

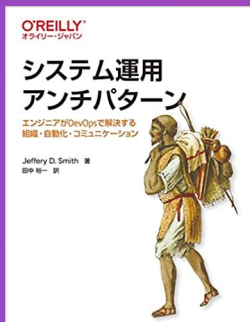
GitHub Copilotで できること一挙ご紹介



田中 裕一

プリンシパルソリューションズエンジニア
GitHub

@yuichielectric



本日のトピック

GitHub Copilotに搭載されている様々な機能、使い方について
「コード補完」、「チャット」、「エディタ機能への統合」という
3つのトピックに沿ってご紹介します。

※ Visual Studio Code 1.87（2024年2月版）を前提としています。





コード補完

いかに適切な文脈を渡すか



GitHub Copilot



LLM



文脈

提案



Visual Studio



Neovim



VS Code



JetBrains IDEs

runtime.go

course.rb

time.js

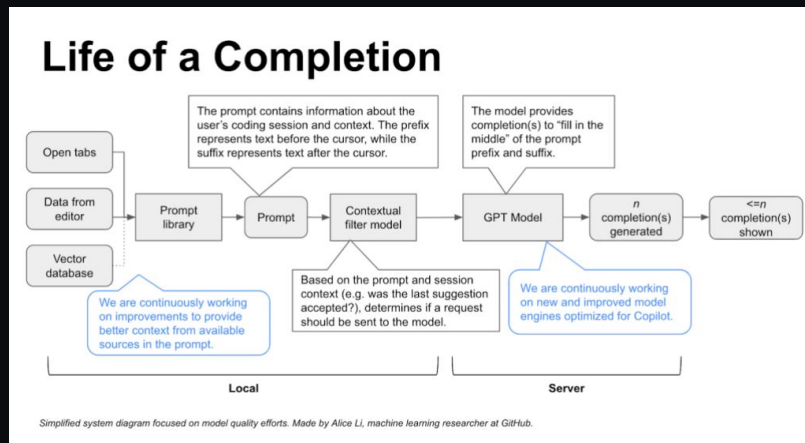
IsPrimeTest.java

```
1 package main
2
3 type Run struct {
4     Time int // in milliseconds
5     Results string
6     Failed bool
7 }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```



文脈とは？

