

# GenAI Coding Assistant

# Objectives

---

- Overview of Gen AI Developer Tools
- Features of Github CoPilot
- Understanding the importance of the Co in CoPilot
- Best Practices

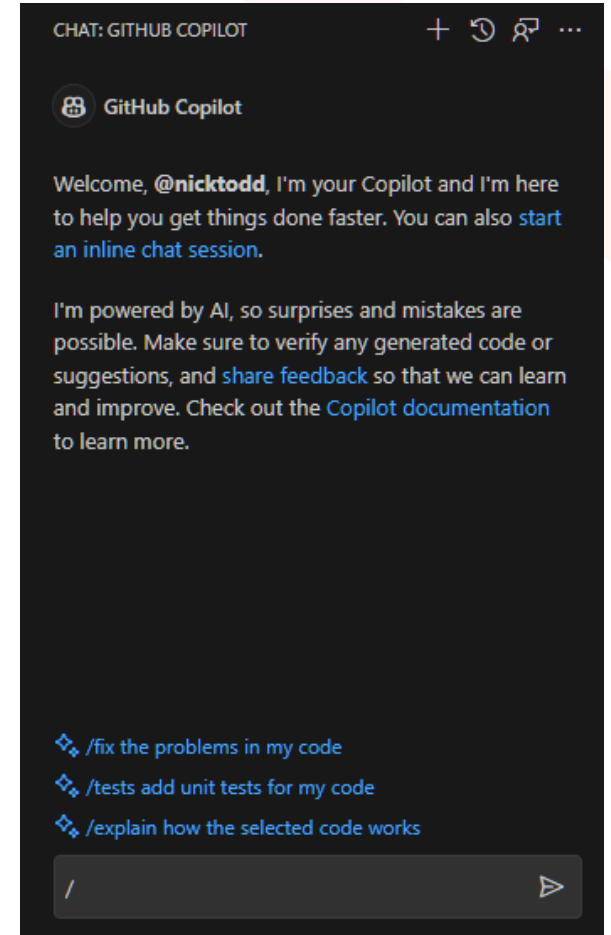
# Overview of GenAI Tools for Developers

---

- Github CoPilot
- SourceGraph Cody
- Amazon Q Developer
- Deepseek R1
- Kimi (no VSCode Plugin yet)

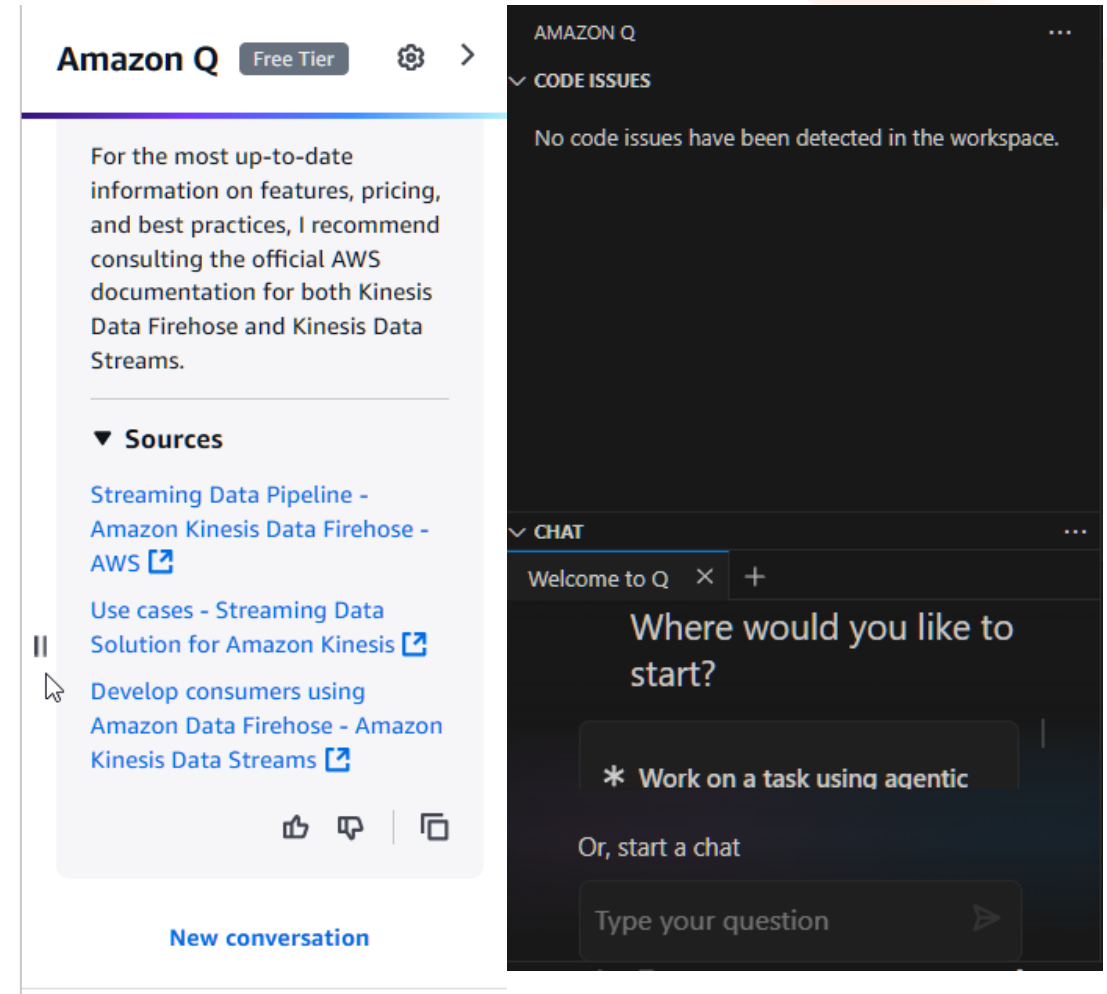
# GitHub CoPilot

- This was the first developer tool to gain significant traction
  - <https://github.com/features/copilot>
- Available in free and paid for versions
- Currently uses
  - Claude 3.5
  - Sonnet
  - GPT-4o
  - as of Dec 2024
- Can be used in multiple IDEs



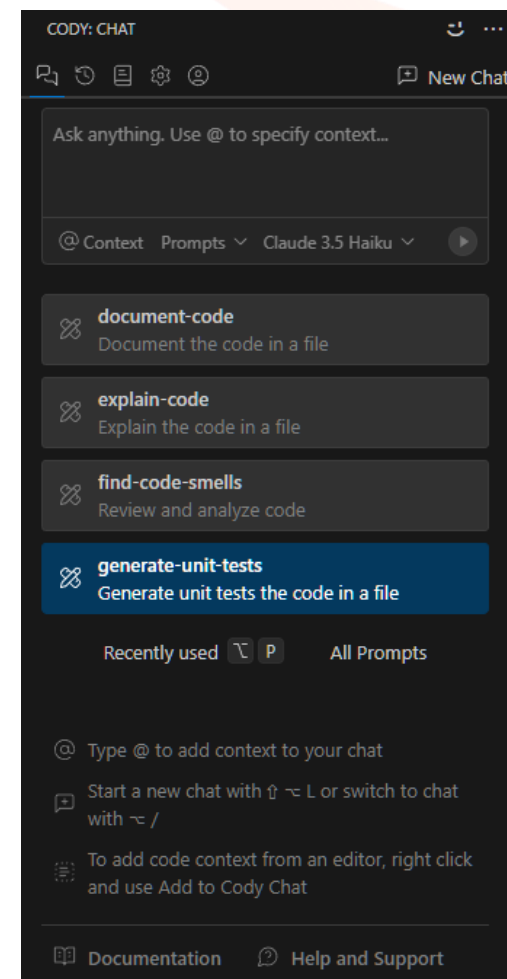
# Amazon Q Developer

- The Amazon Q Developer tool provides Gen AI capabilities
  - <https://aws.amazon.com/q/developer/>
- Amazon Q Developer operates
  - In the IDE (similar to the other tools)
  - In the AWS Management Console
  - On the console
- You will require an AWS Builder ID
  - Free and easy to obtain
- Multiple IDEs are supported



# SourceGraph Cody

- Cody is the tool from SourceGraph
  - <https://sourcegraph.com/cody>
- Has a free tier and then paid options
- Works in multiple IDEs



# Deepseek R1

---

- Deepseek is a Chinese AI model
- Needs account
- Works in multiple IDEs



# GenAI Features

---

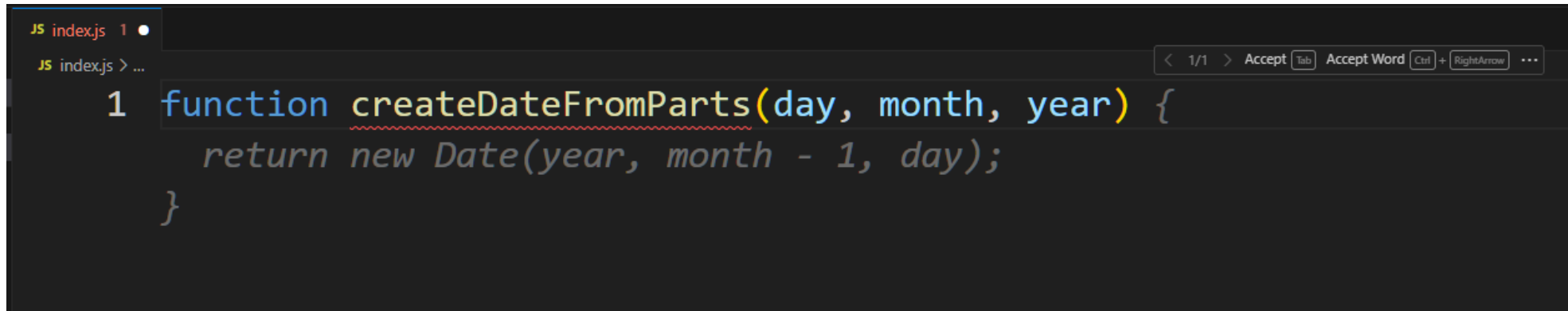
- All GenAI tools have a growing set of features including
  - Autocompletion
  - Requesting suggestions
  - Context Menus
    - Generating documentation
    - Getting explanations
    - Fixing errors
    - Generating tests
  - Chat
  - Project creation



# AutoCompletion

---

- One of the most useful features of GitHub CoPilot tools is the autocompletion capability
  - You begin typing some code and it suggests completions



```
JS index.js 1 •
JS index.js > ...
1 function createDateFromParts(day, month, year) {
    return new Date(year, month - 1, day);
}
```

The screenshot shows a code editor interface. At the top, there's a tab labeled 'JS index.js' with a cursor icon. Below the tab, the code is displayed on a dark background. The first line is '1 function createDateFromParts(day, month, year) {'. The function name 'createDateFromParts' is highlighted with a red squiggly line underneath it, indicating a suggestion. The rest of the function body is 'return new Date(year, month - 1, day);' followed by a closing brace '}'. In the top right corner of the editor, there are several buttons: '< 1/1 > Accept Tab Accept Word Ctrl + RightArrow ...'.

