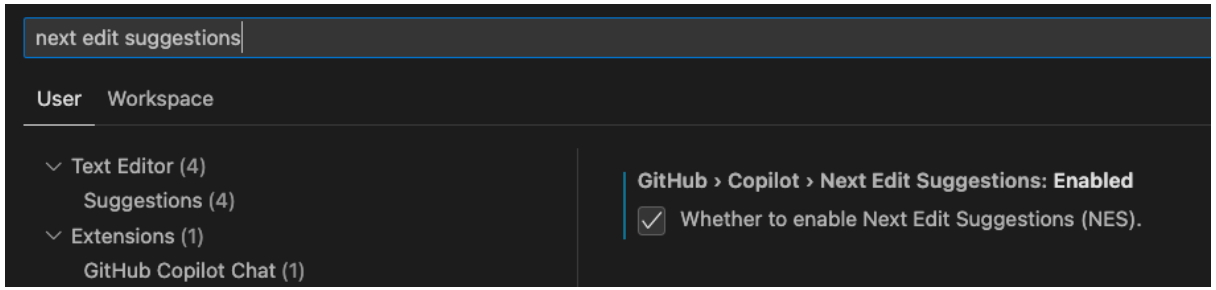
- VSCodeが一番GitHub Copilotとの統合が進んでいるのでVSCodeで紹介します
- IntelliJ、XcodeにもGitHub Copilotのプラグインはあります
  - （使ったことないので詳しくは分からない・・・）
- CursorはGitHub Copilotと同等の機能があり、2025/04現在ではVSCodeよりも若干先行している
  - ちょうど4月末からサイボウズでもCursorの本運用が始まる予定
  - でもVSCode + GitHub Copilotもかなり追いついたので、1ヶ月ぐらいVSCodeを試してみてもらえると嬉しい



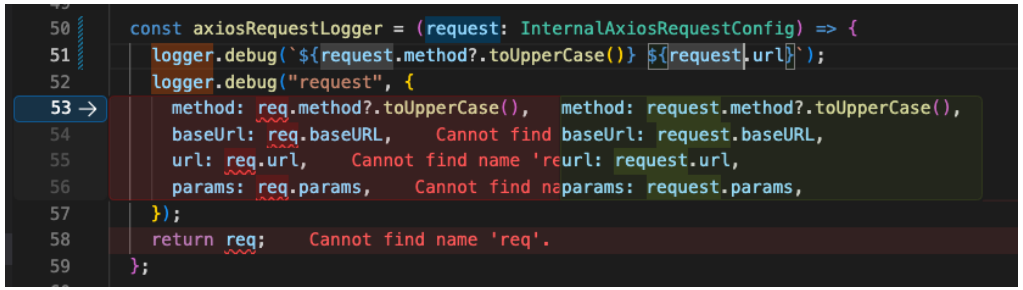
2025/04/04: https://github.blog/news-insights/product-news/github-copilot-agent-mode-activated/

GitHub Copilot

- いわゆるタブ補完
  - 編集地点の前後のコードが参考にされている
  - クラスや関数の実装前にあらかじめコメントを書いておくと補完が賢くなる
- Copilot Next Edit Suggestions（NES）という新機能を有効にすると、1箇所変更すると関連するコードの修正も自動で提案してくれるようになる
  - 賢いタブ補完で有名なCursorっぽい機能
  - VSCodeのSettings -> Next Edit Suggestionsで検索