GitHub Copilot拡張によって、**現在のファイルのカーソルの前後のソースコードや、別のタブで開いているファイルの中から**、関連するコードスニペット（コメント含む）を選択し、優先順位、フィルタリングを経た上でプロンプトに追加される情報のこと。

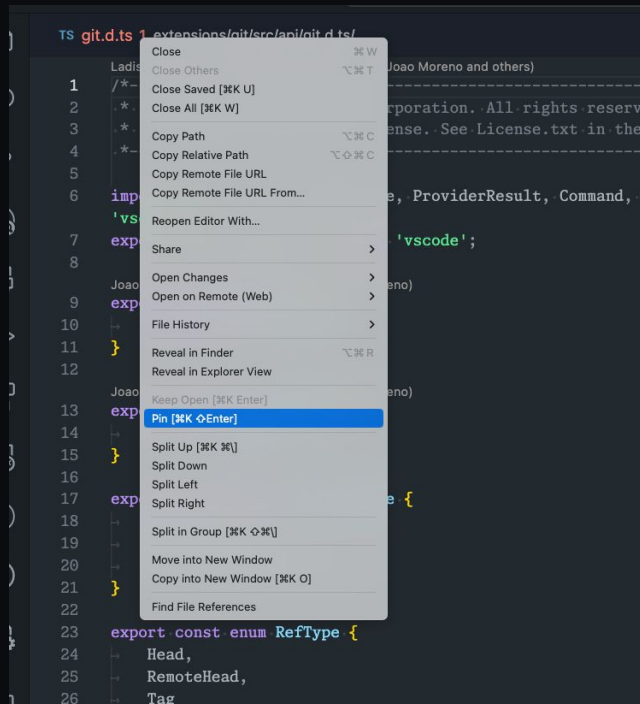


関連するファイルは常に関いておこう

ピンどめしておくとかき忘れを防ぐことができる。

例)

- TypeScriptプロジェクトで使うライブラリの型定義ファイル（.d.tsファイル）
- テストコードを書く時のテスト対象のコード、もしくはその逆



名前は大事

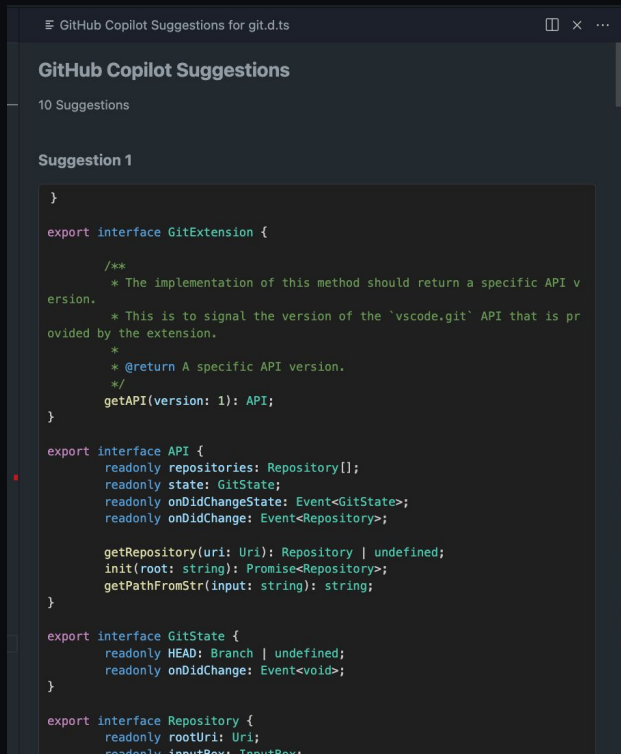
関連するコード箇所を探す上で、変数や関数の名前は大きなヒントとなる。一貫性のある命名規則を用い、同一の概念に対して異なる名前をつけないように気をつけましょう。

後述する名前の提案機能も合わせて活用しましょう。



良い提案を得るために

- 複数の提案がある場合もあるので、それを確認してみる。
 - コマンドパレットで“GitHub Copilot: Open Completions Panel”
- コメントとして、欲しいコードについての説明を自然言語で書く。
 - 必要に応じて例も書く
 - 例) 「引数Xに〇〇という入力を渡したら××という値を返す関数」
- 自分でコードを書き始めてみる。
 - 書き始めたコードもCopilotにとっての文脈となる。



