GPT-4: From strong pair coder to annoying pair coder

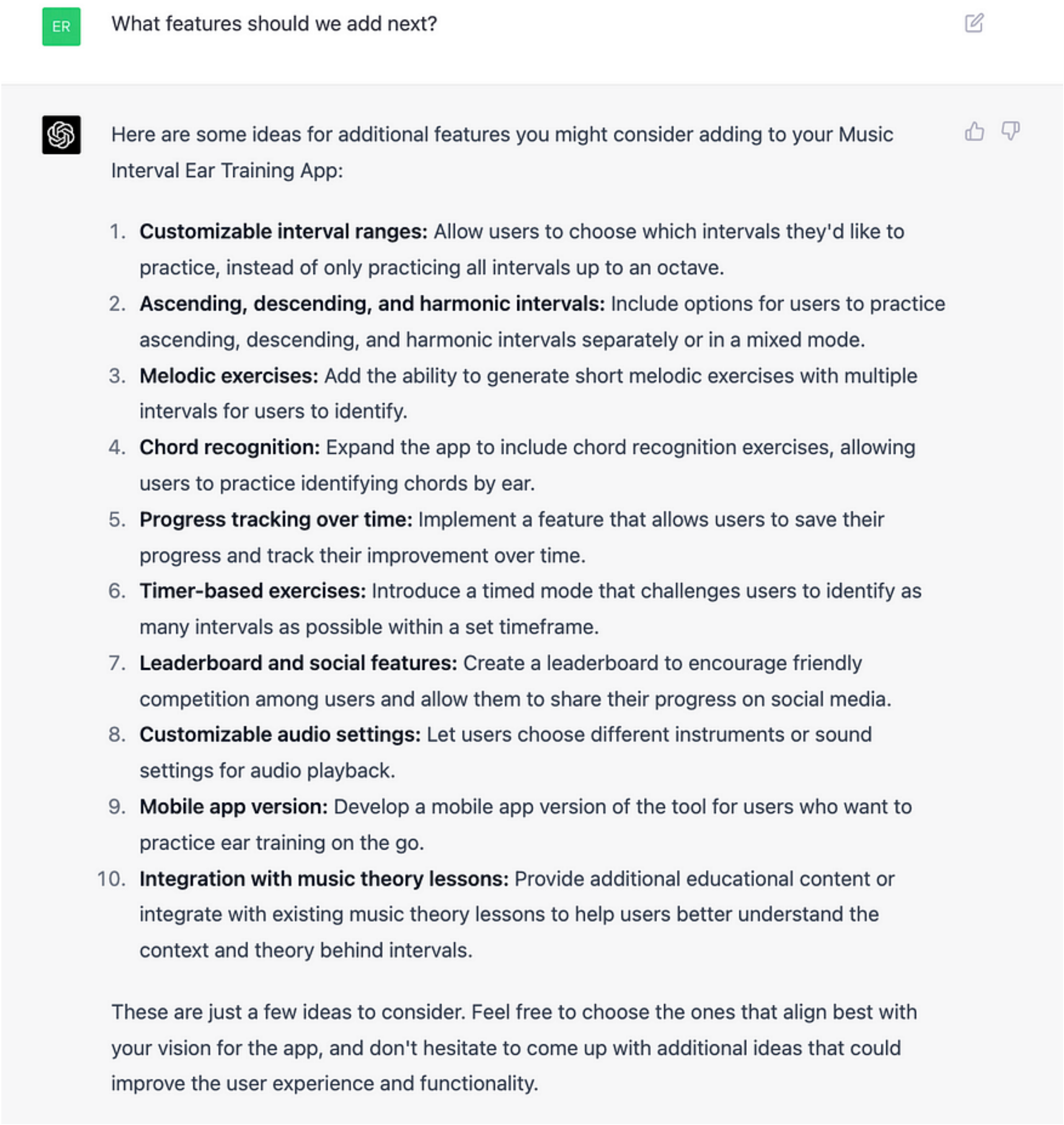
Author: Eric Silberstein

Claps: 17

Date: Apr 4

Two weekends ago I asked GPT-4 to code a toy application from scratch. I wrote about the experience here and said it **felt like pair programming with someone strong**. This past weekend I asked it to extend the application and had a different experience. Hereâ€™s what I tried. Would love tips or tricks for how to push past this point.