- You can then **Accept** the suggestion using **Tab**

# Exercise: Start using CoPilot

- In your VM open Visual Studio Code / 在您的虚拟机中打开 Visual Studio Code
- Sign in to your GitHub Account in VS Code / 在 VS Code 中登录您的 GitHub 帐户
- Go in the webbrowser to github.com -> Settings -> CoPilot and activate free version of CoPilot / 在网页浏览器中访问 github.com -> 设置 -> CoPilot, 并激活 CoPilot 免费版
- Create an empty file in VSCode and get try out auto-completion / 在 VSCode 中创建一个空文件并试用自动补全功能

## GitHub Copilot

### Your AI Pair Programmer

Trained on billions of lines of code, [GitHub Copilot](#) turns natural language prompts into coding suggestions across dozens of languages.

[Start using Copilot Free](#) or [try Copilot Pro](#)

```
1 import urllib.request, json
2
3 def get_repositories(org):
4     """List all names of GitHub repositories for an org"""
5     url = 'https://api.github.com/orgs/' + org
6     request = urllib.request.Request(url)
7     response = urllib.request.urlopen(request)
8     data = json.loads(response.read().decode())
9     return [repo['name'] for repo in data]
```

## Visibility

### Show Copilot

Enable Copilot for all GitHub features, including navigation bar, search, and dashboard. When disabled, Copilot will be hidden and unavailable. This setting does not apply to Copilot search on GitHub Docs.

Enabled ▾

## Get Copilot from an organization

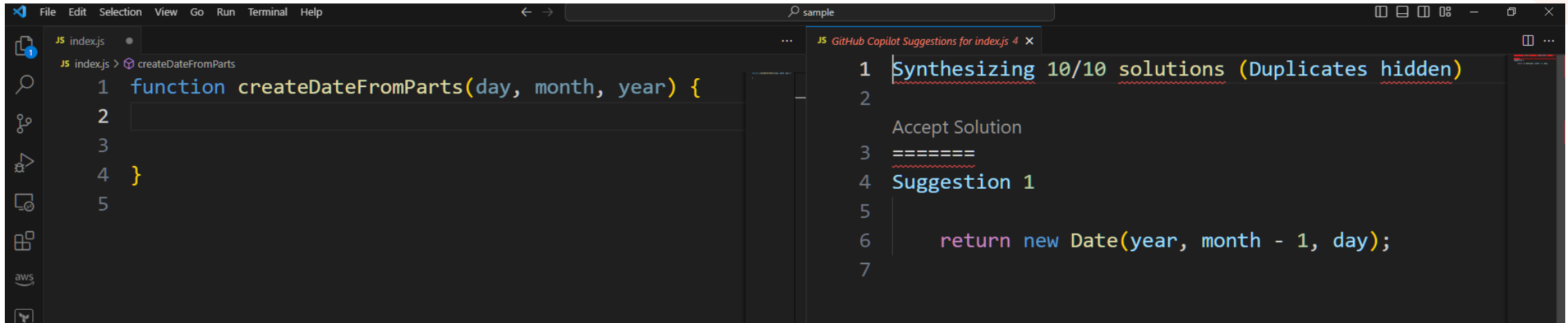
Organizations can provide their members (including you) and their teams access to GitHub Copilot.

You do not belong to any organizations.

[Create an organization](#)

# Request Suggestions

- Using **Ctrl-Enter**, you can request suggestions from GitHub CoPilot



The screenshot shows the Visual Studio Code editor with a file named `index.js`. The code defines a function `createDateFromParts` with parameters `day`, `month`, and `year`. The function body is currently empty. A sidebar on the right, titled "GitHub Copilot Suggestions for index.js 4", displays the following suggestions:

```
1 Synthesizing 10/10 solutions (Duplicates hidden)
2
3 Accept Solution
4 =====
5 Suggestion 1
6     return new Date(year, month - 1, day);
7
```

- Multiple suggestions are then displayed and you can select the preferred suggestion

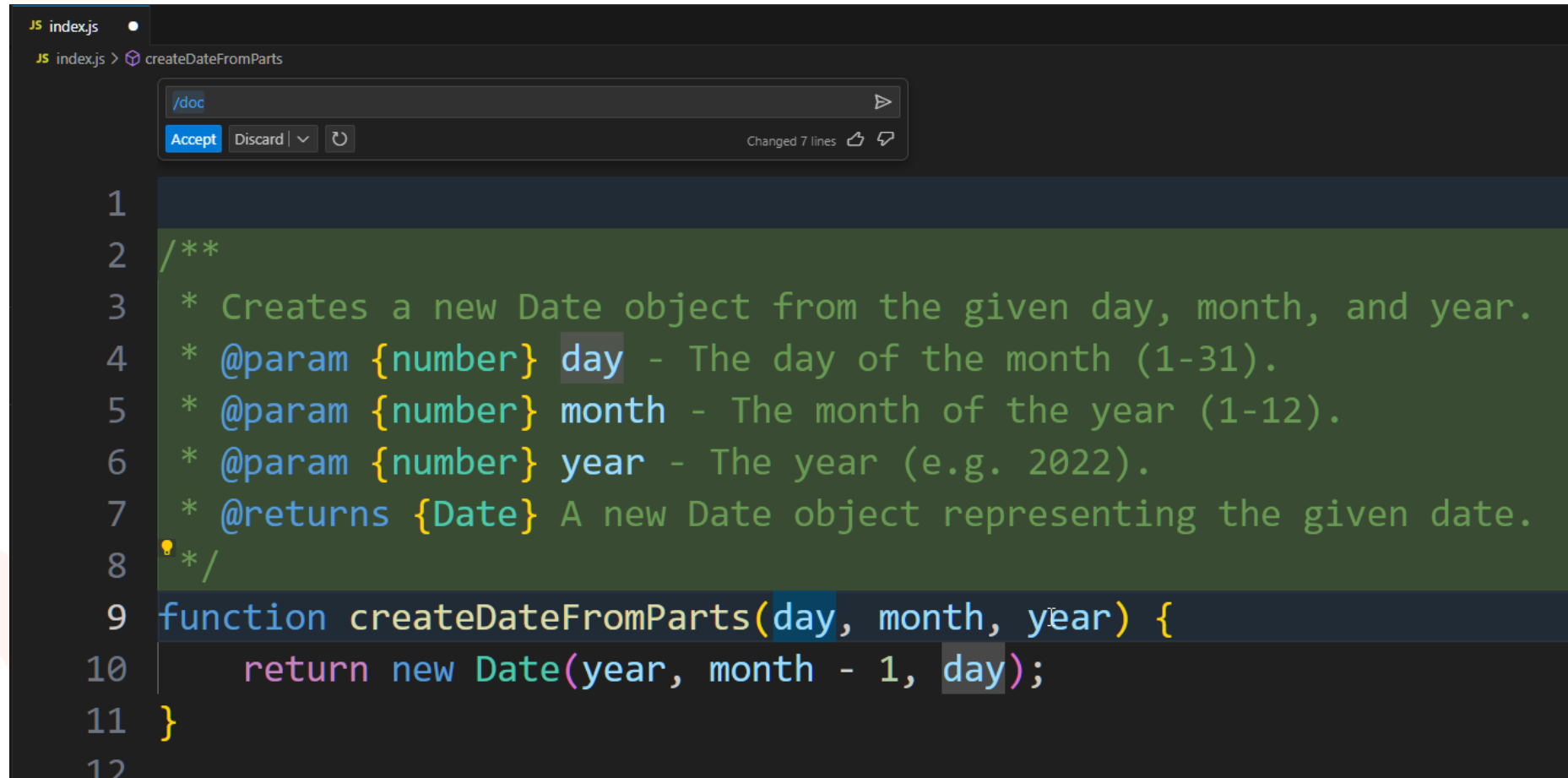
# Context Menu

---

- You can bring up the GitHub CoPilot context menu using **Ctrl-I**
- From this menu you can then enter one of the following commands
  - **/doc**
  - **/explain**
  - **/fix**
  - **/test**