```
export interface GitExtension {  
  
    /**  
     * The implementation of this method should return a specific API version.  
     * This is to signal the version of the 'vscode.git' API that is provided by the extension.  
     */  
    @return A specific API version.  
    getAPI(version: 1): API;  
}  
  
export interface API {  
    readonly repositories: Repository[];  
    readonly state: GitState;  
    readonly onDidChangeState: Event<GitState>;  
    readonly onDidChange: Event<Repository>;  
  
    getRepository(uri: Uri): Repository | undefined;  
    init(root: string): Promise<Repository>;  
    getPathFromStr(input: string): string;  
}  
  
export interface GitState {  
    readonly HEAD: Branch | undefined;  
    readonly onDidChange: Event<void>;  
}  
  
export interface Repository {  
    readonly rootUri: Uri;  
    readonly initBox: InputBox;
```



チャット

スラッシュコマンド、エージェント、コンテキスト変数
を使いこなす



GitHub Copilot Chat

エディタ上のコンテキストを踏まえた、Copilotとのチャット

✓ コーディングに関する質問への回答

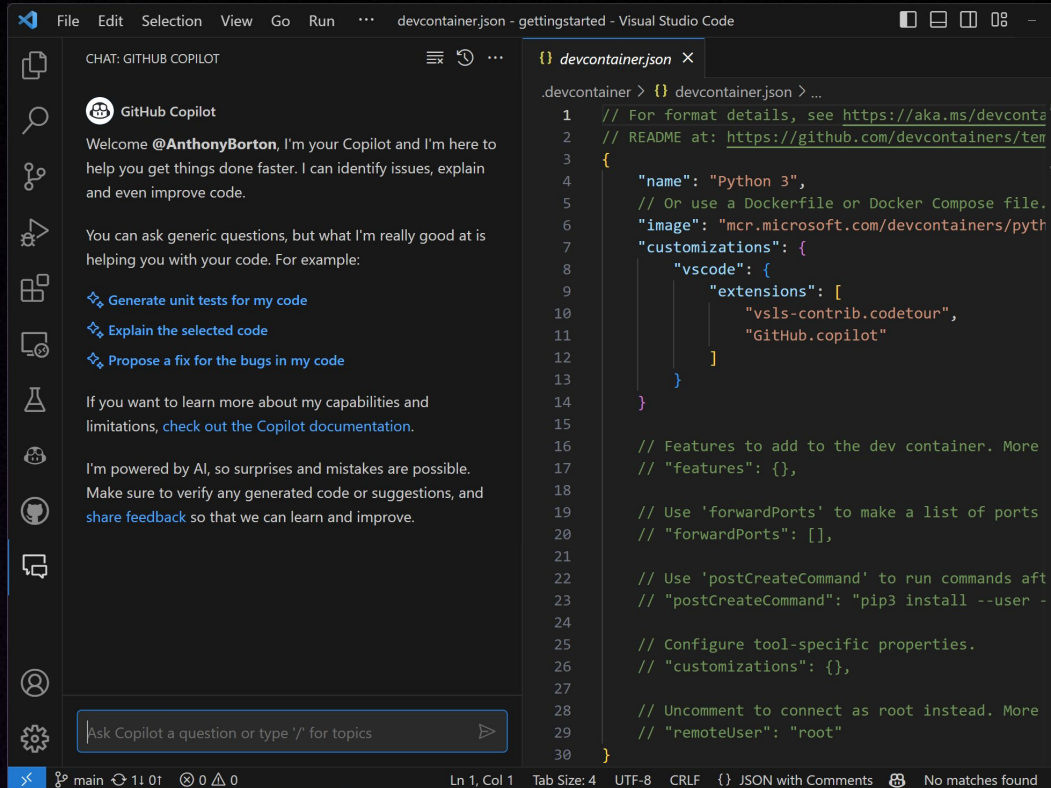
✓ コードの説明

✓ ユニットテストの生成

✓ コードの修正提案

✓ 可読性の向上

✓ 言語変換



yuichielectric

ここのcommitMessagePath変数には具体的にどういった値が渡されますか？



GitHub Copilot

Used 1 reference >

`commitMessagePath` は、Gitのコミットメッセージを含むファイルのパスを表す文字列です。このパスは、Gitリポジトリのルートディレクトリからの相対パスまたは絶対パスである可能性があります。

このメソッドが呼び出されると、`commitMessagePath` には、ユーザーが編集するためのコミットメッセージの一時ファイルのパスが渡されます。この一時ファイルは、`git commit` コマンドが呼び出されたときに作成され、ユーザーがコミットメッセージを入力するためのテンプレートとなります。

したがって、`commitMessagePath` の具体的な値は、現在のGitリポジトリと操作のコンテキストによります。例えば、`/path/to/repo/.git/COMMIT_EDITMSG` のような値が含まれる可能性があります。

