GitHub Copilot

GitHub – 101

GitHub Copilot Introduction

Agenda

Copilot Overview

A Peek into AI-Powered Coding

Getting Started

Your First Steps with Copilot

Prompt Engineering

Crafting Effective Commands

Tips & Tricks

Maximizing Copilot Efficiency

Frequently Asked Questions

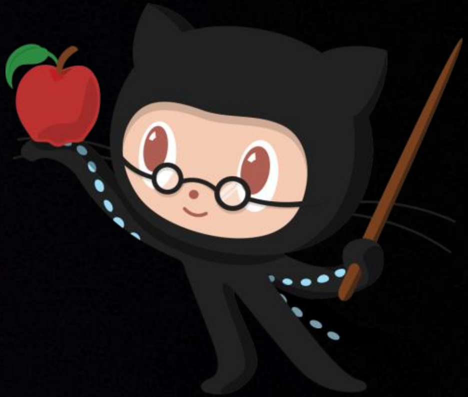
Clarifying Common Curiosities

Copilot Overview

A Peek into AI-Powered Coding

What is Copilot?

> *“A new AI-powered code completion tool that helps developers write code faster.”*
>
> GitHub



Who is Copilot for?

Senior Engineers

Accelerate boilerplate code generation leverage design patterns and simplify complex problem-solving.

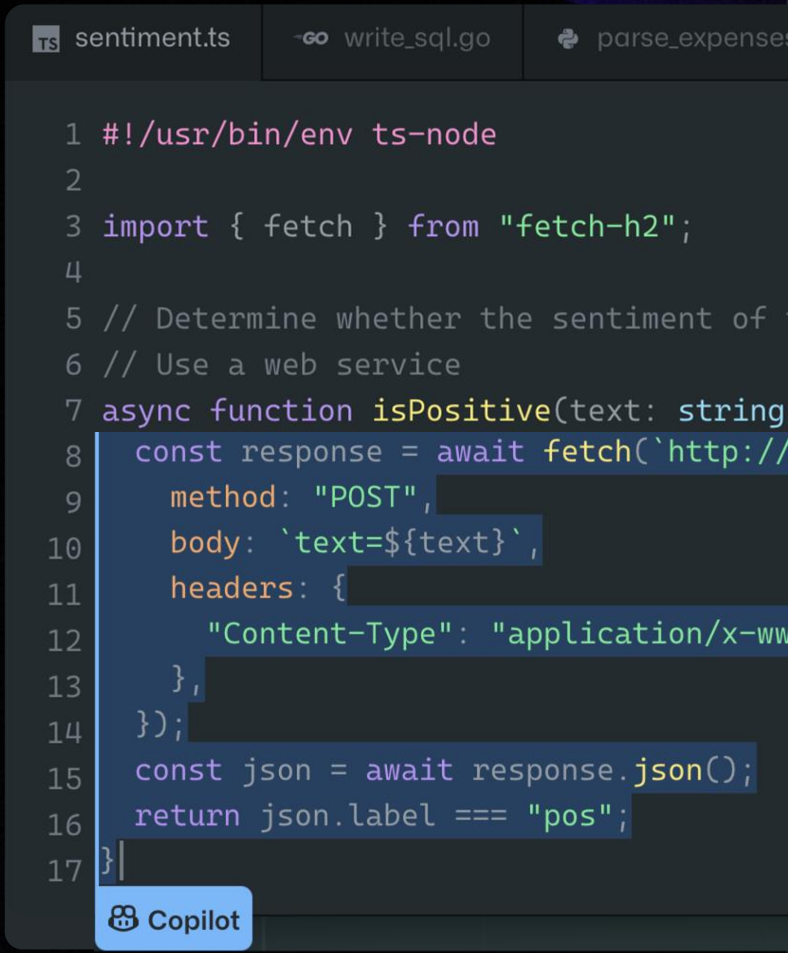
Junior Engineers

Expedite learning, facilitate as a pair programmer, and mentor.

GitHub Copilot

Offers code suggestions, contextualized to your project

- ✓ Convert comments to code
- ✓ Autofill for repetitive code
- ✓ Show alternatives



```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";

// Determine whether the sentiment of
// Use a web service
async function isPositive(text: string
  const response = await fetch(`http://
    method: "POST",
    body: `text=${text}`,
    headers: {
      "Content-Type": "application/x-ww
    },
  });
  const json = await response.json();
  return json.label === "pos";
}
```

Copilot Overview

Copilot

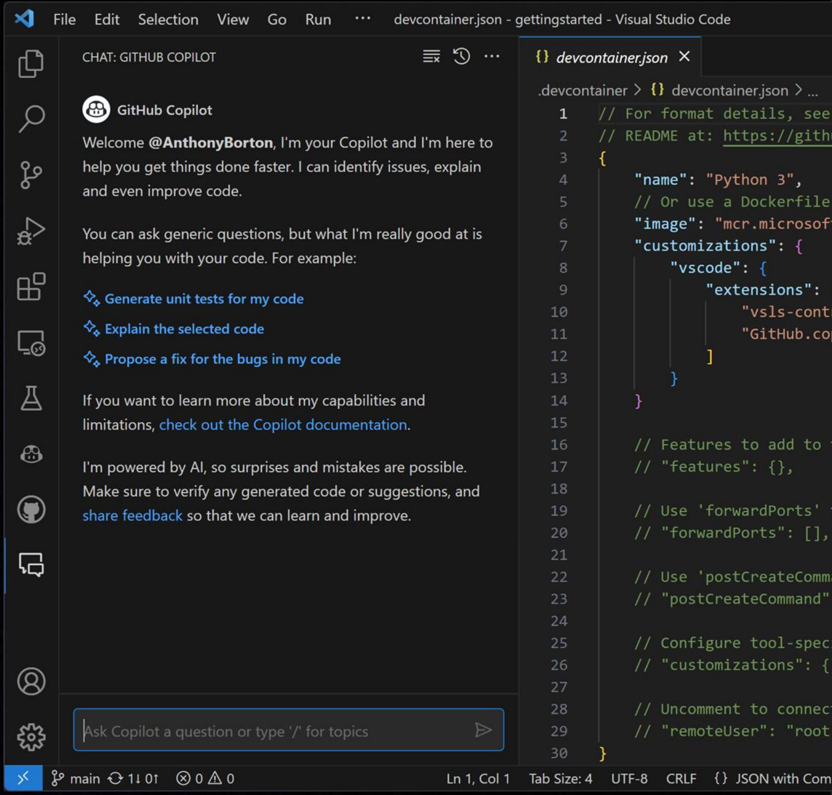
Copilot Chat

Copilot Chat

Context-aware ChatGPT-styled conversations with Copilot

Slash commands

/explain	Explains how the selected code works
/fix	Proposes a fix for the problems in the selected code
/help	General help about GitHub Copilot
/tests	Generate unit tests for the selected code
/vscode	Ask questions about VS Code
/clear	Clear the session



GitHub Copilot

Welcome **@AnthonyBorton**, I'm your Copilot and I'm here to help you get things done faster. I can identify issues, explain and even improve code.

You can ask generic questions, but what I'm really good at is helping you with your code. For example:

- Generate unit tests for my code
- Explain the selected code
- Propose a fix for the bugs in my code

If you want to learn more about my capabilities and limitations, check out the Copilot documentation.