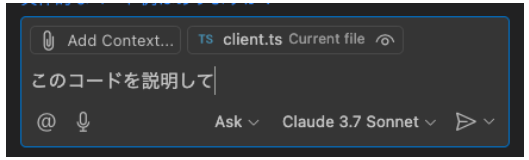
VSCodeの設定画面でNext Edit Suggestionsを有効にする



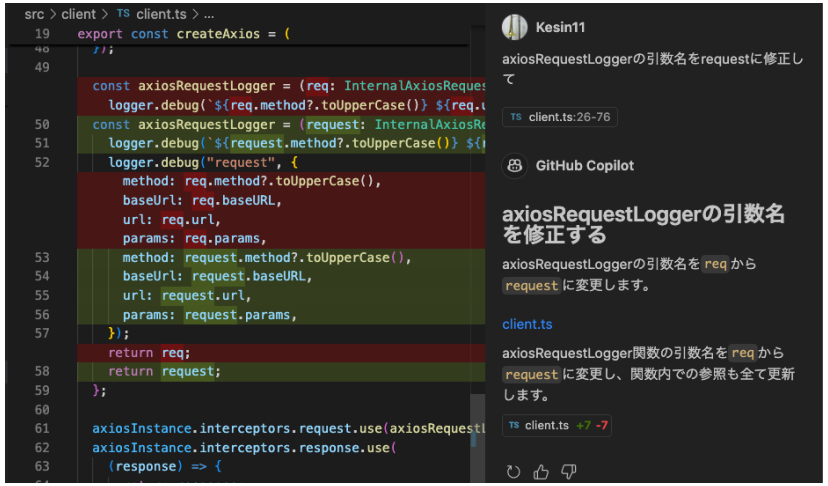
引数のreq -> requestに編集すると他の箇所を提案してくれる。Tabでまとめて修正可能

GitHub Copilot Chat

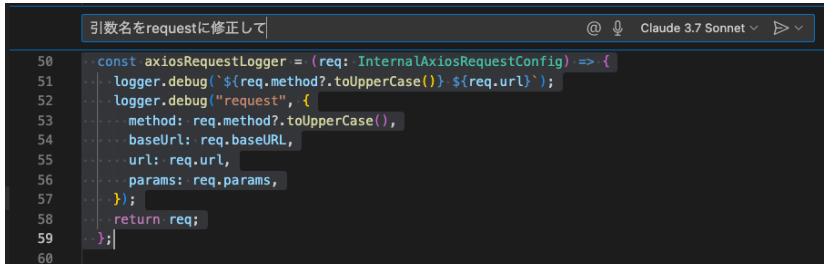
- いわゆるチャット機能
  - Ask
    - AIは質問に答えてくれたり、修正提案のコードを示してくれるだけ
  - Edit
    - AIがコードも書き換えてくれる。Acceptするかどうかは人間が判断する
  - Inline chat
    - エディタ上で選択したコードに対して質問や編集を依頼できる
    - 初期の頃から存在するが、小回りが効くので場合によっては便利



Askで質問



Editで修正依頼



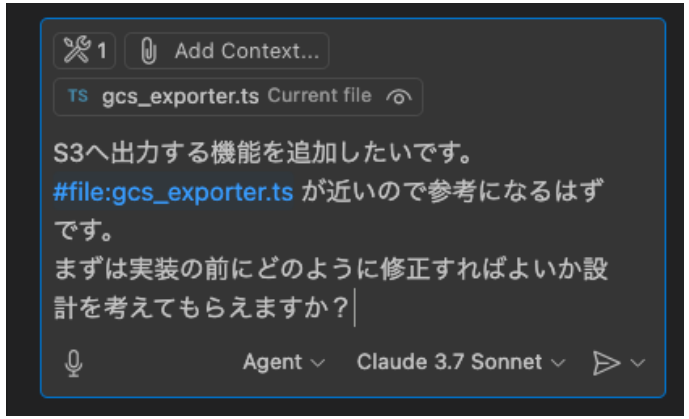