```
34
35  async handle({ commitMessagePath }: GitEditorRequest): Promise<any> {
36      if (commitMessagePath) {
37          const uri = Uri.file(commitMessagePath);
38          const doc = await workspace.openTextDocument(uri);
39          await window.showTextDocument(doc, { preview: false });
40
41          return new Promise((c) => {
42              const onClose = window.tabGroups.onDidChangeTabs(async (tabs) => {
43                  if (tabs.closed.some(t => t.input instanceof TabInputText && t
44                      )) {
45                      onClose.dispose();
46                      return c(true);
47                  }
48              });
49          });
50      }
51
52      getEnv(): { [key: string]: string } {
53          const config = workspace.getConfiguration('git');
54          return config.get<boolean>('useEditorAsCommitInput') ? this.env : {};
55      }
56
57      getTerminalEnv(): { [key: string]: string } {
58          const config = workspace.getConfiguration('git');
59          return config.get<boolean>('useEditorAsCommitInput') && config.get<boolean>
```

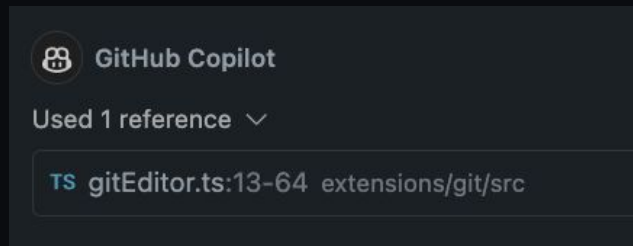


何が文脈として使われるか？

[前提] 基本的には開いているファイルが文脈の候補。

ただし例外もあり（後述）。

1. エディタ上でテキストを選択している場合
→ 選択している箇所のテキストが使われる
2. 何も選択していない場合
→ アクティブなエディタで表示されている範囲のテキストが使われる
3. コンテキスト変数で指定する（後述）



言語設定

`github.copilot.chat.localeOverride`

チャットの返答や
コミットメッセージの自動生成
(後述) の言語を指定可能



Github > Copilot > Chat: Locale Override

Specify a locale that Copilot should respond in, e.g. `en` or `fr`. By default, language locale.

ja



auto

default

en

fr

it

de

es

ru

zh-CN

zh-TW

ja

ko

cs

pt-br

tr

pl

日本語

l be opened.

Github > Copilot > Chat: Welcome Message

コンテキスト変数

#で始まり、プロンプトの中に含めることで文脈を指定するために使われるもの。

右の例の

#file:BookDatabaseImpl.java
の部分がコンテキスト変数。

この例だと、指定されたファイル全体が文脈の対象となる。

 yuichielectric

このファイルの中から脆弱性のあるコードを探してください。 #file:BookDatabaseImpl.java

 GitHub Copilot

Used 1 reference >

この `BookDatabaseImpl.java` ファイルには、SQLインジェクションの脆弱性が存在します。具体的には、`getBooksByTitle` メソッドでユーザーからの入力を直接SQLクエリに組み込んでいます。

```
public List<Book> getBooksByTitle(String name) throws BookServiceException {  
    /*...*/  
    String query = "SELECT * FROM books WHERE title LIKE '%" + name + "%'";  
    /*...*/  
}
```

このコードは、ユーザーが `name` としてSQLコマンドを入力すると、そのコマンドがデータベースで実行される可能性があります。これを修正するには、プリペアドステートメントを使用してユーザー入力を安全にエスケープする必要があります。

修正後のコードは次のようになります。