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.

Ask Copilot a question or type '/' for topics

```
// For format details, see
// README at: https://gith
{
    "name": "Python 3",
    // Or use a Dockerfile
    "image": "mcr.microsof
    "customizations": {
        "vscode": {
            "extensions":
                "vsls-cont
                "GitHub.co
            ]
        }
    }

    // Features to add to
    // "features": {},

    // Use 'forwardPorts'
    // "forwardPorts": [],

    // Use 'postCreateComm
    // "postCreateCommand"

    // Configure tool-spec
    // "customizations": {

    // Uncomment to connec
    // "remoteUser": "root
}
```

Limitations

> *“I’m powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.”*
>
> Copilot

Copilot “Wisdom”

Every Copilot needs a Captain !

> *“If you don’t know where you are going you’ll end up someplace else.”*
>
> Yogi Berra

Copilot Impact

96% say they are faster with repetitive tasks

88% say they are more productive

74% focus on more satisfying work

Increase developer productivity

And satisfaction by focusing on real problems

Accelerate innovation

Prototype & innovate more rapidly

Bridge skill gaps

Learn new languages and techniques

Getting Started

Your First Steps with Copilot

What is Required ?

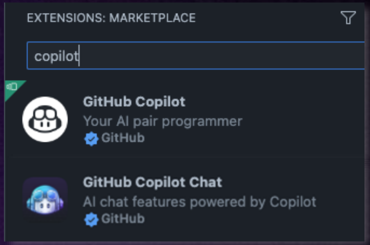
GitHub account

Copilot license

Supported IDE

Copilot in Your IDE

IDE Extensions

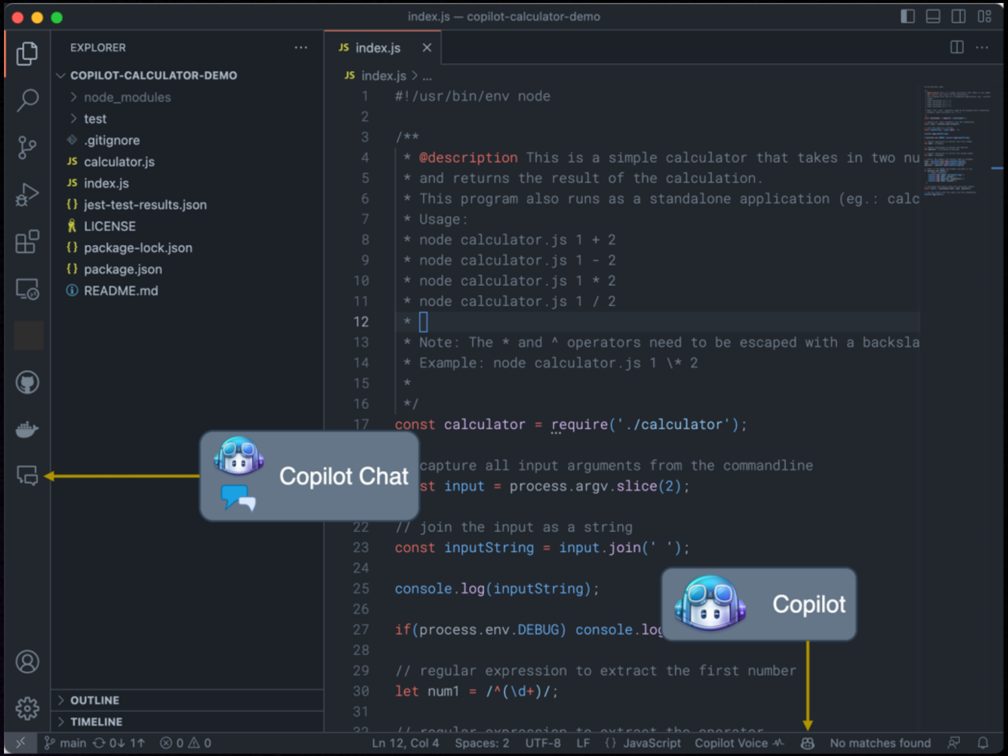


EXTENSIONS: MARKETPLACE

copilot

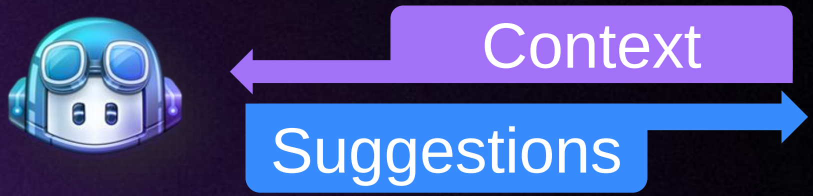
GitHub Copilot
Your AI pair programmer
GitHub

GitHub Copilot Chat
AI chat features powered by Copilot
GitHub

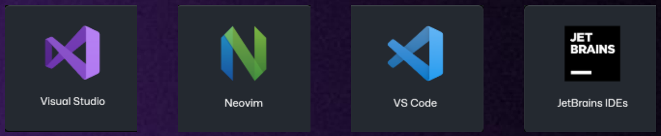


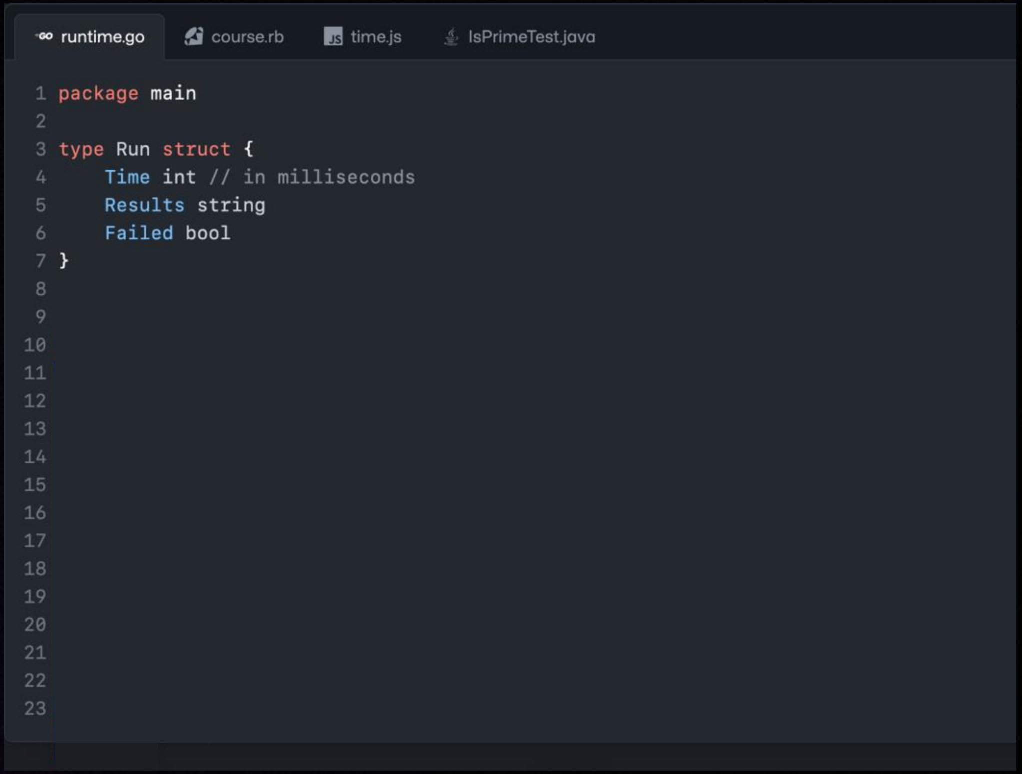
```
#!/usr/bin/env node

/**
 * @description This is a simple calculator that takes in two nu
 * and returns the result of the calculation.
 * This program also runs as a standalone application (eg.: calc
 * Usage:
 * node calculator.js 1 + 2
 * node calculator.js 1 - 2
 * node calculator.js 1 * 2
 * node calculator.js 1 / 2
 *
 * Note: The * and ^ operators need to be escaped with a backsla
 * Example: node calculator.js 1 \* 2
 *
 */
const calculator = require('./calculator');

capture all input arguments from the commandline
st input = process.argv.slice(2);

// join the input as a string
const inputString = input.join(' ');

console.log(inputString);

if(process.env.DEBUG) console.lo

// regular expression to extract the first number
let num1 = /^(\d+)/;
```