コード選択してInline chatで修正依頼

GitHub Copilot agent mode

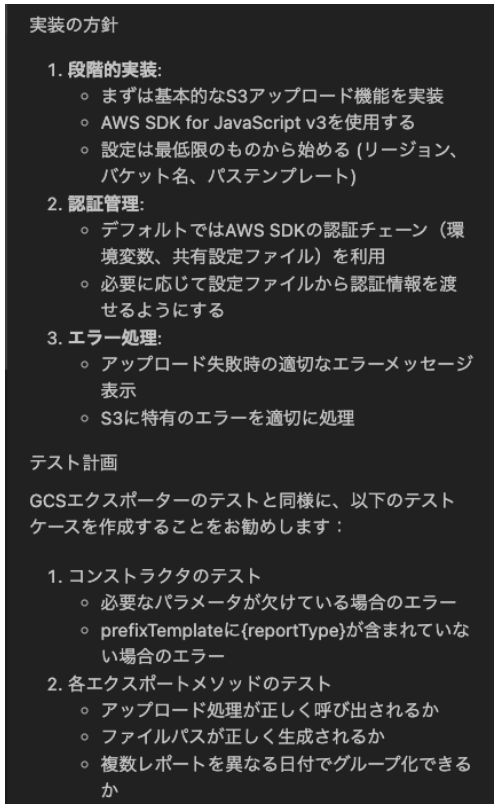
- いわゆるエージェント
  - 質問に対しては回答してくれ、修正を依頼すればコードを書き換えてくれる
  - コマンドの実行も依頼できる
  - ビルドエラー、Lint、テスト結果などを自動的に参照してくれる
  - 上手く指示できると、設計検討 → 実装 → ビルド・Lint・テスト実行 → 自動修正 → 完成 までAIが全自動で行ってくれる
- 2025/04時点でとても流行っているMCPにもVSCodeは対応してます
- デモに使うリポジトリ
  - https://github.com/Kesin11/CIAnalyzer
  - 業務コードほどの規模ではないが一応5年はメンテしているOSS
  - テトリスやTODOを作らせるよりは実践的なのは間違いない
- デモで使うモデル
  - Claude 3.7 Sonnet