```
public List<Book> getBooksByTitle(String name) throws BookServiceException {  
    /*...*/  
    PreparedStatement stmt = connection.prepareStatement("SELECT * FROM book  
    stmt.setString(1, "%" + name + "%");  
    ResultSet results = stmt.executeQuery();  
    /*...*/  
}
```

この修正により、ユーザー入力は適切にエスケープされ、SQLインジェクション攻撃から保護されます。

主要なコンテキスト変数一覧

<code>#selection</code>	アクティブなエディタで選択されている箇所
<code>#file</code>	特定のファイル（VS Codeのファイル選択UIが表示される）
<code>#editor</code>	アクティブなエディタで表示されている箇所
<code>#terminalLastCommand</code>	アクティブなターミナルの最後に実行したコマンドの結果
<code>#terminalSelection</code>	アクティブなターミナルの選択されている箇所



コンテキスト変数の利用例

VS Code内のターミナルで、git diffで差分を表示した上で以下のプロンプトを実行

以下の観点でコードをレビューして、修正が必要な箇所を指摘してください。

- コードをさらに簡潔に書くことができるかどうか？
- このコードはコンパイルエラーやランタイムエラーを引き起こさないか？
- セキュリティバグを入れ込んでしまっていないかどうか？

#terminalLastCommand





yuichielectric

以下の観点でコードをレビューして、修正が必要な箇所を指摘してください。

- コードをさらに簡潔に書くことができるかどうか？
- このコードはコンパイルエラーやランタイムエラーを引き起こさないか？
- セキュリティバグを入れ込んでしまっていないかどうか？

#terminalLastCommand



GitHub Copilot

このコードはTypeScriptで書かれており、Gitリポジトリのバージョンを取得するための関数を提供しています。以下に、指摘したいいくつかの点を示します。

1. **簡潔さ**: このコードは既になんかなり簡潔に書かれていますが、一部の冗長な部分をさらに簡潔にすることができます。例えば、`let refs: {[ref:string]:string} = {};` は `let refs = {};` と書くことができます。TypeScriptは動的に型を推論するため、この場合の型注釈は必要ありません。
2. **エラー**: このコードは基本的にはコンパイルエラーやランタイムエラーを引き起こさないように見えます。ただし、ファイルの読み取りに失敗した場合や、期待される形式でないデータが含まれている場合など、エラーが発生する可能性があります。これらのエラーは適切にキャッチされ、`undefined` が返されるようになっています。
3. **セキュリティ**: このコードはファイルシステムにアクセスしますが、そのアクセスはリポジトリのパスに限定されています。そのため、任意のファイルにアクセスされるリ



スラッシュ コマンド

/ (スラッシュ) で始まる
コマンドの事。

プロンプト中に指定することで
Copilotへの依頼を簡潔に記載
することが可能。

右の例では /explain を使って
コードの説明を依頼している。

yuichielectric

@workspace /explain

GitHub Copilot used @workspace /explain

Used 1 reference >

このコードはJavaScriptで書かれた正規表現です。この正規表現は、特定の数値パターンを検出するために使用されます。具体的には、正または負の整数、小数、または指数表記の数値をマッチングします。

詳細に分解すると以下ようになります：

- `^` は文字列の開始を表します。
- `(-)?` はオプションのマイナス記号を表します。つまり、数値が負であることを示すことができます。
- `[0-9\.\+]` は1つ以上の数字または小数点を表します。これにより、整数または小数をマッチングします。
- `(e(-)?[0-9\+])?` はオプションの指数部分を表します。`e` の後にオプションのマイナス記号と1つ以上の数字が続きます。これにより、指数表記の数値（例えば `1.23e-4`）をマッチングします。
- `$` は文字列の終了を表します。

したがって、この正規表現は、例えば `"123"`, `"-123"`, `"123.456"`, `"-123.456"`, `"1.23e4"`, `"1.23e-4"`, `"-1.23e4"`, `"-1.23e-4"` などのパターンにマッチします。

```
16 ... 'subtract': function(a, b) { return a - b },
17 ... 'multiply': function(a, b) { return a * b },
18 ... 'divide': function(a, b) { return a / b },
19 ... };
20
21 ... if (!req.query.operation) {
22 ...   throw new Error("Unspecified operation");
23 ... }
24
25 ... var operation = operations[req.query.operation];
26
27 ... if (!operation) {
28 ...   throw new Error("Invalid operation: " + req.query.operat
29 ... }
30
31 ... if ([!req.query.operand1 ||
32 ...   !req.query.operand1.match(/^(-)?[0-9\.]?(e(-)?[0-9\+])?
33 ...   req.query.operand1.replace(/[-0-9e]/g, '').length > 1)
34 ...   throw new Error("Invalid operand1: " + req.query.operand
35 ... }
36
37 ... if (!req.query.operand2 ||
38 ...   !req.query.operand2.match(/^(-)?[0-9\.]?(e(-)?[0-9\+])?
39 ...   req.query.operand2.replace(/[-0-9e]/g, '').length > 1)
40 ...   throw new Error("Invalid operand2: " + req.query.operand
41 ... }
42
43 ... res.json({ result: operation(req.query.operand1, req.query
44 ... });
45
```

主要なスラッシュコマンド

<code>/explain</code>	選択されたコードを説明
<code>/tests</code>	選択されたコードのユニットテストを生成
<code>/doc</code>	選択されたコードのドキュメンテーションコメントを生成 (後述のインラインチャットのみ)
<code>/fix</code>	選択されたコードに含まれる問題を修正
<code>/new</code>	指定されたワークスペースの雛形を生成



エージェント

@で始まり、プロンプトの中に含めることで、どこから文脈を収集するかを指定する。

右の例の

@workspace

の部分がエージェントの指定。

この例だと、開いていないファイルも文脈の対象となる。

 **yuichielectric** ×

@workspace この電卓アプリに冪乗の演算を追加するためにはどのファイルをどのように修正したら良いですか？

 **GitHub Copilot** used @workspace

Used 7 references >

電卓アプリに冪乗の演算を追加するためには、以下のファイルを修正する必要があります：

- public/index.html** : 新たなボタンを追加して、冪乗の演算を行うためのUIを提供します。例えば、次のようにボタンを追加できます：