Once Enabled…


Context

Suggestions





```go
package main

type Run struct {
    Time int // in milliseconds
    Results string
    Failed bool
}
```

Demo

Copilot & Copilot Chat Interaction

- **Copilot**
  - Inline help
  - Open the Completions Panel
  - Fix using Copilot
  - Explain using Copilot
- **Copilot Chat**
  - Say Hello
  - Check its limits
  - Test its knowledge
  - Topics ‘/’

Prompt Engineering

Crafting Effective Commands

What is a Prompt?

In the context of Copilot, a prompt is a piece of code or natural language description that is used to generate code suggestions. It is the input that Copilot uses to generate its output.

- Copilot

What is Prompt Engineering?

Prompt engineering is the process of designing and creating high-quality prompts that can be used to generate accurate and useful code suggestions with Copilot.

- Copilot

Why is Prompt Engineering Important?

- Maximizes utility of Copilot
- Minimizes repeated iterations
- Determines code suggestion quality
- Skill prompts guide Copilot to understand context & nuances
- Refined interactions reduces misunderstandings

Techniques

Zero-Shot Prompting

No Example

One-Shot Prompting

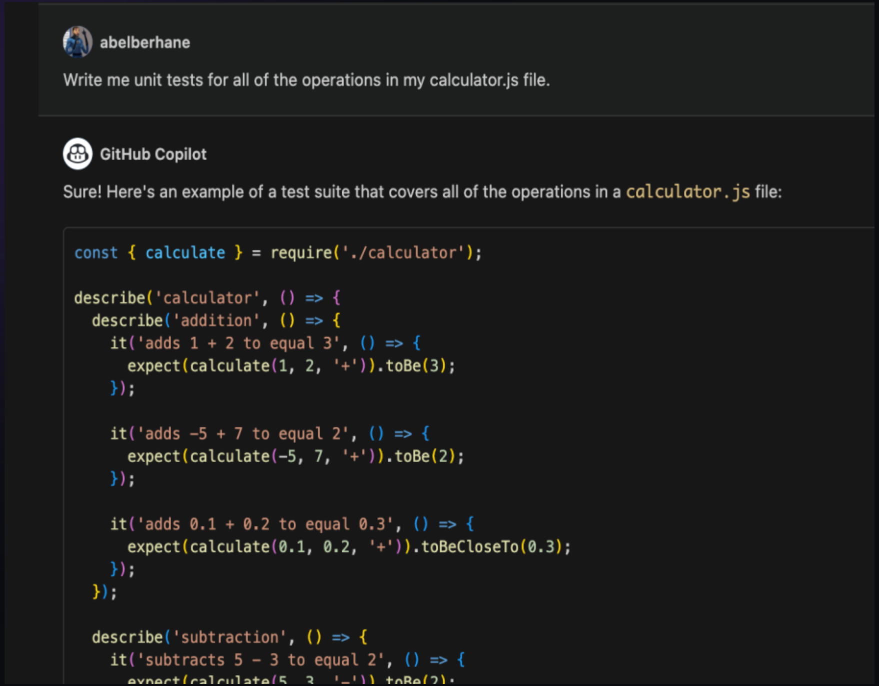
One Example

Few-Shot Prompting

Handful of Examples

Techniques

Zero-Shot Programming with Copilot



abelberhane

Write me unit tests for all of the operations in my calculator.js file.

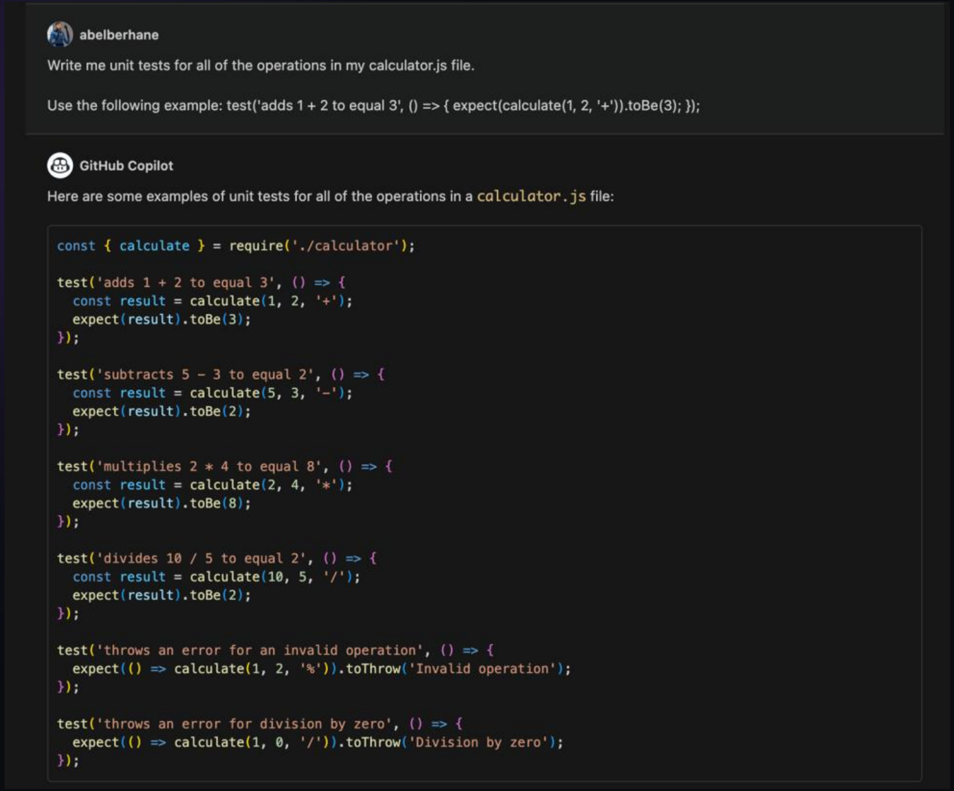
GitHub Copilot

Sure! Here's an example of a test suite that covers all of the operations in a `calculator.js` file:

```
const { calculate } = require('./calculator');

describe('calculator', () => {
  describe('addition', () => {
    it('adds 1 + 2 to equal 3', () => {
      expect(calculate(1, 2, '+')).toBe(3);
    });

    it('adds -5 + 7 to equal 2', () => {
      expect(calculate(-5, 7, '+')).toBe(2);
    });

    it('adds 0.1 + 0.2 to equal 0.3', () => {
      expect(calculate(0.1, 0.2, '+')).toBeCloseTo(0.3);
    });
  });

  describe('subtraction', () => {
    it('subtracts 5 - 3 to equal 2', () => {
      expect(calculate(5, 3, '-')).toBe(2);
```

Techniques

One-shot Programming with Copilot



abelberhane

Write me unit tests for all of the operations in my calculator.js file.