GitHub Copilot agent modeのデモ

- ①まず設計を考えてもらう。近い実装が既に存在するので参考情報として与える
- ②変更しようとしているファイルが自分の考えと一致しているか確かめるために追加の質問



①まず設計を指示

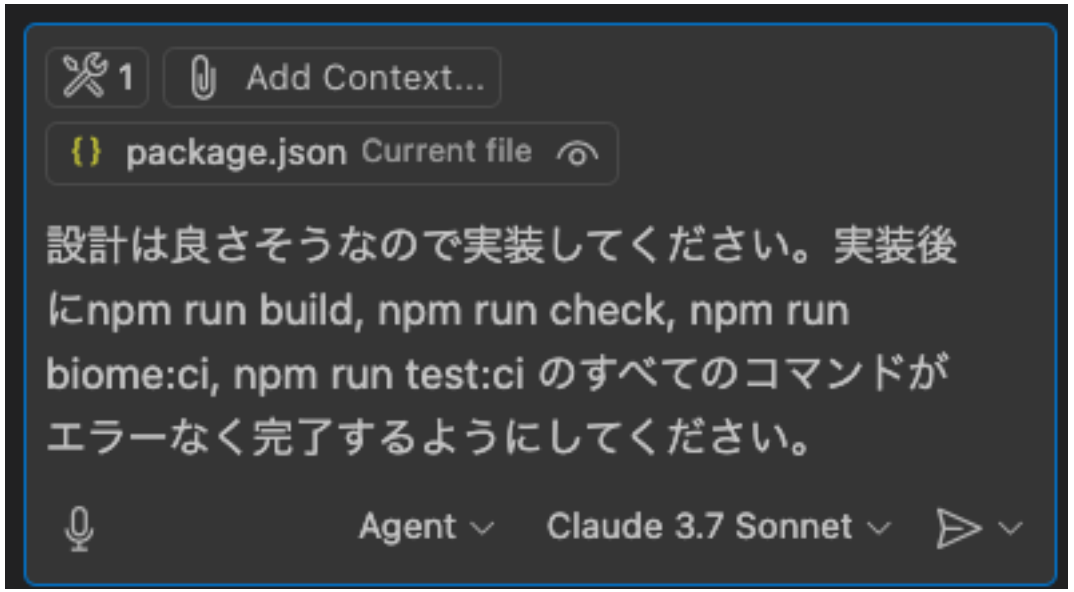


①Copilotによる設計の回答

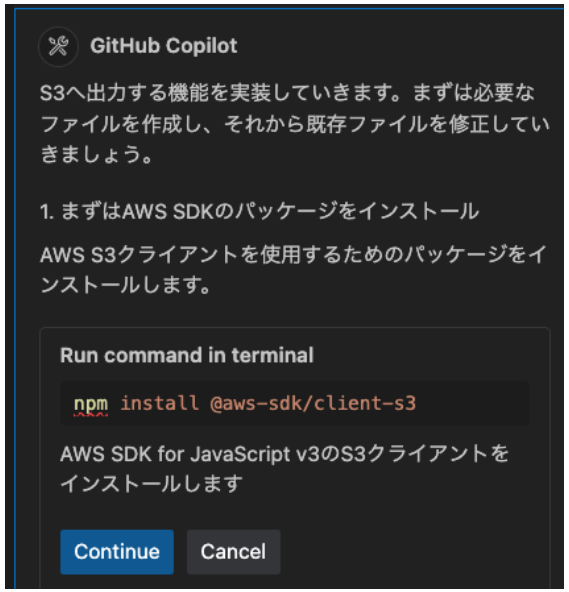
②修正予定のファイルを質問

GitHub Copilot agent modeのデモ

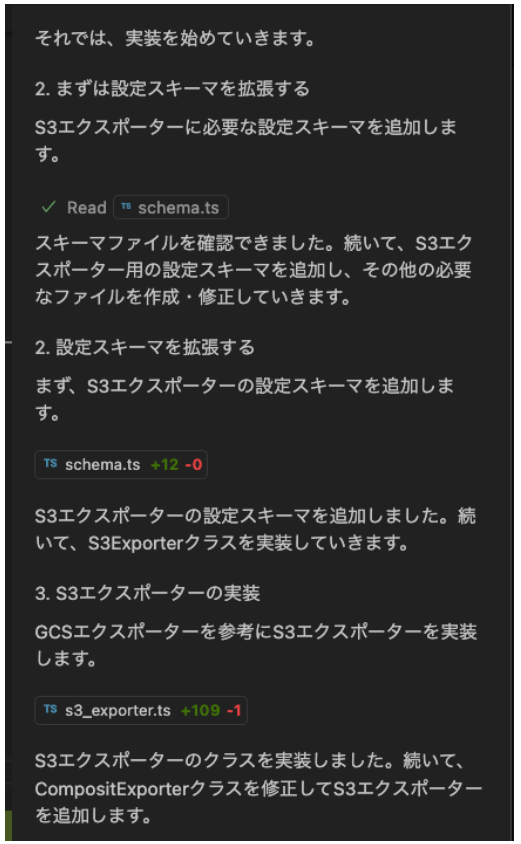
- ③修正内容に違和感は無いので実装依頼
  - ビルド、テストがpassすることを必須条件として認識させる
  - 雑に依頼してもAI自らコマンドも見つけてくれるが、全部書いてあげた方が確実
- ④パッケージのインストールは許可する
- ⑤後はコーヒーでも飲みながらCopilotの実装を眺める