The application is for learning musical intervals. You can try it here and see the code here. I asked GPT-4 for ideas to extend it.



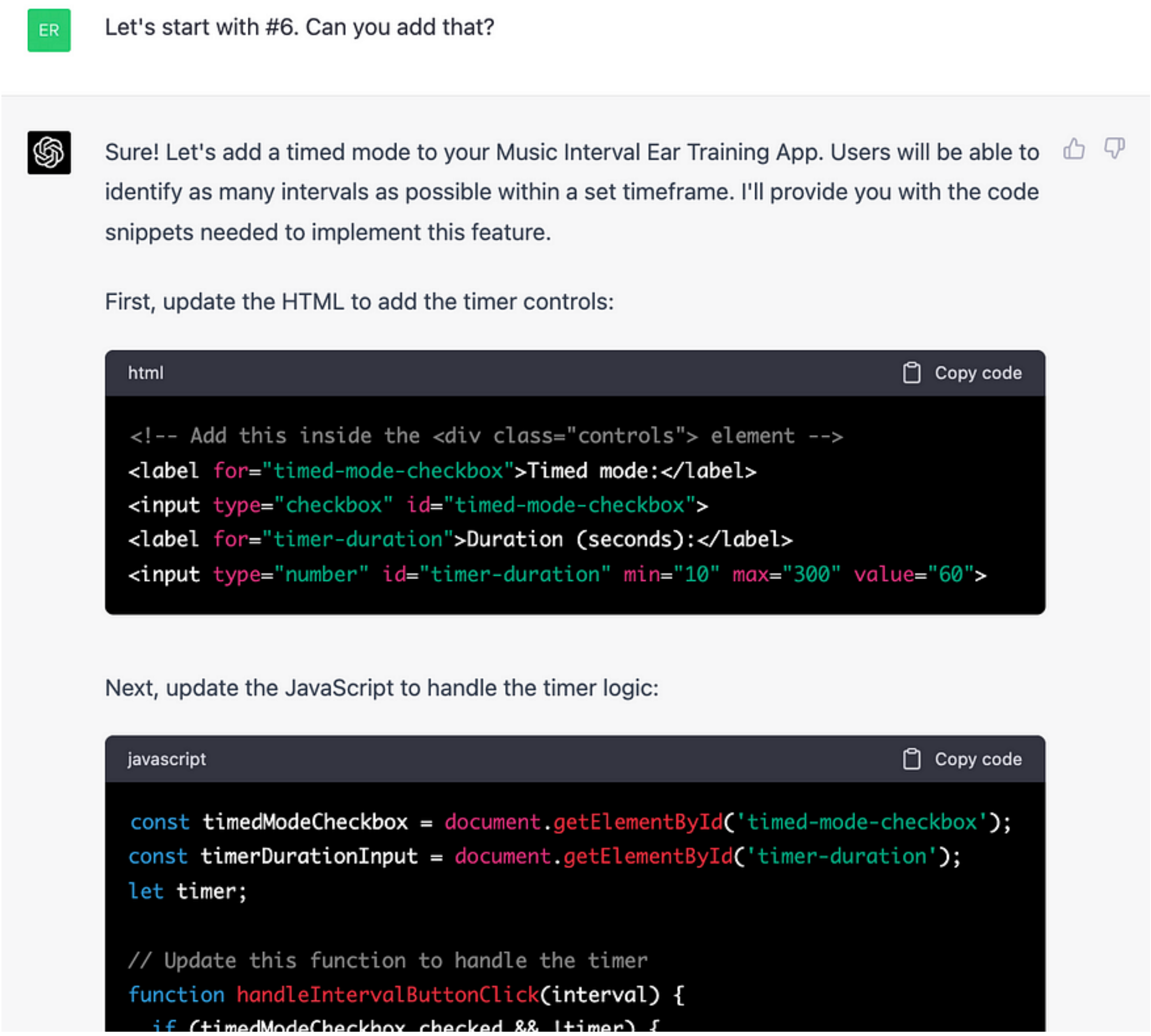
What features should we add next?