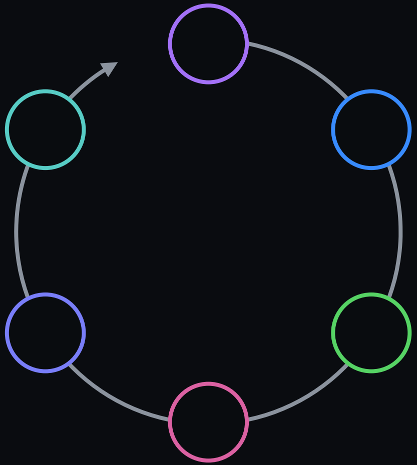
Use the following example: test('adds 1 + 2 to equal 3', () => { expect(calculate(1, 2, '+')).toBe(3); });

GitHub Copilot

Here are some examples of unit tests for all of the operations in a `calculator.js` file:

```
const { calculate } = require('./calculator');

test('adds 1 + 2 to equal 3', () => {
  const result = calculate(1, 2, '+');
  expect(result).toBe(3);
});

test('subtracts 5 - 3 to equal 2', () => {
  const result = calculate(5, 3, '-');
  expect(result).toBe(2);
});

test('multiplies 2 * 4 to equal 8', () => {
  const result = calculate(2, 4, '*');
  expect(result).toBe(8);
});

test('divides 10 / 5 to equal 2', () => {
  const result = calculate(10, 5, '/');
  expect(result).toBe(2);
});

test('throws an error for an invalid operation', () => {
  expect(() => calculate(1, 2, '%')).toThrow('Invalid operation');
});

test('throws an error for division by zero', () => {
  expect(() => calculate(1, 0, '/')).toThrow('Division by zero');
});
```

Cornerstones of Prompting



Control Output Verbosity

Chaining Prompts

Provide Examples

Evaluate Quality

Give Direction

Format your Response

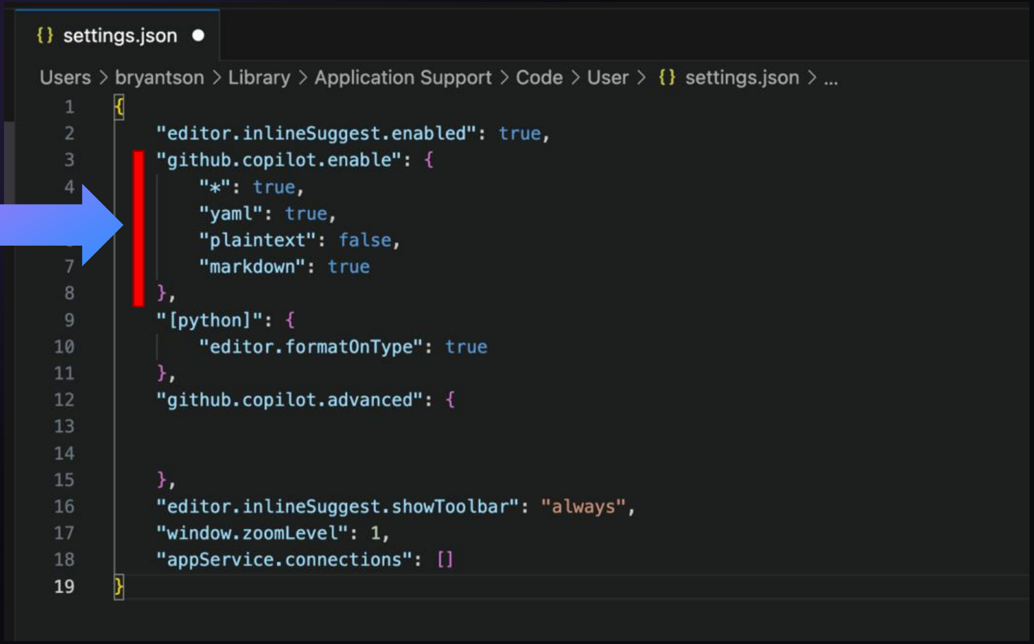
Tips & Tricks

Maximizing Copilot Efficiency

Enable non-programming Meta Files

In **settings.json**, add

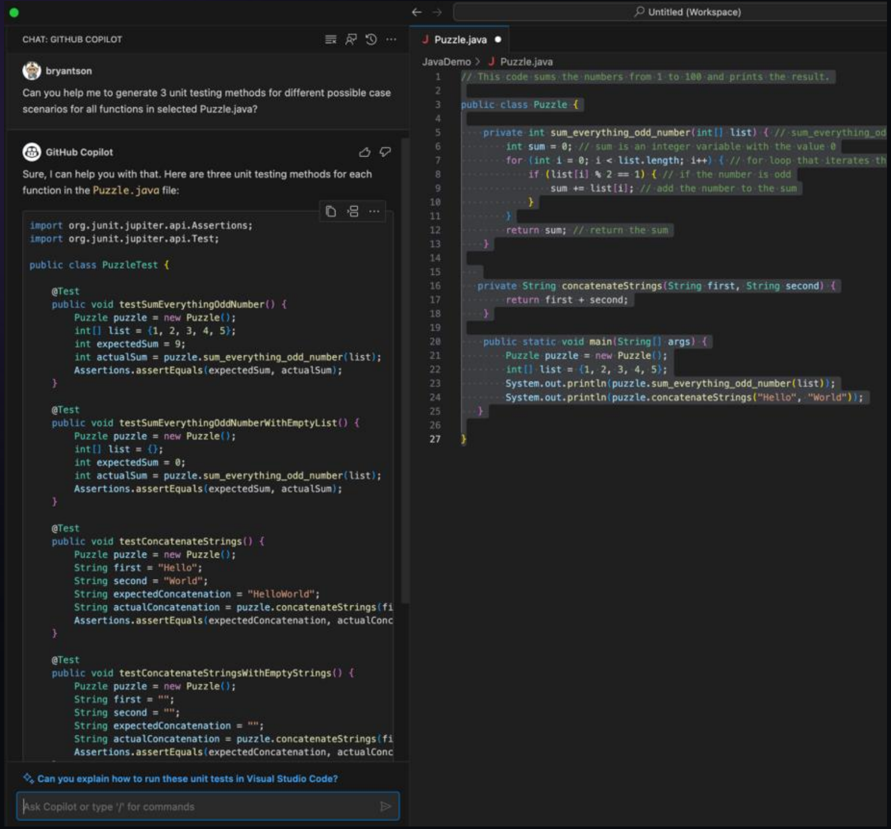
```
“github.copilot.enable”:
{
   “*”: true,
  “yaml”: true,
  “plaintext”: false,
  “markdown”: true
}
```


{} settings.json ●
Users > bryantson > Library > Application Support > Code > User > {} settings.json > ...

```
{
    "editor.inlineSuggest.enabled": true,
    "github.copilot.enable": {
        "*": true,
        "yaml": true,
        "plaintext": false,
        "markdown": true
    },
    "[python]": {
        "editor.formatOnType": true
    },
    "github.copilot.advanced": {


    },
    "editor.inlineSuggest.showToolbar": "always",
    "window.zoomLevel": 1,
    "appService.connections": []
}
```

Unit Testing Generation - Copilot Chat

Works best for general unit testing case generation



bryantson

Can you help me to generate 3 unit testing methods for different possible case scenarios for all functions in selected Puzzle.java?