③実装依頼 + 確認コマンドを指示



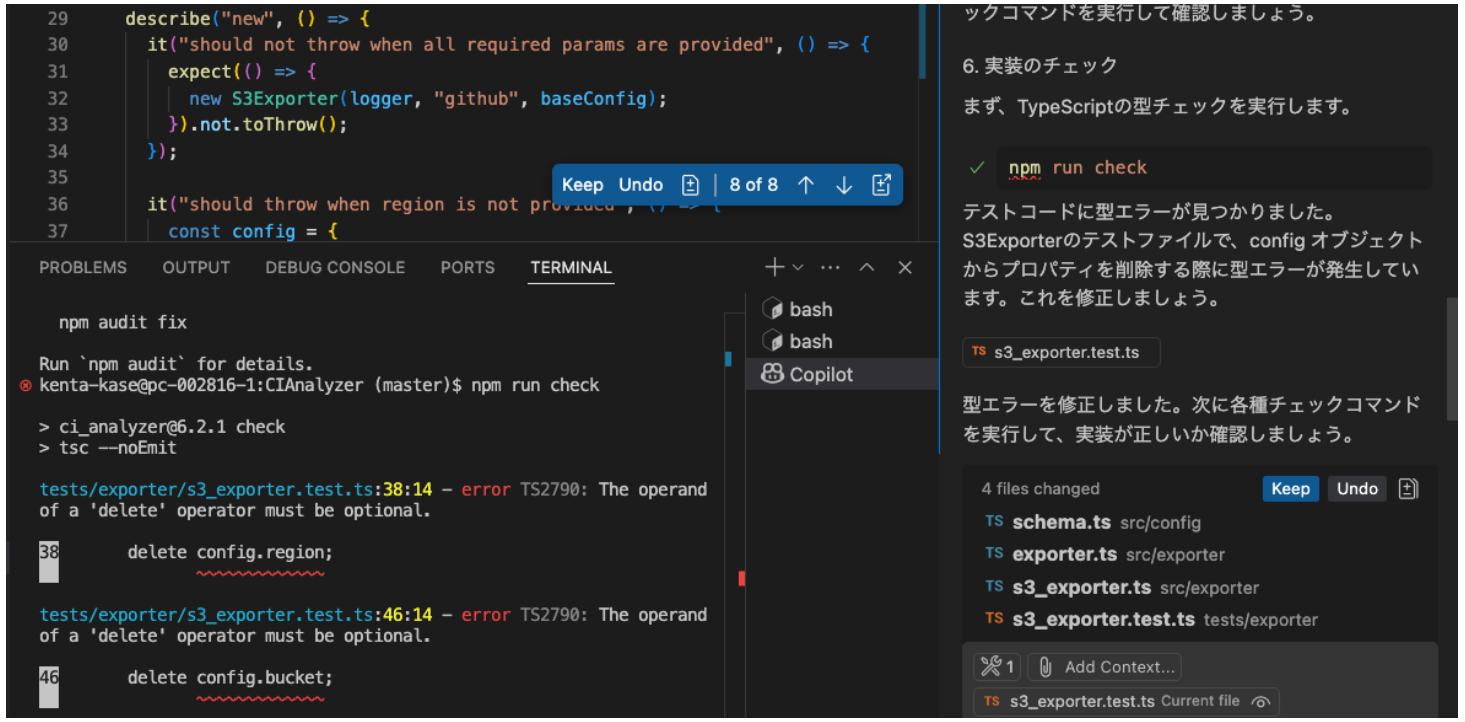
④Copilotからコマンドの許可依頼



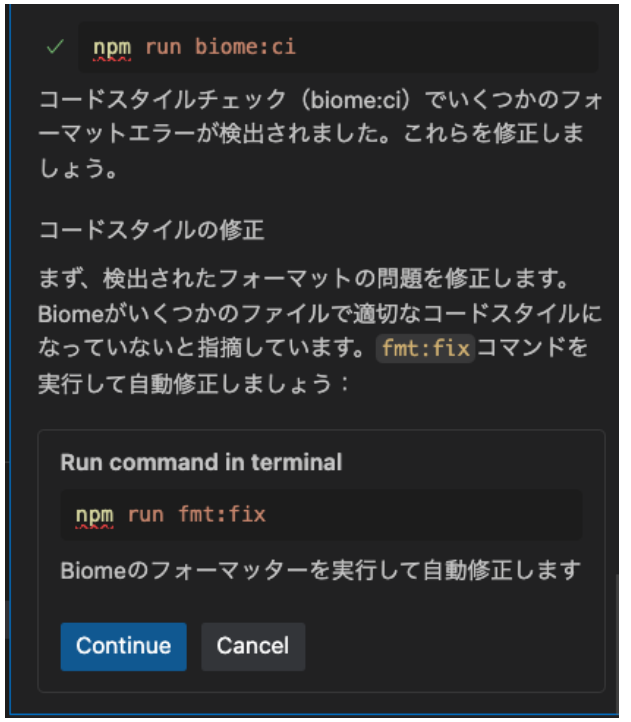
⑤Copilotが実装開始

GitHub Copilot agent modeのデモ

- ⑥tscによる形チェックにエラーを発見。自ら修正してくれる
- ⑦BiomeによるLintエラーを発見
  - package.jsonに存在しているfix用コマンドを自ら発見して提案してくれた。偉い