# Using /doc to Generate Documentation

- Using **/doc** you can generate documentation for your code



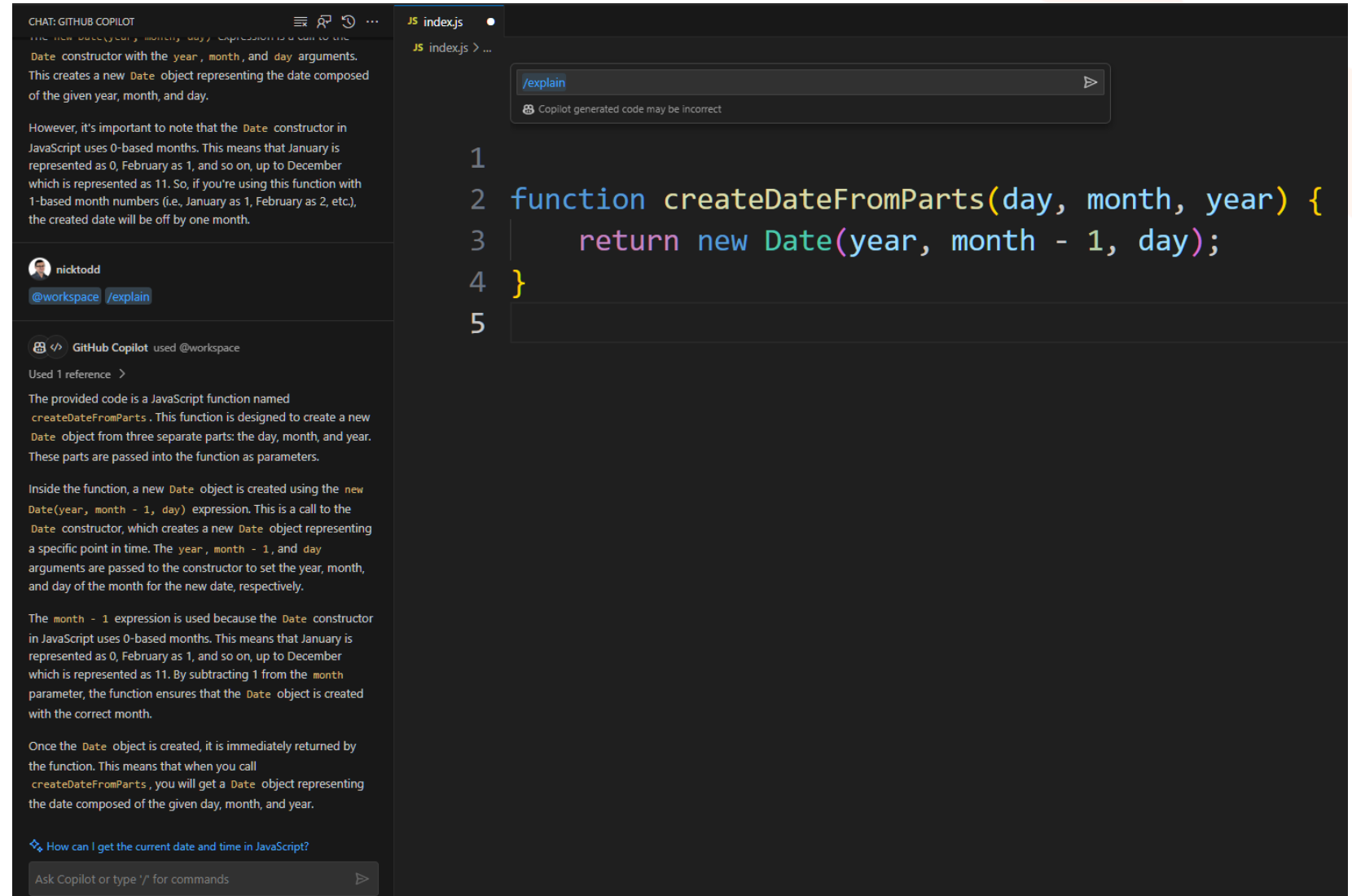
```
JS index.js
JS index.js > createDateFromParts

/doc
Accept Discard Changed 7 lines

1
2 /**
3  * Creates a new Date object from the given day, month, and year.
4  * @param {number} day - The day of the month (1-31).
5  * @param {number} month - The month of the year (1-12).
6  * @param {number} year - The year (e.g. 2022).
7  * @returns {Date} A new Date object representing the given date.
8  */
9 function createDateFromParts(day, month, year) {
10     return new Date(year, month - 1, day);
11 }
12
```

# Using /explain to Understand

- Using **/explain** describes the following code in the GitHub CoPilot Chat window



# Fixing Problems

---

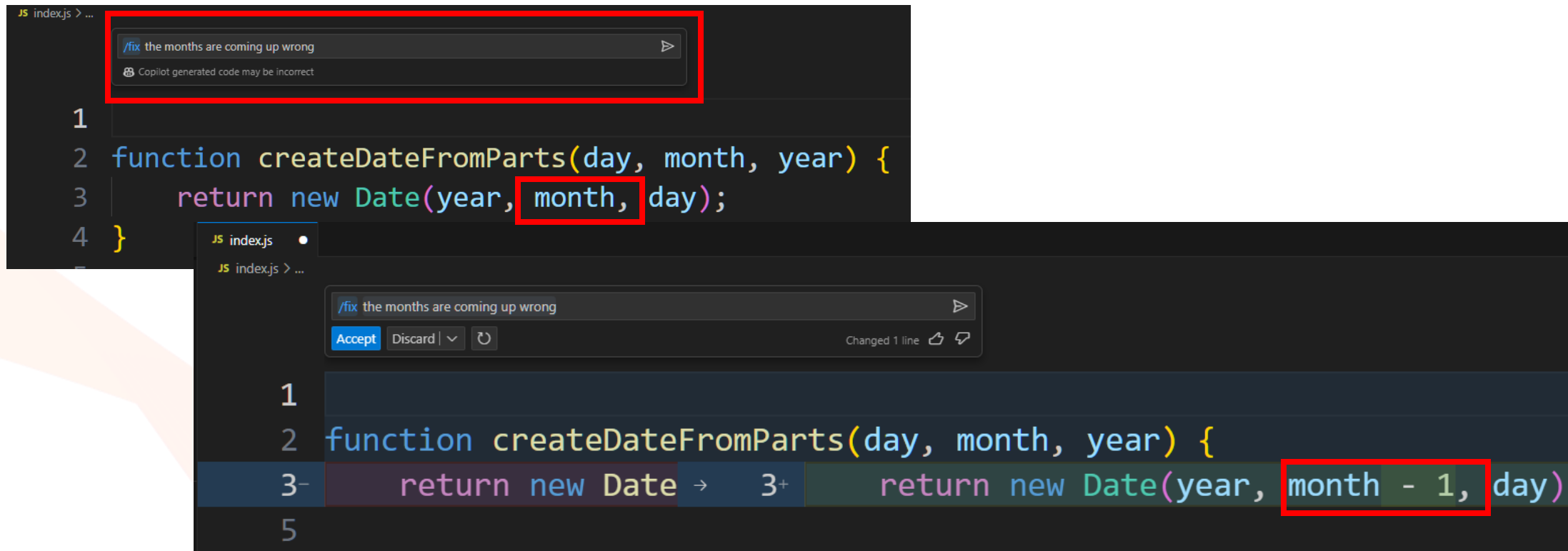
- Let's introduce a deliberate problem with our function

```
function createDateFromParts(day, month, year)
{
    return new Date(year, month-1, day);
}
```

```
function createDateFromParts(day, month, year)
{
    return new Date(year, month, day);
}
```

# Using CoPilot to Fix the Issue

- The code now has the error
- I can ask GitHub CoPilot to fix the error with **/fix** followed by a description of the problem



The screenshot shows a VS Code editor with a JavaScript file named `index.js`. The initial code is:

```
1  
2 function createDateFromParts(day, month, year) {  
3   return new Date(year, month, day);  
4 }
```

A Copilot chat window is open with the prompt `/fix the months are coming up wrong`. Below the prompt, it says "Copilot generated code may be incorrect".

The code is then updated to:

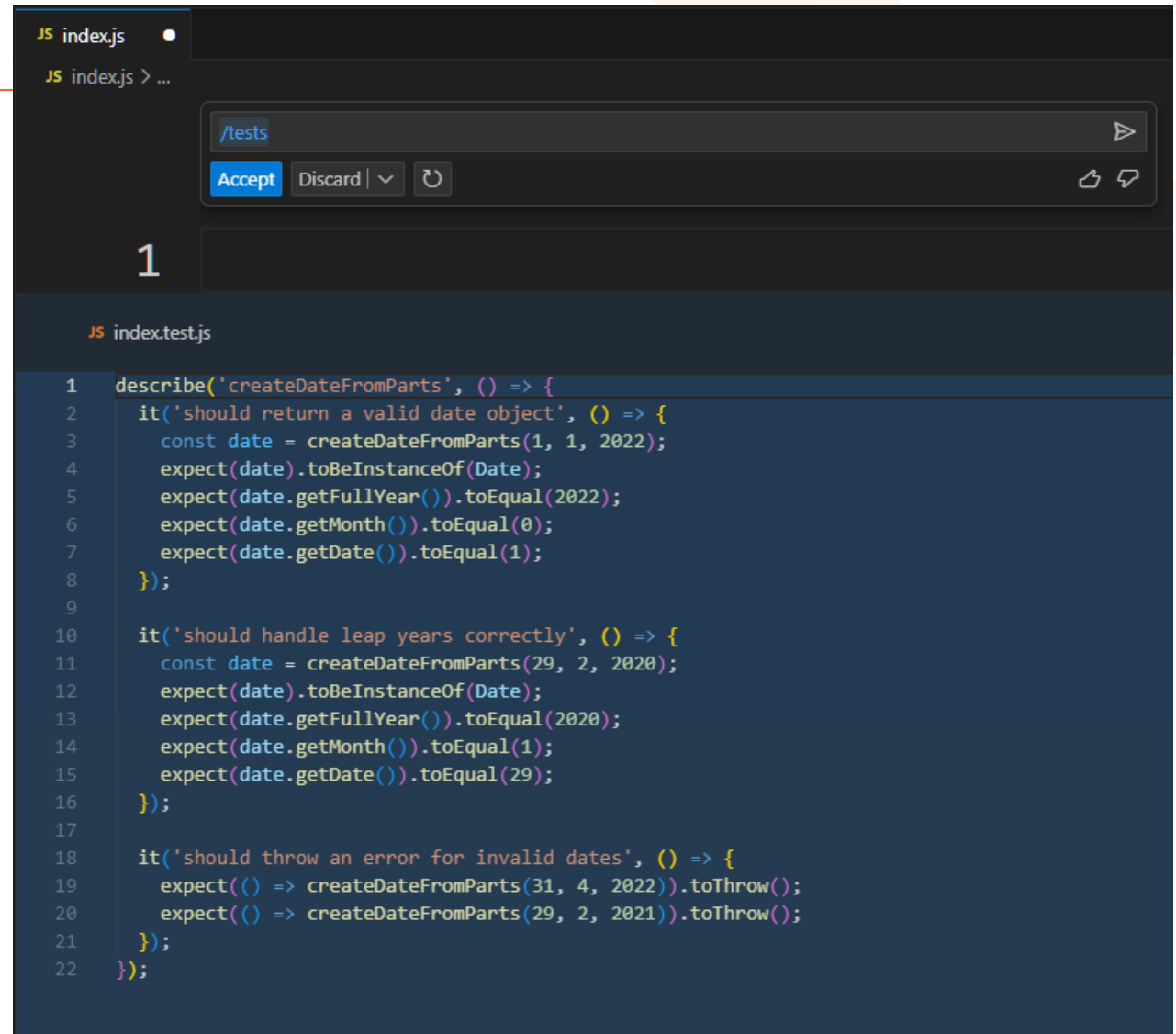
```
1  
2 function createDateFromParts(day, month, year) {  
3-   return new Date → 3+   return new Date(year, month - 1, day);  
5 }
```