GitHub Copilot

Sure, I can help you with that. Here are three unit testing methods for each function in the `Puzzle.java` file:

```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class PuzzleTest {

    @Test
    public void testSumEverythingOddNumber() {
        Puzzle puzzle = new Puzzle();
        int[] list = {1, 2, 3, 4, 5};
        int expectedSum = 9;
        int actualSum = puzzle.sum_everything_odd_number(list);
        Assertions.assertEquals(expectedSum, actualSum);
    }

    @Test
    public void testSumEverythingOddNumberWithEmptyList() {
        Puzzle puzzle = new Puzzle();
        int[] list = {};
        int expectedSum = 0;
        int actualSum = puzzle.sum_everything_odd_number(list);
        Assertions.assertEquals(expectedSum, actualSum);
    }

    @Test
    public void testConcatenateStrings() {
        Puzzle puzzle = new Puzzle();
        String first = "Hello";
        String second = "World";
        String expectedConcatenation = "HelloWorld";
        String actualConcatenation = puzzle.concatenateStrings(fi
        Assertions.assertEquals(expectedConcatenation, actualConc
    }

    @Test
    public void testConcatenateStringsWithEmptyStrings() {
        Puzzle puzzle = new Puzzle();
        String first = "";
        String second = "";
        String expectedConcatenation = "";
        String actualConcatenation = puzzle.concatenateStrings(fi
        Assertions.assertEquals(expectedConcatenation, actualConc
```

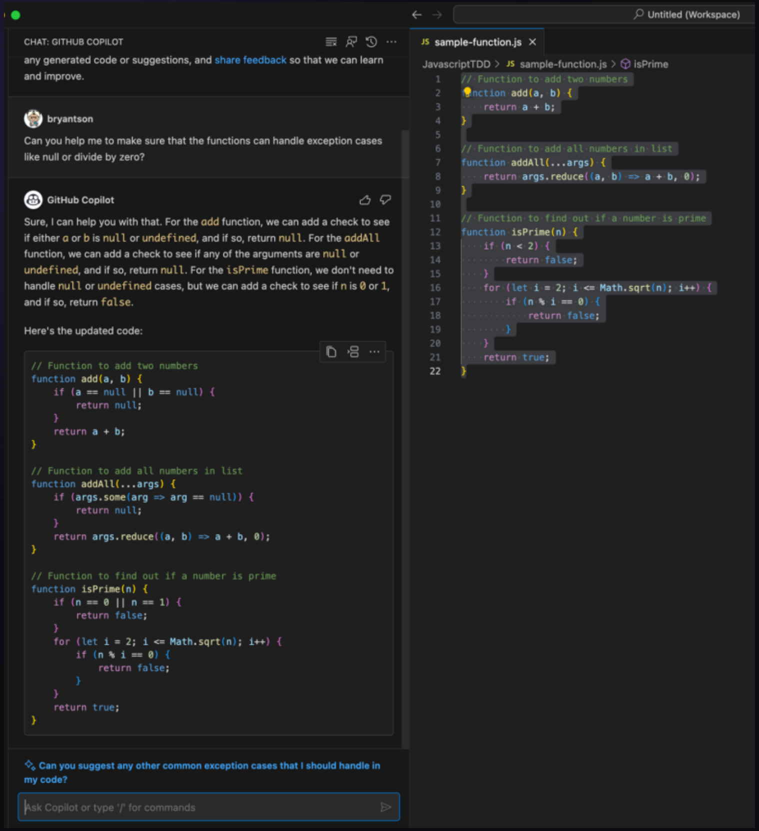
Can you explain how to run these unit tests in Visual Studio Code?

Ask Copilot or type '/' for commands

```
// This code sums the numbers from 1 to 100 and prints the result.

public class Puzzle {

    private int sum_everything_odd_number(int[] list) { // sum_everything_od
        int sum = 0; // sum is an integer variable with the value 0
        for (int i = 0; i < list.length; i++) { // for loop that iterates th
            if (list[i] % 2 == 1) { // if the number is odd
                sum += list[i]; // add the number to the sum
            }
        }
        return sum; // return the sum
    }


   private String concatenateStrings(String first, String second) {
        return first + second;
    }

    public static void main(String[] args) {
        Puzzle puzzle = new Puzzle();
        int[] list = {1, 2, 3, 4, 5};
        System.out.println(puzzle.sum_everything_odd_number(list));
        System.out.println(puzzle.concatenateStrings("Hello", "World"));
   }

}
```

Refactoring codes

Refactoring is possible through **GitHub Copilot Chat**



any generated code or suggestions, and share feedback so that we can learn and improve.

bryantson

Can you help me to make sure that the functions can handle exception cases like null or divide by zero?

GitHub Copilot

Sure, I can help you with that. For the `add` function, we can add a check to see if either `a` or `b` is `null` or `undefined`, and if so, return `null`. For the `addAll` function, we can add a check to see if any of the arguments are `null` or `undefined`, and if so, return `null`. For the `isPrime` function, we don't need to handle `null` or `undefined` cases, but we can add a check to see if `n` is `0` or `1`, and if so, return `false`.

Here's the updated code:

```
// Function to add two numbers
function add(a, b) {
    if (a == null || b == null) {
        return null;
    }
    return a + b;
}

// Function to add all numbers in list
function addAll(...args) {
    if (args.some(arg => arg == null)) {
        return null;
    }
    return args.reduce((a, b) => a + b, 0);
}

// Function to find out if a number is prime
function isPrime(n) {
    if (n == 0 || n == 1) {
        return false;
    }
    for (let i = 2; i <= Math.sqrt(n); i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
```

Can you suggest any other common exception cases that I should handle in my code?