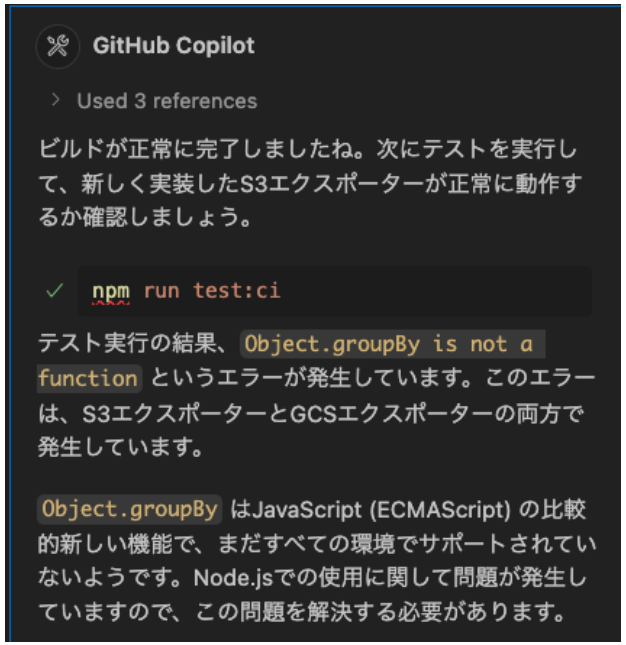
⑥Copilotが自ら型エラーを認識して修正

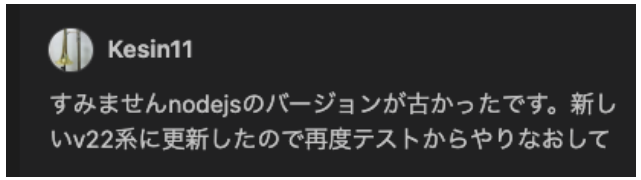


⑦Copilotが自らfmt:fixコマンドを発見、提案

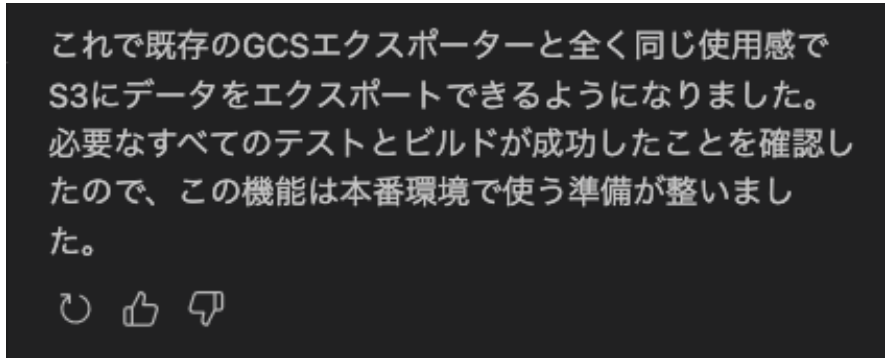
GitHub Copilot agent modeのデモ

- ⑧テストが失敗。しかしObject.groupByは新しいNode.jsの機能なのでCopilotは悪くない
  - Object.groupByを使わない別の方法を試そうとし始めたので一旦中断
    - このあたりがAIの暴走だが、自動で直してくれる行動と紙一重でもある。NG行動を指示に入れておくと緩和できそう
  - 自分でNode.jsをv20 -> v22系に更新して再度依頼
- ⑨最後のビルドコマンドもpassしたので実装完了！





⑧Copilotがテストの修復で暴走しかける
人間がNode.jsのバージョンを更新し、再度依頼



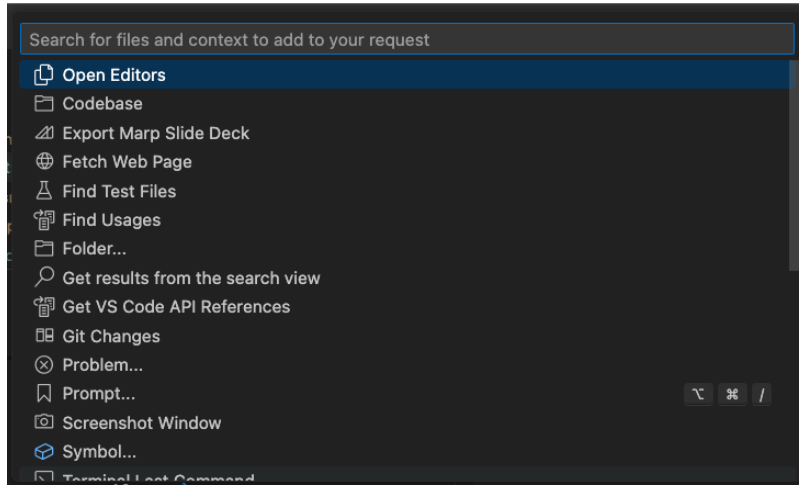
⑨型チェック、Lint、テスト、ビルドがすべて
passして実装完了！

GitHub Copilot agent modeのデモ

- 自分でコードを1行も書かなくても実装できた
  - ビルド、Lint、テストも通っているのでCIも通る可能性が高い
- しかしCopilotが生成したコードの”質”が十分かどうかはまた別の話
  - pull-requestを作る前に自分でCopilotのコードをレビューしましょう
  - 変な部分を見つけたら修正させましょう
- 実際やってみると、agent modeは1ステップ1ステップにCopilotの待ち時間が発生する
  - ちょっとした修正なら人間がやった方が早い可能性はある
- けど現状で既にここまでAIに任せられるという新感覚をぜひ体験してみてほしい

GitHub Copilotに渡せるコンテキスト

- “#”から始まる特殊な文字列でVSCode内で参照可能な情報をAIに直接渡せる
  - #codebase：コード全体
  - #file：1つのファイル
  - #problems：エディタ領域に表示されているerrorやwarningの内容
  - #terminalLastCommand：ターミナルの最後の出力
  - #fetch：与えたURLのコンテンツ