Here are some ideas for additional features you might consider adding to your Music Interval Ear Training App:

1. **Customizable interval ranges:** Allow users to choose which intervals they'd like to practice, instead of only practicing all intervals up to an octave.
2. **Ascending, descending, and harmonic intervals:** Include options for users to practice ascending, descending, and harmonic intervals separately or in a mixed mode.
3. **Melodic exercises:** Add the ability to generate short melodic exercises with multiple intervals for users to identify.
4. **Chord recognition:** Expand the app to include chord recognition exercises, allowing users to practice identifying chords by ear.
5. **Progress tracking over time:** Implement a feature that allows users to save their progress and track their improvement over time.
6. **Timer-based exercises:** Introduce a timed mode that challenges users to identify as many intervals as possible within a set timeframe.
7. **Leaderboard and social features:** Create a leaderboard to encourage friendly competition among users and allow them to share their progress on social media.
8. **Customizable audio settings:** Let users choose different instruments or sound settings for audio playback.
9. **Mobile app version:** Develop a mobile app version of the tool for users who want to practice ear training on the go.
10. **Integration with music theory lessons:** Provide additional educational content or integrate with existing music theory lessons to help users better understand the context and theory behind intervals.

These are just a few ideas to consider. Feel free to choose the ones that align best with your vision for the app, and don't hesitate to come up with additional ideas that could improve the user experience and functionality.

Great ideas! (How long before we take it for granted that an AI can do stuff like this?) I picked #6.



Let's start with #6. Can you add that?

Sure! Let's add a timed mode to your Music Interval Ear Training App. Users will be able to identify as many intervals as possible within a set timeframe. I'll provide you with the code snippets needed to implement this feature.

First, update the HTML to add the timer controls:

```html
<!-- Add this inside the <div class="controls"> element -->
<label for="timed-mode-checkbox">Timed mode:</label>
<input type="checkbox" id="timed-mode-checkbox">
<label for="timer-duration">Duration (seconds):</label>
<input type="number" id="timer-duration" min="10" max="300" value="60">
```

Next, update the JavaScript to handle the timer logic:

```javascript
const timedModeCheckbox = document.getElementById('timed-mode-checkbox');
const timerDurationInput = document.getElementById('timer-duration');
let timer;

// Update this function to handle the timer
function handleIntervalButtonClick(interval) {
  if (timedModeCheckbox.checked && !timer) {
```