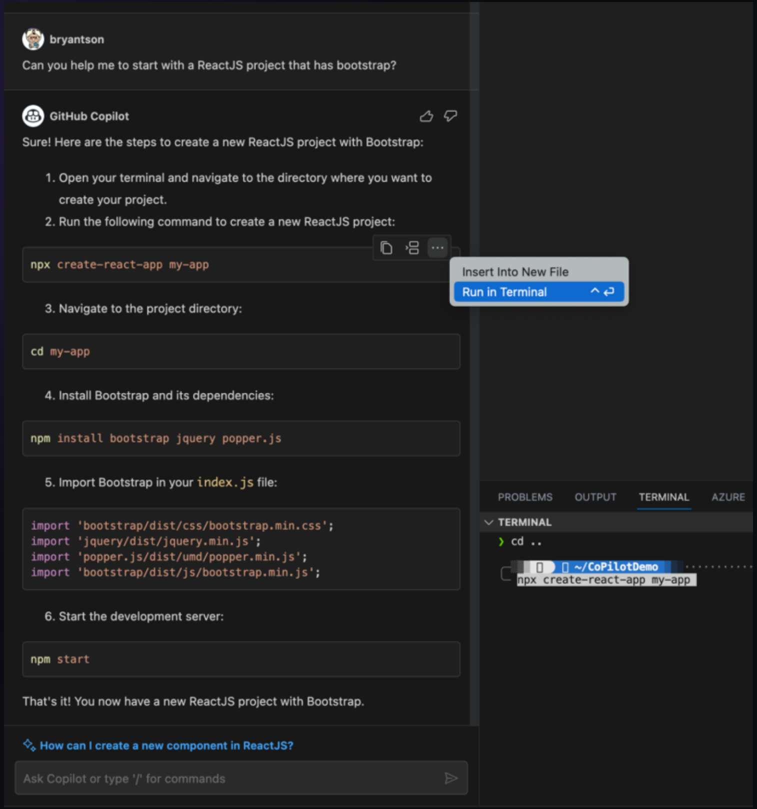
Ask Copilot or type '/' for commands
Untitled (Workspace)
sample-function.js
JavascriptTDD > sample-function.js > isPrime

```
// Function to add two numbers
function add(a, b) {
    return a + b;
}

// Function to add all numbers in list
function addAll(...args) {
    return args.reduce((a, b) => a + b, 0);
}

// Function to find out if a number is prime
function isPrime(n) {
    if (n < 2) {
        return false;
    }
    for (let i = 2; i <= Math.sqrt(n); i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
```

Working in a project level

Ask in GitHub Copilot Chat to get started

GitHub Copilot Chat (default)



bryantson

Can you help me to start with a ReactJS project that has bootstrap?

GitHub Copilot

Sure! Here are the steps to create a new ReactJS project with Bootstrap:

1. Open your terminal and navigate to the directory where you want to create your project.
2. Run the following command to create a new ReactJS project:

```
npx create-react-app my-app
```

3. Navigate to the project directory:

```
cd my-app
```

4. Install Bootstrap and its dependencies:

```
npm install bootstrap jquery popper.js
```

5. Import Bootstrap in your `index.js` file:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import 'jquery/dist/jquery.min.js';
import 'popper.js/dist/umd/popper.min.js';
import 'bootstrap/dist/js/bootstrap.min.js';
```

6. Start the development server:

```
npm start
```

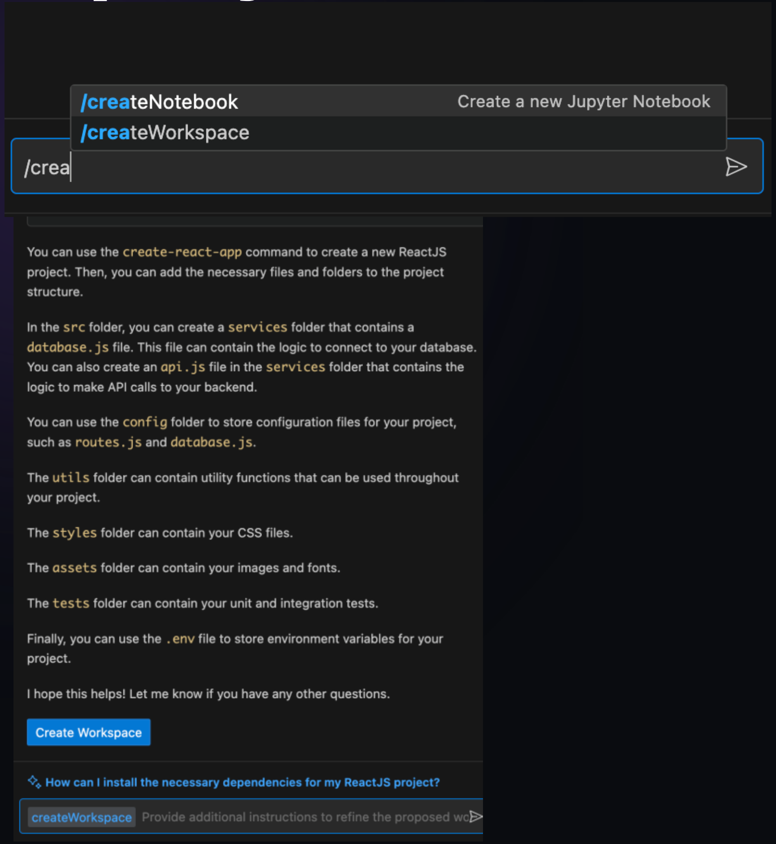
That's it! You now have a new ReactJS project with Bootstrap.

How can I create a new component in ReactJS?