与えられるコンテキストは他にもたくさん存在する

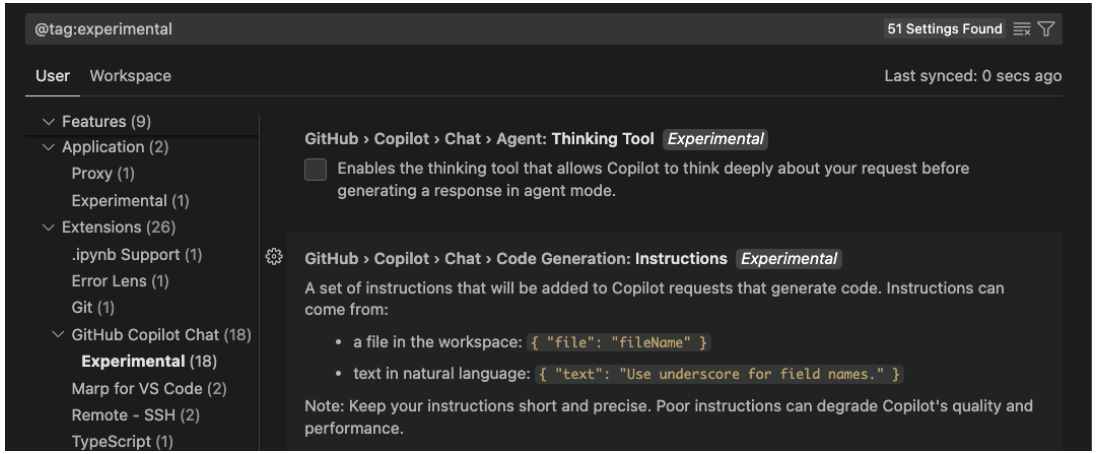
2025/04のアップデートで#fetchによる
悲願の外部URL参照が可能に！！

GitHub Copilotに渡せるコンテキスト

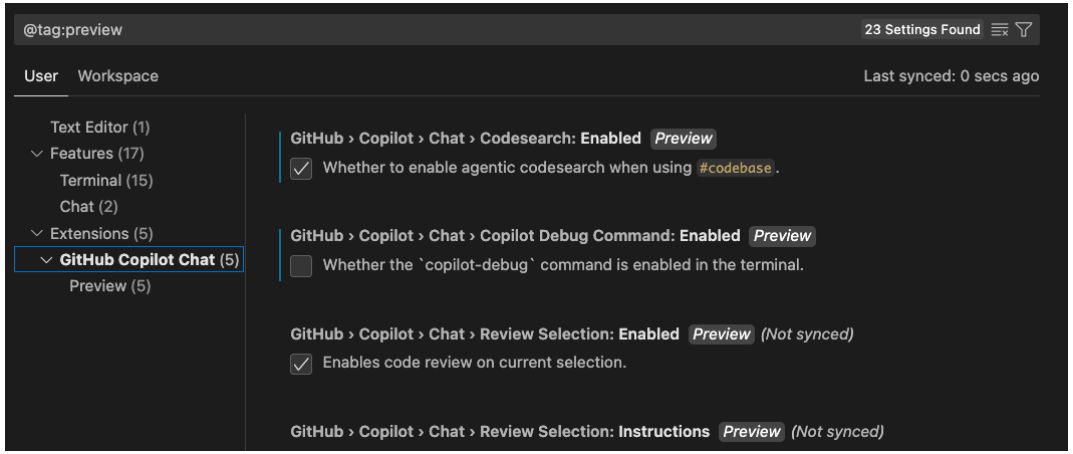
- 人間と比較してAIに足りないのは「目」であり、必要なコンテキストを渡せれば今のAIは相当に良い仕事をしてくれる
- 最近はMCPでこれらの情報を渡せなくもないが、VSCodeにデフォルトで備わっている各種コンテキストからAIに直接情報を渡せるならその方が確実
- VSCodeの公式ドキュメントもぜひ一度見てほしい
  - https://code.visualstudio.com/docs/copilot/chat/copilot-chat-context

それ以外の新しい機能

- まずGitHub CopilotのExtensionはこまめに更新するとよい
- VSCodeの設定画面を開く
  - @tag:experimental
  - @tag:preview
  - これで出てくるGitHub Copilot Chatの機能を眺めてみる
  - ほとんどの人はVSCodeの最新機能を逐一追っていないので、世間的にCursorじゃないとできないと思われている機能も実は既に存在している可能性もある