```
<button class="btn" onClick="operationPressed('^')">^
```

- public/client.js** : **operationPressed** 関数を修正して、新たに追加した冪乗の演算子処理できるようにします。また、新たな状態を追加して、冪乗の演算を行うためのオペランドを管理します。
- api/controller.js** : ここでは、冪乗の演算を行うロジックを追加します。具体的なコードは、現在のコードベースに依存しますが、一般的には **Math.pow** 関数を使用して冪乗の演算を行います。
- api/routes.js** : 必要に応じて、新たなエンドポイントを追加して、冪乗の演算を行うリクエストを処理します。

以上の修正を行うことで、電卓アプリに冪乗の演算を追加することができます。

主要なエージェント

@workspace	ワークスペース全体についてのチャット
@terminal	ターミナルについてのチャット
@vscode	VS Codeについてのチャット





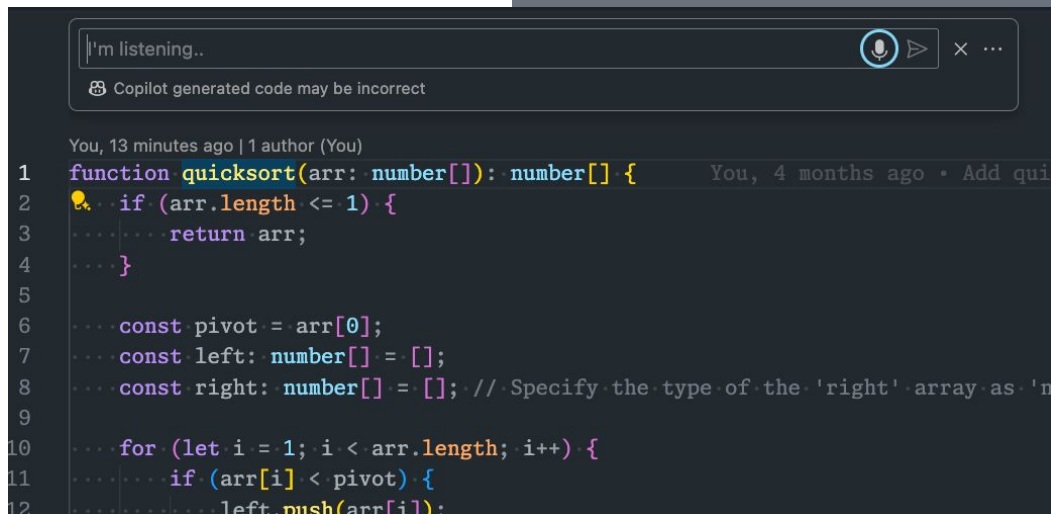
エディタ機能への統合



インライン チャット

チャットペインだけでなく
エディタの中でチャットを
実行することが可能。

ここで提案されたコードは
エディタ上に直接追記される。