The change is highlighted with a red box around `month - 1`. The Copilot chat window shows the "Accept" button and "Discard" option, along with a "Changed 1 line" indicator.



# Creating Tests

- You can generate tests using **/tests**

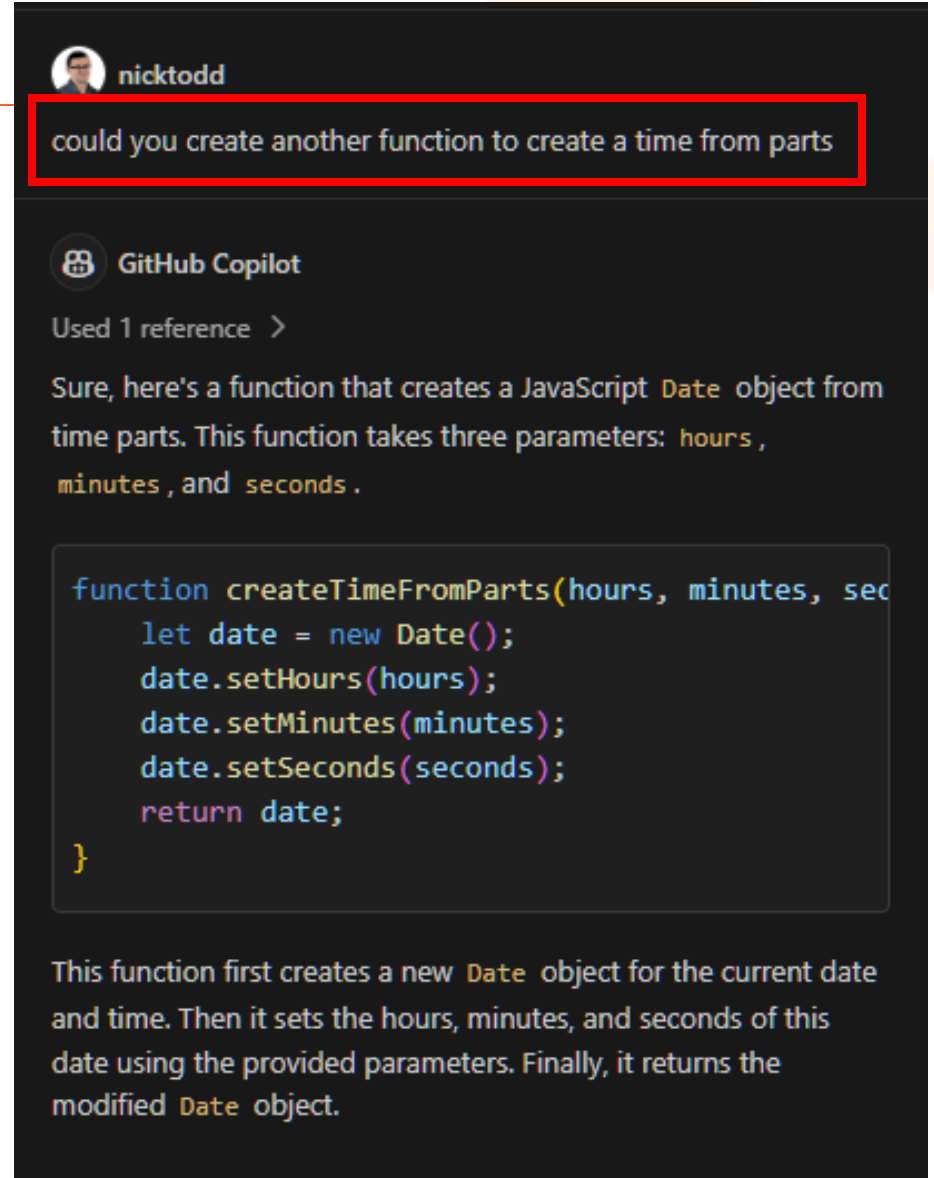


The screenshot shows a code editor with a dark theme. At the top, a file explorer shows 'index.js' and 'index.js > ...'. Below it, a command palette is open with the input '/tests'. The palette has buttons for 'Accept', 'Discard', and a refresh icon. A large number '1' is displayed below the command palette. The main editor area shows a file named 'index.test.js' with the following JavaScript code:

```
1 describe('createDateFromParts', () => {
2   it('should return a valid date object', () => {
3     const date = createDateFromParts(1, 1, 2022);
4     expect(date).toBeInstanceOf(Date);
5     expect(date.getFullYear()).toEqual(2022);
6     expect(date.getMonth()).toEqual(0);
7     expect(date.getDate()).toEqual(1);
8   });
9
10  it('should handle leap years correctly', () => {
11    const date = createDateFromParts(29, 2, 2020);
12    expect(date).toBeInstanceOf(Date);
13    expect(date.getFullYear()).toEqual(2020);
14    expect(date.getMonth()).toEqual(1);
15    expect(date.getDate()).toEqual(29);
16  });
17
18  it('should throw an error for invalid dates', () => {
19    expect(() => createDateFromParts(31, 4, 2022)).toThrow();
20    expect(() => createDateFromParts(29, 2, 2021)).toThrow();
21  });
22 });
```

# GitHub CoPilot Chat

- In addition to the /tests /fix /doc /explain options, there is also a Chat option
- The Chat option is like using ChatGPT capability but context sensitive and within the IDE



The screenshot shows a chat interface with a dark background. At the top, a user profile for 'nicktodd' is visible. Below it, a red-bordered box contains the user's message: 'could you create another function to create a time from parts'. The response from 'GitHub Copilot' follows, indicating it 'Used 1 reference >'. The response text says: 'Sure, here's a function that creates a JavaScript `Date` object from time parts. This function takes three parameters: `hours`, `minutes`, and `seconds`.' Below this text is a code block containing a JavaScript function: 

```
function createTimeFromParts(hours, minutes, seconds) {
  let date = new Date();
  date.setHours(hours);
  date.setMinutes(minutes);
  date.setSeconds(seconds);
  return date;
}
```

 At the bottom, a paragraph explains: 'This function first creates a new `Date` object for the current date and time. Then it sets the hours, minutes, and seconds of this date using the provided parameters. Finally, it returns the modified `Date` object.'

nicktodd

could you create another function to create a time from parts

GitHub Copilot

Used 1 reference >

Sure, here's a function that creates a JavaScript `Date` object from time parts. This function takes three parameters: `hours`, `minutes`, and `seconds`.

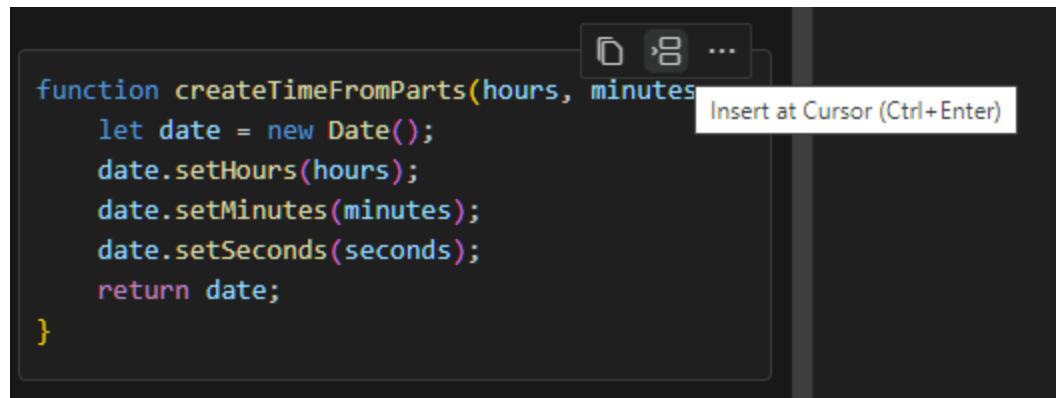
```
function createTimeFromParts(hours, minutes, seconds) {
  let date = new Date();
  date.setHours(hours);
  date.setMinutes(minutes);
  date.setSeconds(seconds);
  return date;
}
```

This function first creates a new `Date` object for the current date and time. Then it sets the hours, minutes, and seconds of this date using the provided parameters. Finally, it returns the modified `Date` object.