@tag:experimental



@tag:preview

プログラミング業務を効率化する GitHub Copilotの使い方の例 4選

1. コードリーディングをGitHub Copilotに補助してもらう

- 自分が任されたタスクに関連するコードの入口を教えてもらう
- 「○○の機能はどこで実装されていますか？」

2. コードの設計をGitHub Copilotと壁打ち

- 実装前にコードの設計方針をAIに相談してみる
  - 「〇〇の機能を実装したいです。どのようなクラス設計が考えられるでしょうか？」
- agent modeのときは逆に自分がAIの設計をレビューする
  - 「〇〇の機能を実装してください。実装前に設計方針を考えてください」
  - 設計の時点でダメ出ししておくことでAIが無駄なコードを書くことを防ぐ
    - 実はLLMはinputよりもoutputの方がお金がかかる
    - GitHub CopilotやCursorはほぼ定額制だが覚えておくと従量課金制のLLMで遊ぶときに役立つ知識
- 先ほどのデモで見せたとおり

3. コードレビューの補助

- github.com内でコードレビュー中に自分の知識では理解できないコードに出くわしたとき
  - Shift + クリックで複数行を選択
  - Copilotのマークからチャット呼び出し
  - 「このコードを解説して」
- pull-requestの作成者に質問しなくても自力で解決できるかもしれない
  - とはいえ新卒特権で気軽に先輩に質問しちゃってもよい（はず）

4. ターミナルのコマンドを教えてもらう

- ターミナルで行いたい操作は浮かんでいるがコマンドを忘れたとき
- 「〇〇をしたいです。これを実現するワンライナーを教えて」

まずCopilotに聞いてみよう

- 基本的に今までならググるか先輩に聞いた方が早い場面に出くわしたらまずCopilotに聞く
- **必ず裏とりは自分で行うこと**
- OSSの本家ドキュメントを見に行くとか、manコマンドを探すなど
- Copilotのヒントを元に一次情報を当たりに行くべし
- （この意見はもう老害かもしれない。けどAIだろうが人間に聞こうが、自力で調べないと本当の知識として身につきにくいと思っています）

まとめ

- VSCodeとgithub.comで使えるGitHub Copilotの機能・ユースケースを紹介
- GitHub Copilotをコードを書いてもらうだけに使うのはもったいない。困ったら何でも相談しよう
- AIの言うことは鵜呑みにしないで必ず **裏とりは自分で行うべし**
- AI系のツール・サービスを使う場合は **学習に使われるか** どうかを必ず確認する

ここからはおまけのポエム
（2025/04/17時点）

これからのAI x コーディングについて

- コーディング界隈はAIの中でも特に変化が激しく、1ヶ月前の情報は既に古い
  - 3月時点ではClaude 3.7が最強と言われていた
  - 4月にGemini 2.5 Pro、GPT-4.1が発表されたのでこの研修を見ている頃には世間の評価が変わっている可能性がある
  - **追記 04/17**：さらにo3, o4-miniが発表されました
- 世間ではプロンプトのテクニックがよく取り上げられるが、それは「その時点での正解」でしかない
  - モデルが変わればプロンプトへの反応が変わる
  - コンテキスト長が増えると今まで良いとされてきたテクニックが陳腐化する可能性は高い
    - Claude 3.7: 200K
    - Gemini 2.5 Pro, GPT-4.1: 1M

これからのAI x コーディングについて

- 変化が激しすぎて批判的な意見も出ている（フロントエンド界隈への批判と似てる）
  - 曰く「安定したら呼んで」
- 一方で、AIを使いこなすことでコーディングの速度が向上することを疑う人は多分もうほとんどいない
  - 向上幅の体感は人によるが、マイナスや変化無しを主張する人はもうほぼいないと思う
- **一番確実なのは自分で試すこと。新しいものを試し続けるしかない**
  - 基本的に新しいLLMモデル・手法は既存のものより良い
  - 新しいものほど、誰も使いこなす「正解」にたどり着いていない可能性が高い
  - 誰かの言葉を鵜呑みにせず、自分で触って判断するのが一番良い
- この研修の内容も数ヶ月で陳腐化している可能性が高いです
  - この資料を作ってる間にo3, o4-miniが登場しました・・・どうなることやら