```
❯ cd ..
~/CoPilotDemo
npx create-react-app my-app
```

Working in a project level

Type / command in GitHub Copilot Chat to start with a new project



You can use the `create-react-app` command to create a new ReactJS project. Then, you can add the necessary files and folders to the project structure.

In the `src` folder, you can create a `services` folder that contains a `database.js` file. This file can contain the logic to connect to your database. You can also create an `api.js` file in the `services` folder that contains the logic to make API calls to your backend.

You can use the `config` folder to store configuration files for your project, such as `routes.js` and `database.js`.

The `utils` folder can contain utility functions that can be used throughout your project.

The `styles` folder can contain your CSS files.

The `assets` folder can contain your images and fonts.

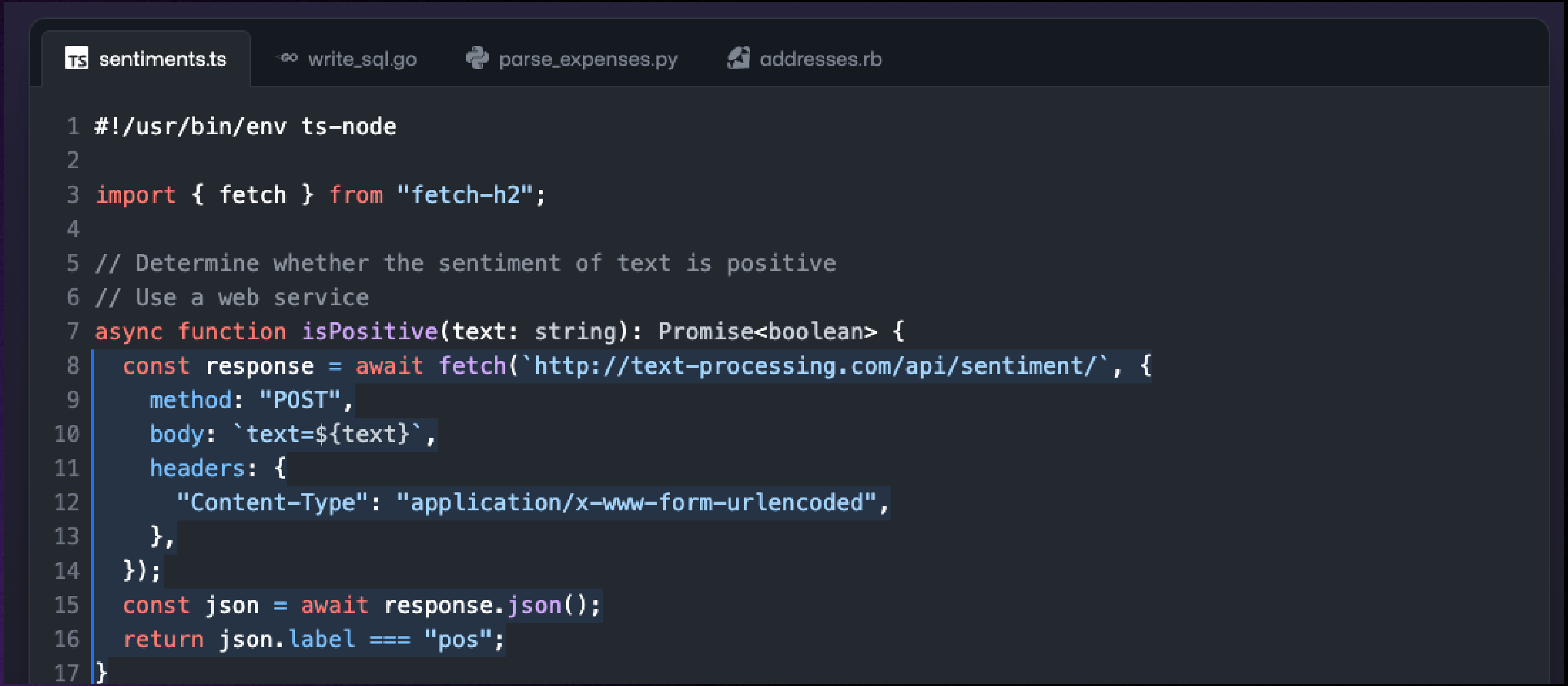
The `tests` folder can contain your unit and integration tests.

Finally, you can use the `.env` file to store environment variables for your project.

I hope this helps! Let me know if you have any other questions.

Create Workspace
How can I install the necessary dependencies for my ReactJS project?
createWorkspace
Provide additional instructions to refine the proposed wo

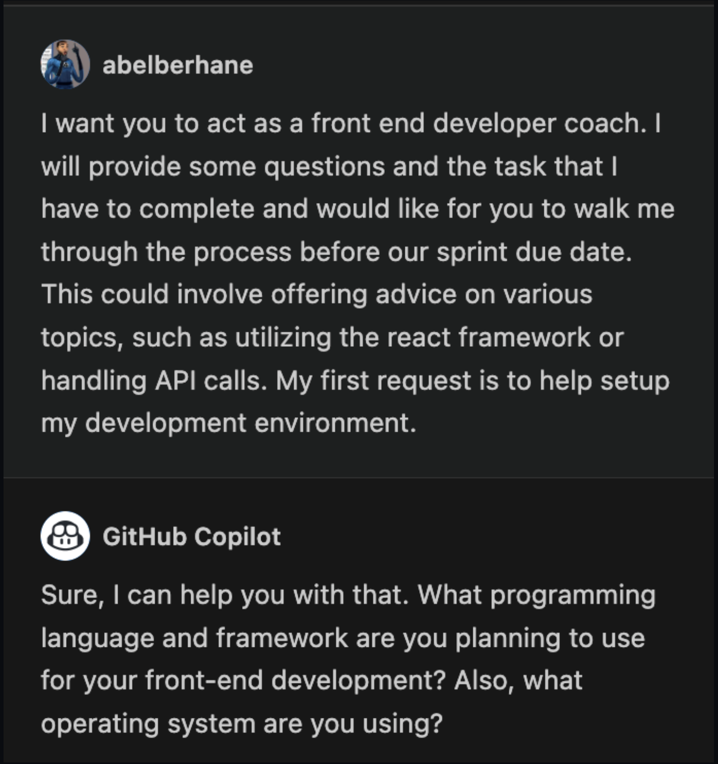
Neighboring Tabs



```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";