# Using the Suggestions

---

- The code can then be inserted into your codebase either at the cursor, or alternatively just copied to the clipboard

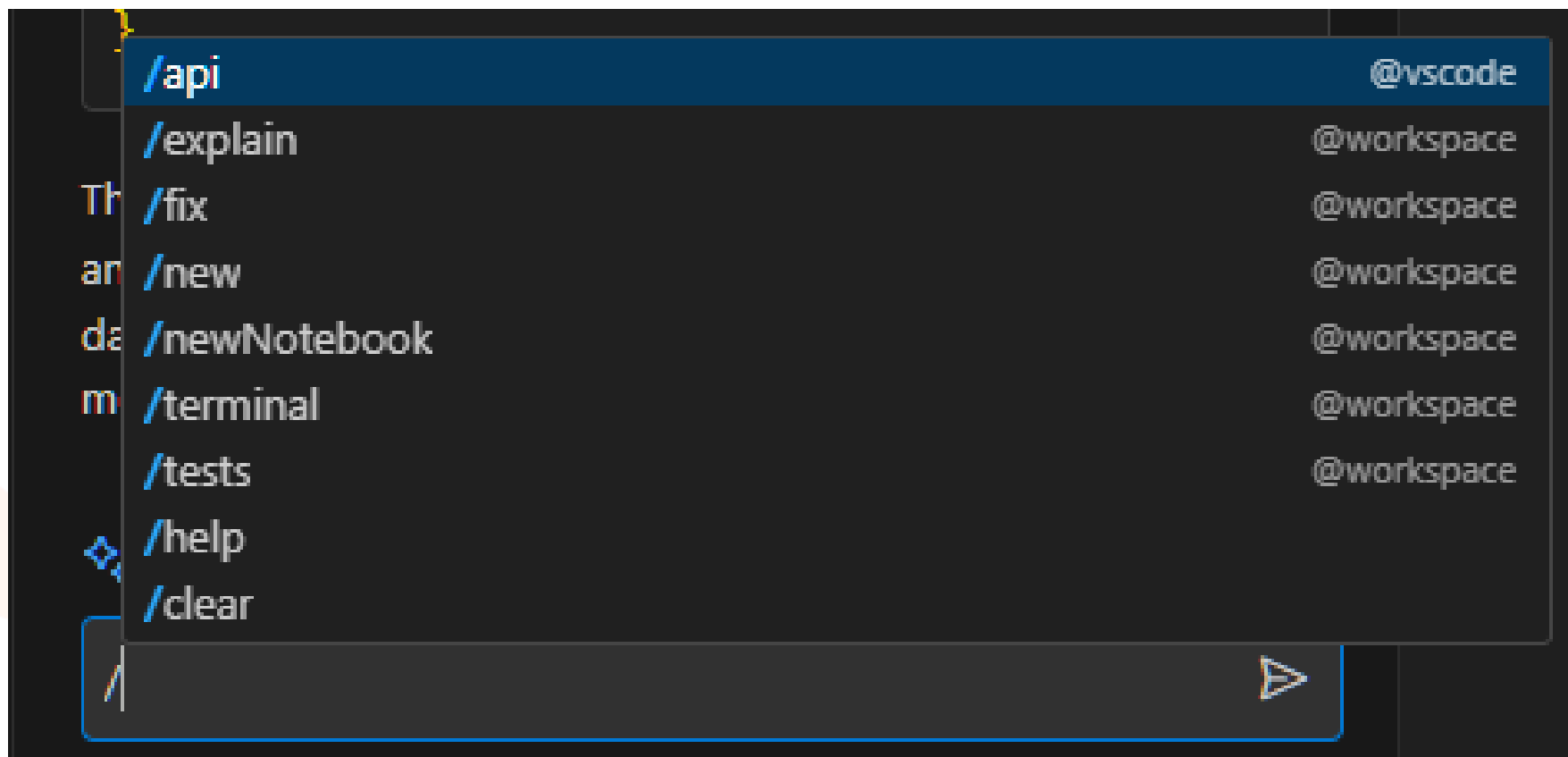


A screenshot of a code editor with a dark theme. It displays a JavaScript function `createTimeFromParts` with parameters `hours` and `minutes`. The function body includes `let date = new Date();`, `date.setHours(hours);`, `date.setMinutes(minutes);`, `date.setSeconds(seconds);`, `return date;`, and a closing brace `}`. A context menu is open over the `minutes` parameter, showing icons for copy, paste, and a menu. A tooltip points to the menu icon with the text "Insert at Cursor (Ctrl+Enter)".

```
function createTimeFromParts(hours, minutes) {  
  let date = new Date();  
  date.setHours(hours);  
  date.setMinutes(minutes);  
  date.setSeconds(seconds);  
  return date;  
}
```

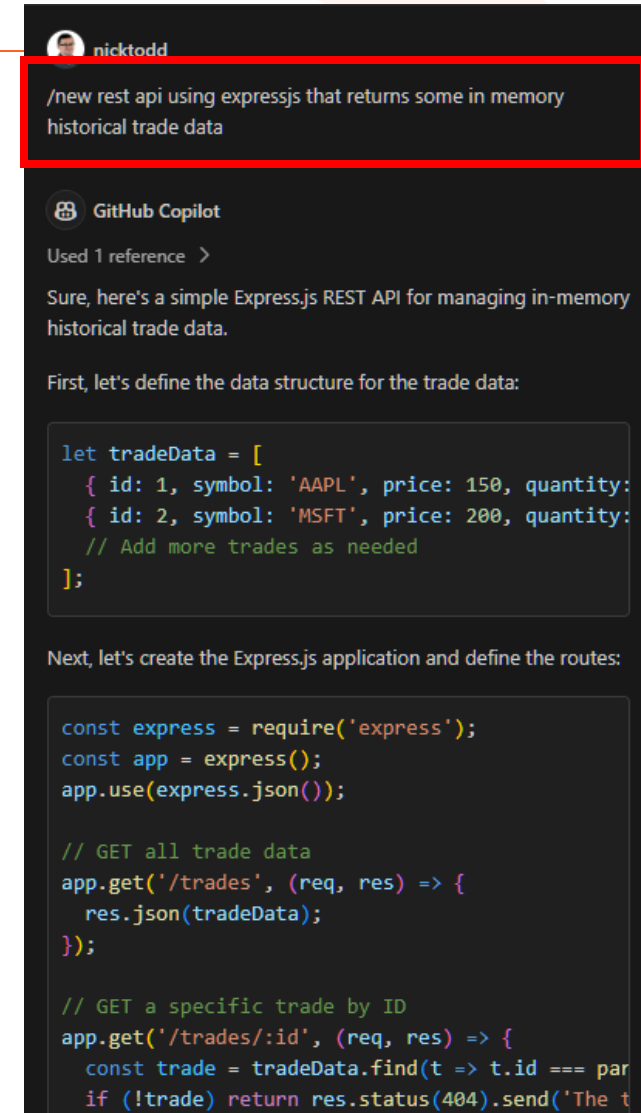
# Chat Commands

- The Chat also introduces a number of additional commands



# Creating Projects using /new

- The **/new** option can be used to create a project from scratch
  - If you type **/new** you will get suggested code fragments



The screenshot shows a GitHub Copilot chat interface. At the top, a user named 'nicktodd' has entered the prompt: '/new rest api using expressjs that returns some in memory historical trade data'. This prompt is highlighted with a red border. Below the prompt, the GitHub Copilot logo is visible, followed by the text 'Used 1 reference >'. The main response from Copilot states: 'Sure, here's a simple Express.js REST API for managing in-memory historical trade data.' It then says 'First, let's define the data structure for the trade data:' and provides a code snippet for an array of trade objects. Finally, it says 'Next, let's create the Express.js application and define the routes:' and provides a code snippet for setting up the Express app and defining GET routes for all trades and by ID.

```
let tradeData = [
  { id: 1, symbol: 'AAPL', price: 150, quantity:
  { id: 2, symbol: 'MSFT', price: 200, quantity:
  // Add more trades as needed
];

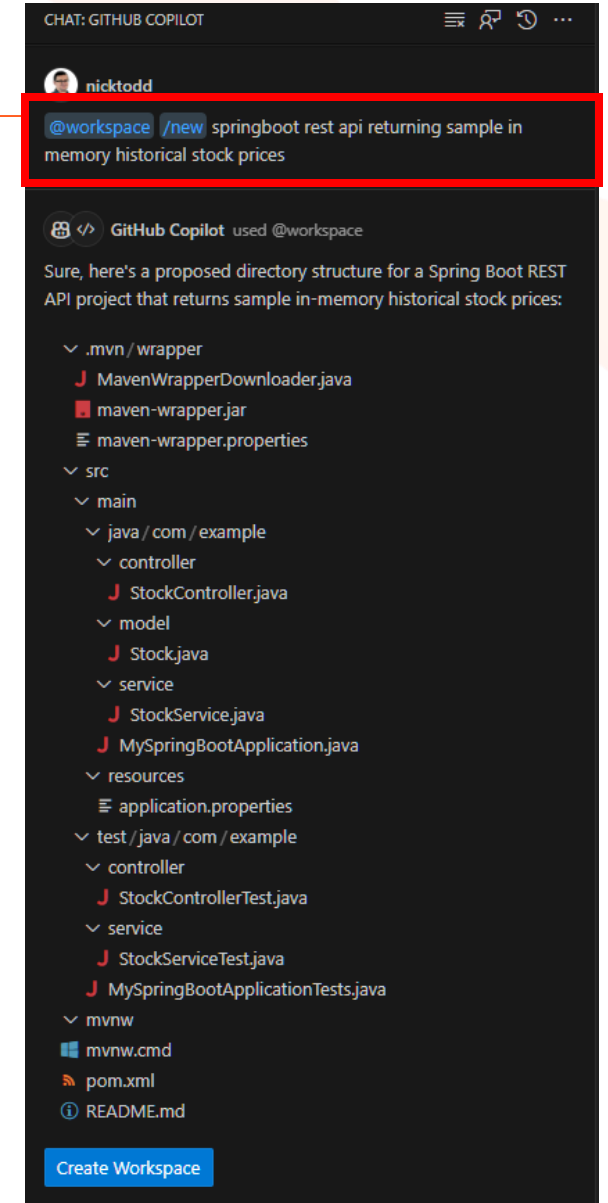
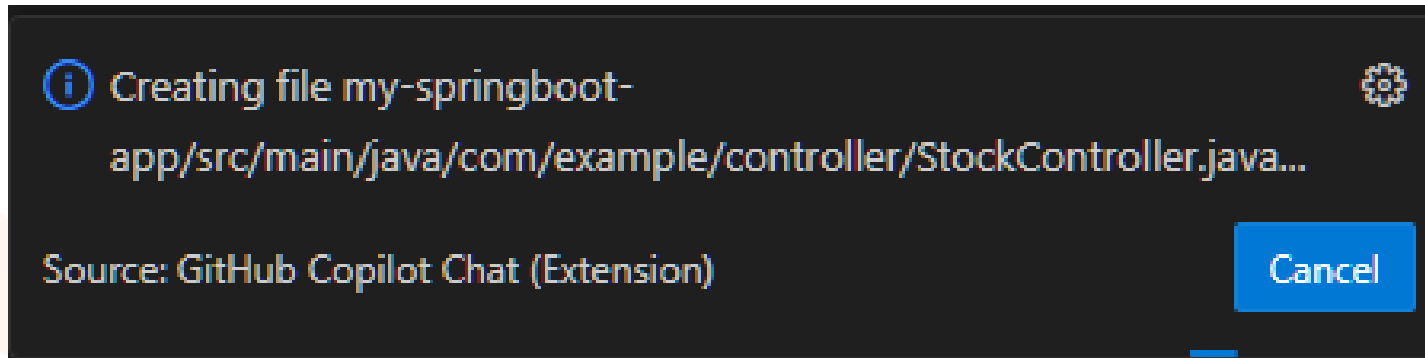
const express = require('express');
const app = express();
app.use(express.json());

// GET all trade data
app.get('/trades', (req, res) => {
  res.json(tradeData);
});

// GET a specific trade by ID
app.get('/trades/:id', (req, res) => {
  const trade = tradeData.find(t => t.id === par
  if (!trade) return res.status(404).send('The t
```