```
1 function quicksort(arr: number[]): number[] {  
2     if (arr.length <= 1) {  
3         return arr;  
4     }  
5  
6     const pivot = arr[0];  
7     const left: number[] = [];  
8     const right: number[] = []; // Specify the type of the 'right' array as 'number[]'  
9  
10    for (let i = 1; i < arr.length; i++) {  
11        if (arr[i] < pivot) {  
12            left.push(arr[i]);  
13        } else {  
14            right.push(arr[i]);  
15        }  
16    }  
17    return [...left, pivot, ...right];  
18 }
```

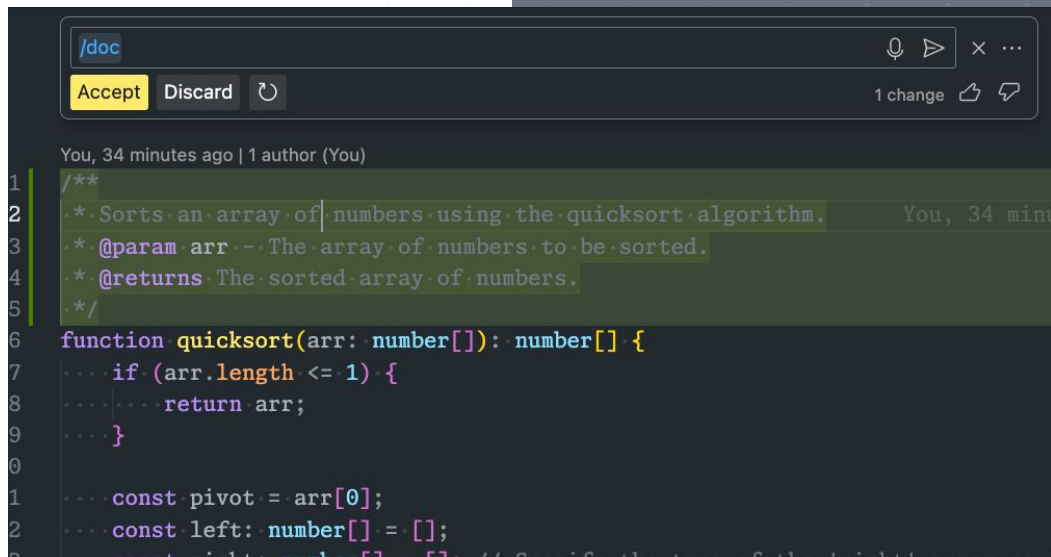

インライン チャット

例) エラー処理のコードを
提案してもらう。



インライン チャット

例) インラインチャットで
/doc を実行してdocstringを
提案してもらう。



The screenshot shows an IDE interface with a dark theme. At the top, there is a search bar containing the text `/doc`. Below the search bar, there are three buttons: `Accept` (highlighted in yellow), `Discard`, and a refresh icon. To the right of these buttons, it says "1 change" followed by two small icons. Below the buttons, there is a text line: "You, 34 minutes ago | 1 author (You)". The main area of the IDE shows a code editor with a JavaScript function `quicksort`. The function signature is `function quicksort(arr: number[]): number[] {`. The function body contains a conditional check `if (arr.length <= 1) { return arr; }` and a pivot selection `const pivot = arr[0];`. The code is highlighted with a green background. The inline chat window is positioned over the code, showing a docstring suggestion for the `quicksort` function. The docstring text is: `/**
 * Sorts an array of numbers using the quicksort algorithm.
 * @param arr - The array of numbers to be sorted.
 * @returns The sorted array of numbers.
 */`. The docstring text is highlighted with a green background.

```
/doc
```

Accept Discard ↺ 1 change

You, 34 minutes ago | 1 author (You)

```
1 /**  
2  * Sorts an array of numbers using the quicksort algorithm.  
3  * @param arr - The array of numbers to be sorted.  
4  * @returns The sorted array of numbers.  
5  */  
6 function quicksort(arr: number[]): number[] {  
7   ... if (arr.length <= 1) {  
8     ... return arr;  
9   }  
10  
11   ... const pivot = arr[0];  
12   ... const left: number[] = [];  
13   ... const right: number[] = [];
```

エラーの 修正提案

```
const pivot = arr[0];
const left = [];
const right = [];

for (let i = 1; i < arr.length; i++) {
  if (arr[i] < pivot) {
    left.push(arr[i]);
  } else {
    right.push(arr[i]);
  }
}
```

Argument of type 'number' is not assignable to parameter of type 'never'. ts(2345)

(parameter) arr: number[]

[View Problem \(⌘F8\)](#) [Quick Fix... \(⌘.\)](#)

You, 4 months ago • Add quicksort algorithm implementation

エディタ上のエラー（赤い波線）
箇所のQuick Fixメニューの中に
Copilotを使うアイテムが表示。

“Fix using Copilot”を選択すると、
修正提案を適用。