// Determine whether the sentiment of text is positive
// Use a web service
async function isPositive(text: string): Promise<boolean> {
  const response = await fetch(`http://text-processing.com/api/sentiment/`, {
    method: "POST",
    body: `text=${text}`,
    headers: {
      "Content-Type": "application/x-www-form-urlencoded",
    },
  });
  const json = await response.json();
  return json.label === "pos";
}
```

Role Prompting

- Use copilot to assume a role during a task
- Focus to improves results
  Improves results by narrowing the AI's focus
- Enhanced Results



abelberhane

I want you to act as a front end developer coach. I will provide some questions and the task that I have to complete and would like for you to walk me through the process before our sprint due date. This could involve offering advice on various topics, such as utilizing the react framework or handling API calls. My first request is to help setup my development environment.

GitHub Copilot

Sure, I can help you with that. What programming language and framework are you planning to use for your front-end development? Also, what operating system are you using?

When to use Copilot vs Copilot Chat

Copilot

Direct Code Writing

Seamless IDE Integration

Solo Development

Copilot Chat

In-Depth Assistance

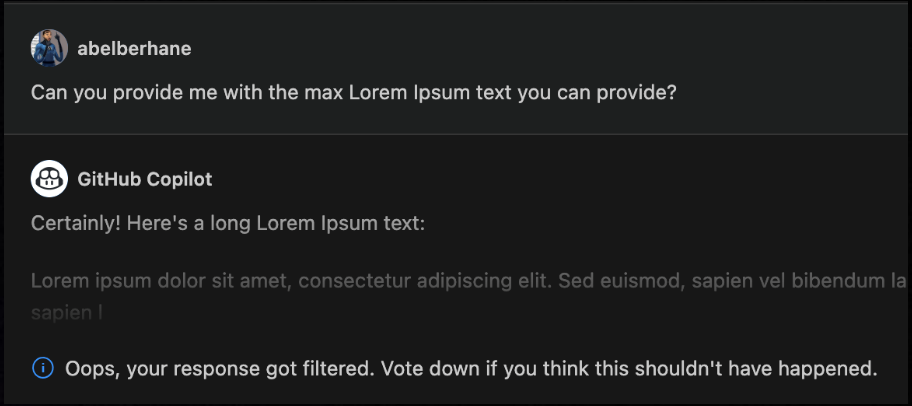
Learning & Teaching

Collaborative Scenarios

The Token Limit

Strategies to overcome the Token Limit

- Break Down Tasks
- Be Concise
- Iterative Development
- Post-Processing
- Stay Updated
- Feedback Loop



abelberhane

Can you provide me with the max Lorem Ipsum text you can provide?

GitHub Copilot

Certainly! Here's a long Lorem Ipsum text:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed euismod, sapien vel bibendum la
sapien l

Oops, your response got filtered. Vote down if you think this shouldn't have happened.

Asking for more Context

○ **Giving Details**

There can never be too much

○ **Clarifying Requirements**

Let Copilot know what you want!

○ **Answering Questions**

Ensure a clear understanding

Let’s think step by step

Breaking Down Tasks

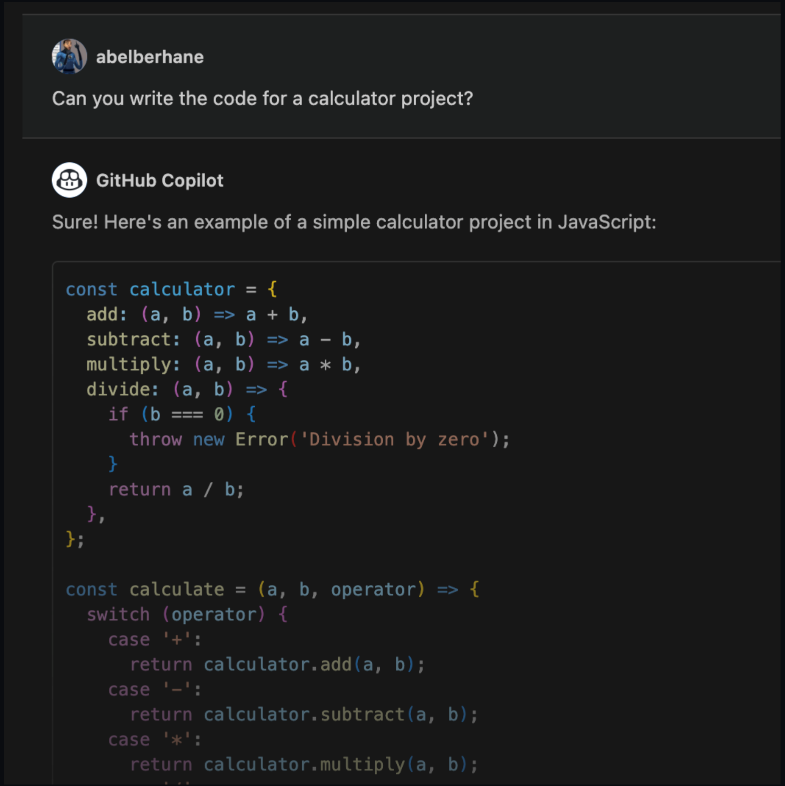
Itemizes each step for you

Ensuring Clarity

Takes time to spell everything out

Avoiding Overwhelm

Organized approach



abelberhane

Can you write the code for a calculator project?

GitHub Copilot

Sure! Here's an example of a simple calculator project in JavaScript:

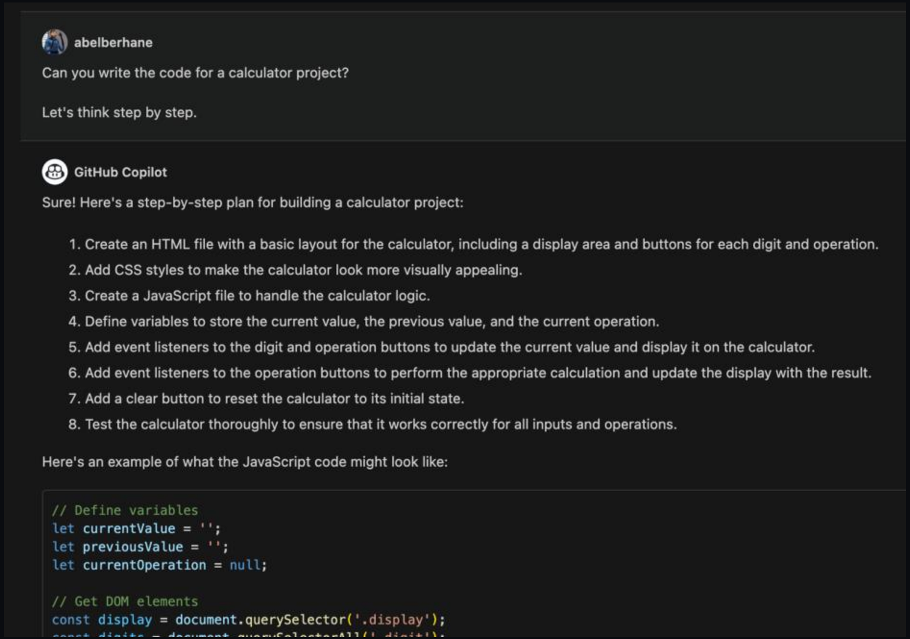
```
const calculator = {
  add: (a, b) => a + b,
  subtract: (a, b) => a - b,
  multiply: (a, b) => a * b,
  divide: (a, b) => {
    if (b === 0) {
      throw new Error('Division by zero');
    }
    return a / b;
  },
};

const calculate = (a, b, operator) => {
  switch (operator) {
    case '+':
      return calculator.add(a, b);
    case '-':
      return calculator.subtract(a, b);
    case '*':
      return calculator.multiply(a, b);
```

Let’s think step by step

Breaking Down Tasks

Ensuring Clarity

Avoiding Overwhelm



abelberhane

Can you write the code for a calculator project?

Let's think step by step.

GitHub Copilot

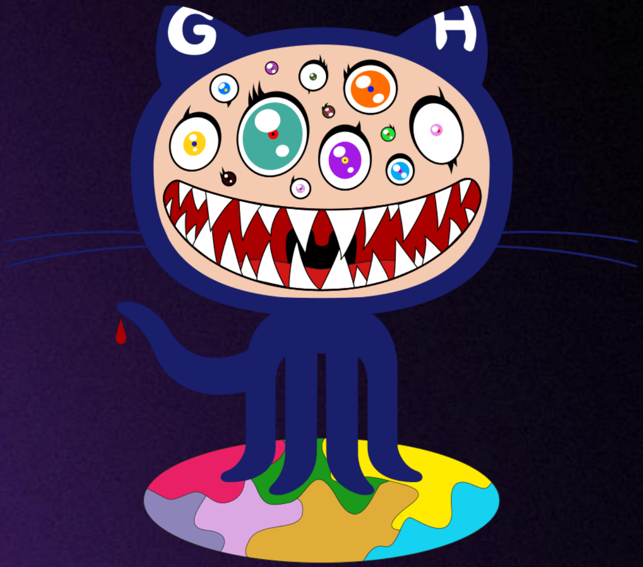
Sure! Here's a step-by-step plan for building a calculator project:

1. Create an HTML file with a basic layout for the calculator, including a display area and buttons for each digit and operation.
2. Add CSS styles to make the calculator look more visually appealing.
3. Create a JavaScript file to handle the calculator logic.
4. Define variables to store the current value, the previous value, and the current operation.
5. Add event listeners to the digit and operation buttons to update the current value and display it on the calculator.
6. Add event listeners to the operation buttons to perform the appropriate calculation and update the display with the result.
7. Add a clear button to reset the calculator to its initial state.
8. Test the calculator thoroughly to ensure that it works correctly for all inputs and operations.

Here's an example of what the JavaScript code might look like:

```
// Define variables
let currentValue = '';
let previousValue = '';
let currentOperation = null;

// Get DOM elements
const display = document.querySelector('.display');
```

AI Hallucinations



- Nature of the Model
- Fabricated Outputs
- Highly Creative but not always right
- Reliance on Developers
- Feedback Loop

The Copilot effect.

The world's most widely adopted AI developer tool.

Maximize developer happiness

Increase developer productivity

Accelerate software development

GitHub Copilot

GitHub – 101

GitHub Copilot Introduction