# Creating the Entire Project

- Entire project scaffolds can be created as well if you use the **@workspace** keyword before the **/new** command
- For example, you can create an entire SpringBoot project



# Errors in the Code

---

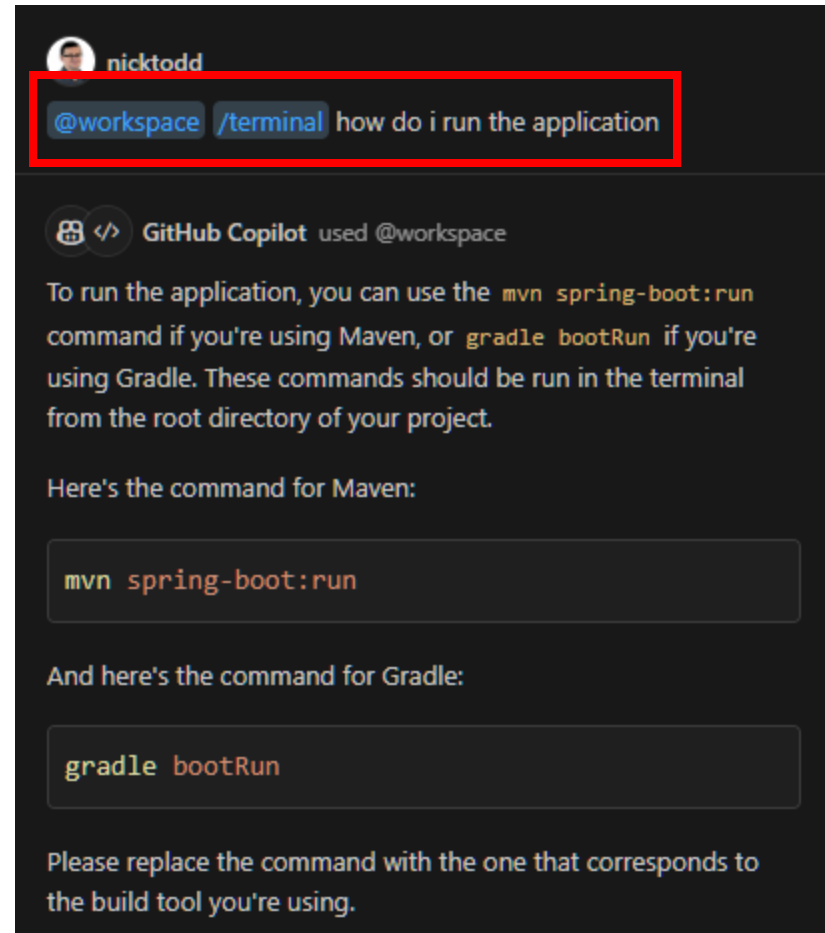
- Although it generates the required scaffolding, it won't necessarily be perfect!
- In the example generated, the entity class had different properties to those used by the constructor call in the service layer

```
@Service
public class StockService {

    private List<Stock> stocks = Arrays.asList(
        new Stock(1L, "Apple Inc.", 150.0),
        new Stock(2L, "Microsoft Corporation", 280.0),
        new Stock(3L, "Amazon.com, Inc.", 3300.0)
    );
}
```

# GitHub CoPilot and the Terminal

- Using **@workspace /terminal** you can ask questions about what to enter at the terminal
- At time of writing, GitHub are creating a CLI AI as well which will mean that you can request AI assistance from the terminal directly





# Upgrading Applications

---

- Tools such as Github CoPilot can be used to upgrade applications from previous versions
- The tool can identify changes that must be made to a project to facilitate the upgrade
- For example, a Java Spring Boot application could be upgraded from one version of Java and one version of SpringBoot
  - The key to this is careful prompting
  - It can be helpful to tell the tool about your own level of understanding so that it can give changes and the relevant context for you to understand the change

# Migrating Legacy Applications

---

- Another real benefit of these tools is the ability to convert from one language to another
  - Perl to Python
  - Java to Rust
  - Angular to React
- You can do things like
  - Go from Legacy - Documentation - New Code
  - Go from Legacy - New Code
- You can experiment with different tools and LLMs
  - Some companies have created AI based utilities to facilitate these tasks

# Emphasising the Co in CoPilot

## How it Works

# How It Works

---

- GitHub Copilot offers suggestions from a model that OpenAI built from billions of lines of open source code
- That means your code will have to be sent to the model in order to get suggestions
- If using it in your own account, you can additionally control whether your code is used to improve the model

☒ Allow GitHub to use my code snippets for product improvements \*

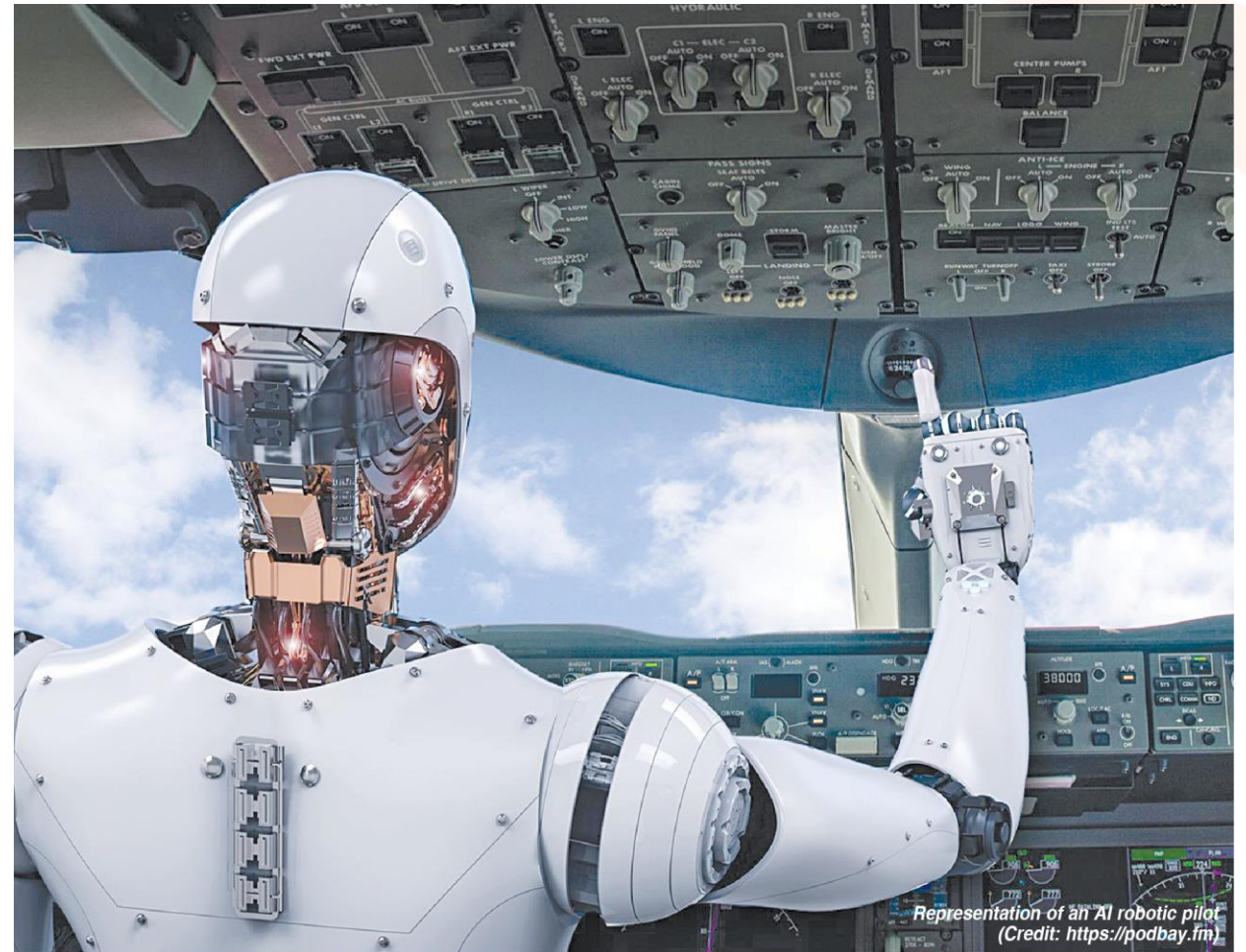
Allow GitHub, its affiliates and third parties to use my code snippets to research and improve GitHub Copilot suggestions, related models and product features.