```
for (let i = 1; i < arr.length; i++) {
  if (arr[i] < pivot) {
    left.push(arr[i]);
  } else {
    right.push(arr[i]);
  }
}

return [...quicksort(left), pivot, ...quicksort(right)];
} ✨
```

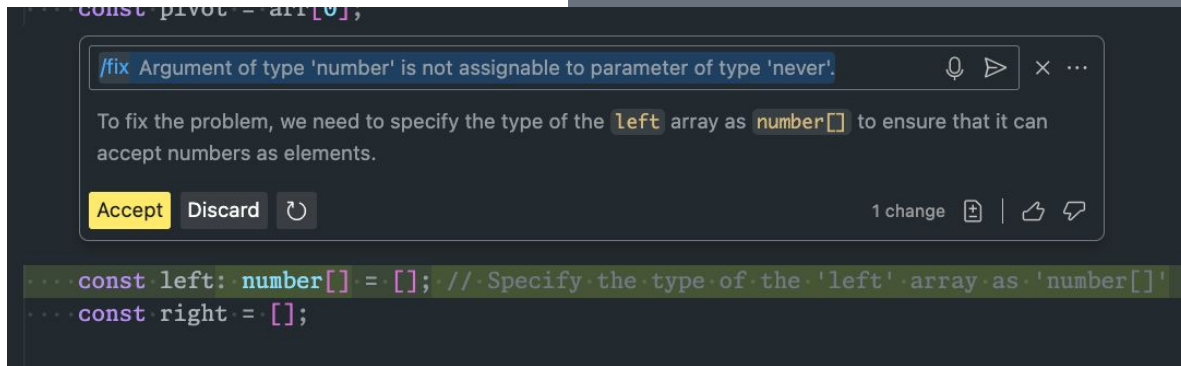
Quick Fix

- Fix using Copilot
- Explain using Copilot

エラーの 修正提案

エディタ上のエラー（赤い波線）
箇所のQuick Fixメニューの中に
Copilotを使うアイテムが表示。

“Fix using Copilot”を選択すると、
修正提案を適用。



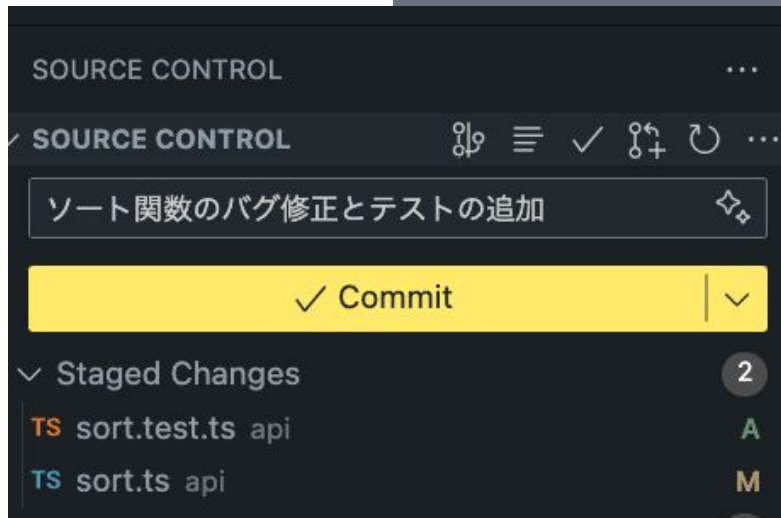
シンボルの rename

コマンドパレットから“Rename Symbol”を実行すると周囲の文脈から適切と思われる名前を提案してくれる。

```
function quicksort(arr: number[]): number[] {
  if (arr.length <= 1) {
    arr
    // inputArray
    // arrayToSort
    // unsortedArray
    Enter to Rename, ⌘Enter to Preview
  }
  for (let i = 1; i < arr.length; i++) {
    if (arr[i] < pivot) {
      left.push(arr[i]);
    }
  }
}
```

コミット メッセージの提 案

現在の変更差分から、それを
説明するコミットメッセージを
提案。



Thank you