More information in [Privacy FAQ](#).

Save

# Pilot Flying / Pilot Monitoring

- In commercial aviation there is always
  - A pilot flying
  - A pilot monitoring
- If you are using GitHub CoPilot who is the pilot monitoring?
  - YOU!!
- You CANNOT abdicate responsibility for your application logic to the AI
  - It's not good enough *yet!*



# Precautions

---

- Code Quality

- The training set for GitHub Copilot may contain insecure coding patterns, bugs, or references to outdated APIs or idioms
- When GitHub Copilot produces suggestions based on this training data, those suggestions may also contain undesirable patterns

- Responsibilities

- You are responsible for ensuring the security and quality of your code
- GitHub recommend the following precautions are followed
  - Rigorous testing
  - IP scanning
  - Tracking for security vulnerabilities



vas



@vasumanmoza

Follow



Claude 4 just refactored my entire codebase in one call.

25 tool invocations. 3,000+ new lines. 12 brand new files.

It modularized everything. Broke up monoliths. Cleaned up spaghetti.

None of it worked.  
But boy was it beautiful.

# Best Practices

---


- Effective Prompts
  - Make it clear
  - Break it down
  - Provide examples
  - Consider Neighbouring tabs
  - Consider code above and below – Fill in the Middle
  - Good naming
- Testing
- Regular Expressions




# Prompting - Make it Clear

- Give good context prompts
  - *I want a function that returns a date from parts*
  - *I want a function that returns a date from a day month and year passed as parameters*

```
JS GitHub Copilot Suggestions for index.js 3 x
1 Synthesizing 9/10 solutions
2
3 Accept Solution
4 =====
5 Suggestion 1
6 // I want a function that returns a date from parts.
7 //
8
9
10 Accept Solution
11 =====
12 Suggestion 2
13 // I want a function that returns a date from parts
14 // I want a function that returns a date from a string
15 // I want a function that returns a date from a date
16
17 const date = require('./date');
18
19 const dateFromParts = date.dateFromParts;
20 const dateFromString = date.dateFromString;
21 const dateFromDate = date.dateFromDate;
22
23 module.exports = {
24   dateFromParts,
25   dateFromString,
26   dateFromDate,
27 };
28
```



```
Synthesizing 9/10 solutions
Accept Solution
=====
Suggestion 1
// I want a function that returns a date from a day month and year passed as parameters.
// This is how I would do it:
function myFunction(day, month, year) {
  return new Date(year, month - 1, day);
}
console.log(myFunction(1, 2, 2020));
// If you run this code you will see that it returns the date 2020-01-31.
```



# Prompting - Break it Down

---

- Break your request down into bytesize chunks

```
// I want a class for a Car  
// with a make and model and get and set methods for the properties  
// i also want a speed property that defaults to 0  
// and I want an accelerate method that increments the speed by 10  
// i also want a brake method that decrements the speed by 10
```



# Prompting - Provide Examples

---

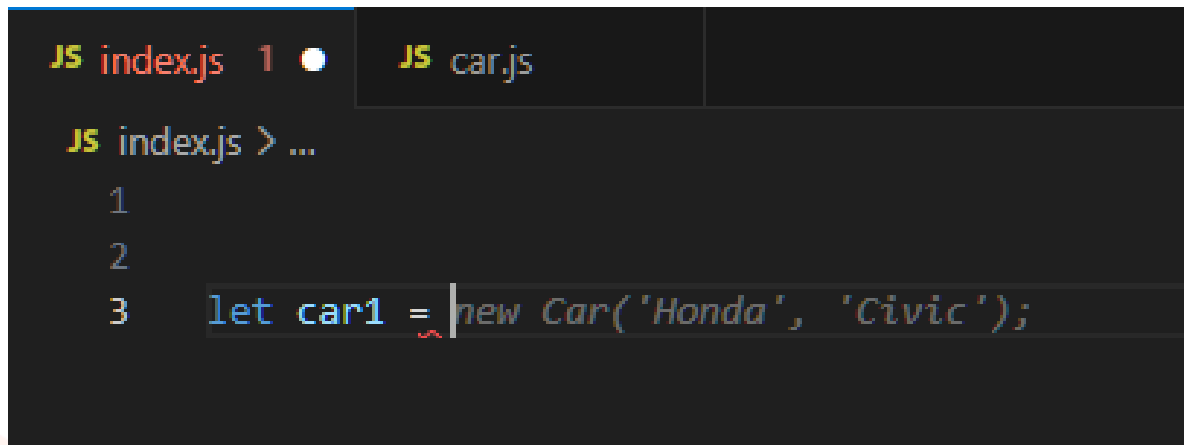
- When prompting, provide examples
  - Copilot will also help you with your prompts!!

```
let people = [  
  { name: 'John', age: 21 },  
  { name: 'Peter', age: 31 },  
  { name: 'Sarah', age: 41 },  
]  
  
// i want an array of just the names eg. ['John', 'Peter', 'Sarah']
```



# Prompting - Neighbouring Tabs

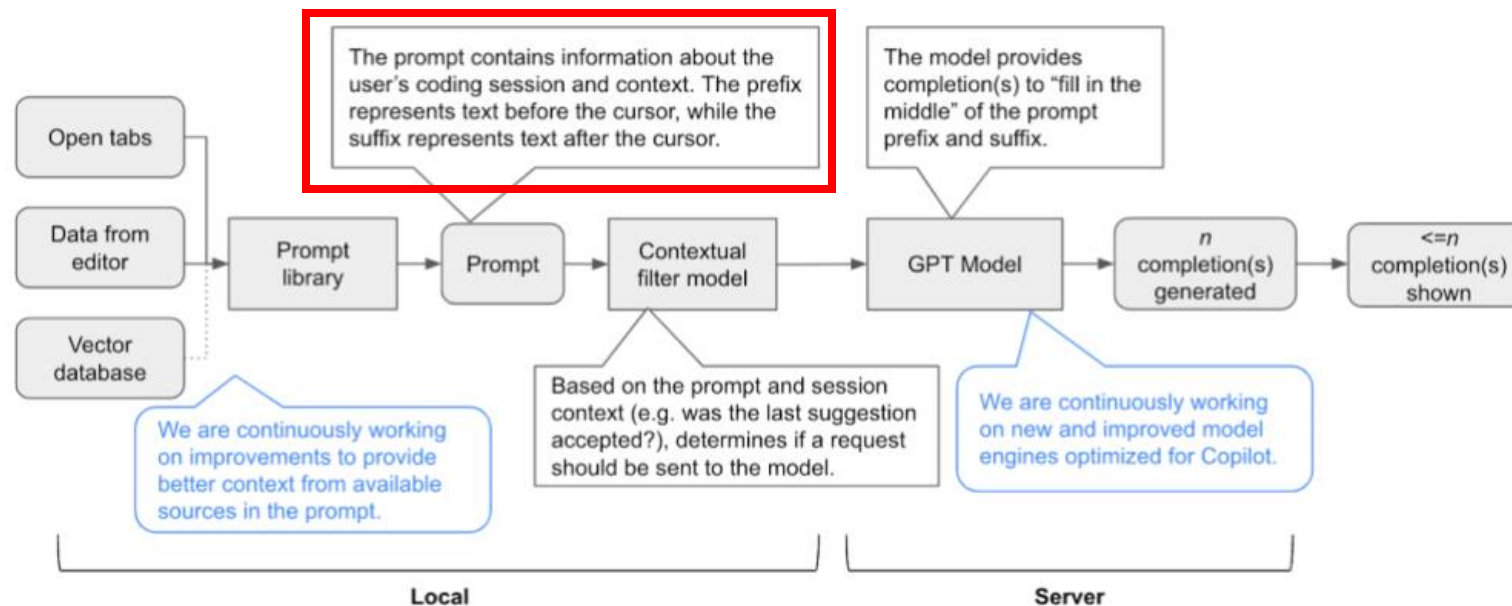
- CoPilot has a feature called Neighbouring tabs
  - ***“Neighboring tabs** is what we call the technique that allows GitHub Copilot to process all of the files open in a developer’s IDE instead of just the single one the developer is working on. By opening all files relevant to their project, developers automatically invoke GitHub Copilot to comb through all of the data and find matching pieces of code between their open files and the code around their cursor—and add those matches to the prompt”*



```
JS index.js 1 ● JS car.js
JS index.js > ...
1
2
3 let car1 = new Car('Honda', 'Civic');
```

# Prompting - Fill in the Middle


- The Code that is sent to GitHub includes the code before *and after* your cursor so it gets better context
  - This is referred to as the **Fill in the Middle Paradigm**




Simplified system diagram focused on model quality efforts. Made by Alice Li, machine learning researcher at GitHub.

# Prompting - Use Good Naming

- As we all know from our development experience, good naming is important to understand code
  - This is also true for your CoPilot!



```
function createDateFromParts(year, month, day) {  
    return new Date(year, month - 1, day)  
}
```



```
function makeDate() {  
    var d = new Date();  
    var formattedDate = "";  
  
    formattedDate += (d.getMonth() + 1) + "_";  
    formattedDate += d.getDate() + "_";  
    formattedDate += d.getFullYear();  
  
    return formattedDate;  
}
```

# Testing

---

- Generating tests - be careful
- Consider the simple date from parts example, this could be what you do
  1. Generate some code to create the date (it uses the month with index 0)
  2. You then generate a test (it will simply test that the month parses with an index of 0)
  3. The test will pass
- The code is however wrong - you trusted your copilot

# 1. Create an Incorrect Function

---

- This should be **month -1** but you didn't notice

```
// export the createDateFromParts function
module.exports = createDateFromParts;

// function to create a date from the parts
function createDateFromParts(day, month, year)
{
    return new Date(year, month, day);
}
```





## 2. Now Create the Incorrect Test – That Passes!

---

- It will create a passing test for your incorrect function
  - The test uses 0s for the months because the function being tested does!
- Everything passes!!

### Suggestion 2


```
// create a unit test for the index.js file
//
// Path: index.test.js
//
// // import the function
// const createDateFromParts = require('./index.js');
//
// // test the function
// describe('createDateFromParts()', () => {
//   it('creates a valid date', () => {
//     expect(createDateFromParts(1, 0,
// 2000)).toEqual(new Date(2000, 0, 1));
//   });
// });
```



# Testing - An Alternative Approach


---

1. When generating tests, consider describing the test that is required without referring to your code being tested



```
// create a unit test for a function called createDateFromParts
// the function is in a file called index.js
// the parts are day, month and year
// and the month should account for the offset of 1
```

Suggestion 1



```
const createDateFromParts = require('./index');

test('should return a date object', () => {
  expect(createDateFromParts(1, 1, 2020)).toBeInstanceOf(Date);
});

test('should return a date object for 1st Jan 2020', () => {
  expect(createDateFromParts(1, 1, 2020)).toEqual(new Date(2020, 0, 1));
});
```

# Regular Expressions

---

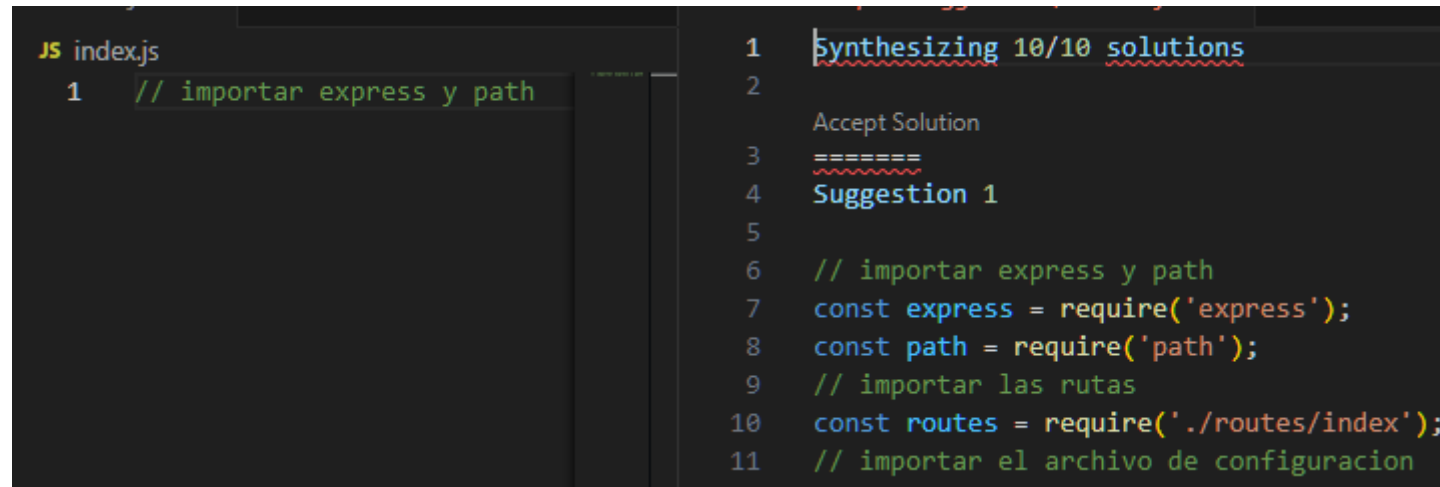
- GitHub CoPilot is very good at creating regular expressions and can save you a lot of time

```
JS index.js > ...  
1  function validatePhoneNumber (phoneNumber) {  
    const regExp = /^\\d{10}$/;  
    return regExp.test(phoneNumber);  
2  
3  }
```

# Translation

---

- For non-English speakers, you can use CoPilot in different languages



The image shows a side-by-side comparison of a code editor. On the left, a file named 'index.js' is open, showing a single line of code: `// importar express y path`. On the right, a dropdown menu displays a suggestion for the same line of code, but in Spanish. The suggestion is titled 'Synthesizing 10/10 solutions' and 'Accept Solution'. Below this, it says 'Suggestion 1' followed by the code: `// importar express y path`, `const express = require('express');`, `const path = require('path');`, `// importar las rutas`, `const routes = require('./routes/index');`, and `// importar el archivo de configuracion`.

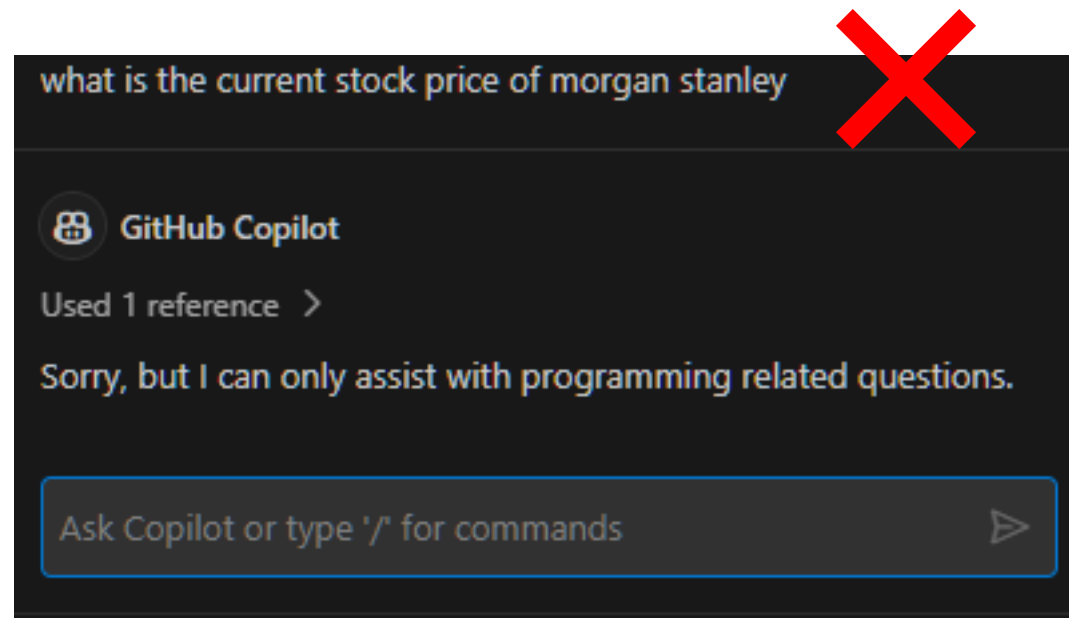
```
JS index.js
1 // importar express y path

1 Synthesizing 10/10 solutions
2 Accept Solution
3 =====
4 Suggestion 1
5
6 // importar express y path
7 const express = require('express');
8 const path = require('path');
9 // importar las rutas
10 const routes = require('./routes/index');
11 // importar el archivo de configuracion
```

# Don't use it as a Chat GPT Alternative!

---

- Do not try and use it for questions outside of the coding space
- It is not designed for that



# Exercise - Creating a Project

---

- Think about a small project. / 设想一个小项目。
- Open the chat window in Visual Studio Code and type **@workspace /new** and then describe your project. / 在 Visual Studio Code 中打开聊天窗口，输入 @workspace /new，然后描述你的项目。
- Experiment with code completion and chat. / 尝试代码补全和聊天。
- Create test, create bugs and let the AI try to fix it. / 创建测试，发现错误，并让 AI 尝试修复。
- Work together in 2 groups on your table. / 在你的桌子上分成两组一起合作。
- Share your successes and frustrations with the other group on your table. / 与同桌的另一组分享你的成功和挫折。
- You have 60 minutes, let's see how far you got. / 你有 60 分钟的时间，让我们看看你完成了多少。

# Suggested Applications

---

- You can create anything you like, but if you are struggling for an idea, you could try any of the below
  - A **speaking clock** application that gives you a time in text for any given time of day - ideally it will work with the current time, but you will need to test it to make sure it works for any time of day
  - A **stock code lookup** service. You provide it with a stock code and it performs a lookup and returns some information about the stock
  - A **currency converter** API
  - A python script to use your webcam for **face detection**
  - An API that takes stock trades and fakes the processing of a transaction

# Exercise – presentation

---

- Make a presentation about your experience with the AI at your table / 做一个关于你与你桌边的人工智能互动的演示。
- Make it short (1-2 minutes) / 请简短一些（1-2 分钟）。
- Use English if possible (project presentation will be in English) / 尽可能使用英语（项目演示将使用英语）。
- This is just for training / 这仅用于培训。



# Summary

---

- Overview of Gen AI Developer Tools
- Features of Github CoPilot
- Understanding the importance of the Co in CoPilot
- Best Practices