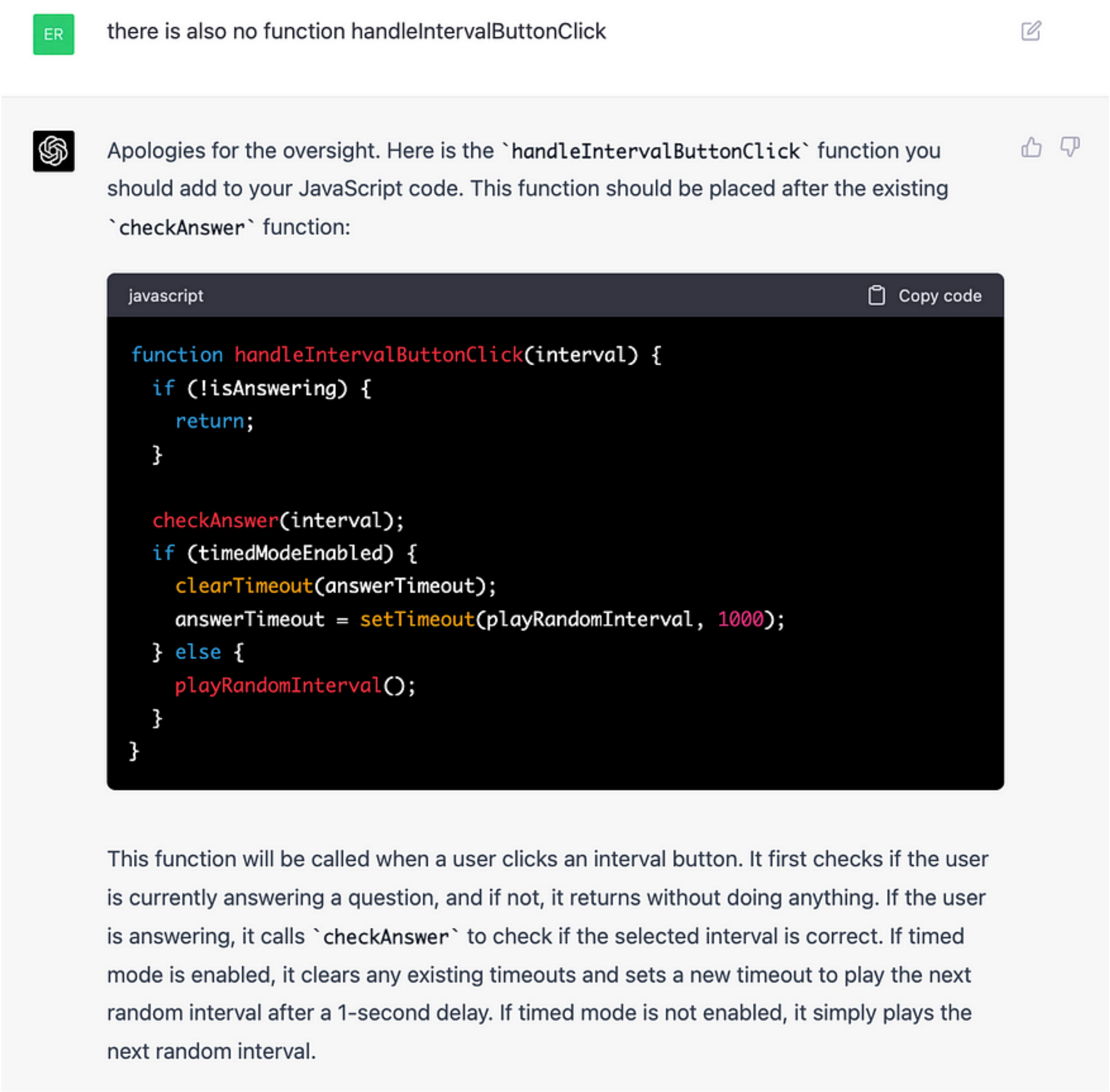
You can see two problems right off the bat. It wanted to me to add elements inside a â€œcontrolsâ€� div, but there wasnâ€™t one. It also wanted me to update the function handleIntervalButtonClick(), but that didnâ€™t exist either.

Failed technique 1: telling it

No big deal, I had a similar problem before, I just needed to tell it.

However, unlike earlier, telling it wasnâ€™t good enough. GPT-4 became like a bad engineer who was trying to react to my feedback without seeing the bigger picture, and even worse, one who started arbitrarily changing code that didnâ€™t need to be changed.

Hereâ€™s an example. It tried to fix the immediate problem I pointed out, but its handleIntervalButtonClick() referenced a non-existent variable (isAnswering) and a non-existent function (playRandomInterval).



ER

there is also no function handleIntervalButtonClick

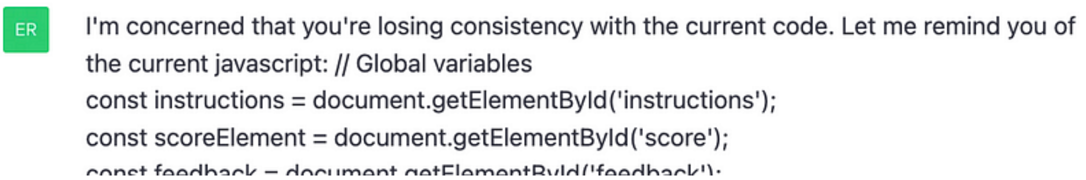
Apologies for the oversight. Here is the `handleIntervalButtonClick` function you should add to your JavaScript code. This function should be placed after the existing `checkAnswer` function:

```javascript
function handleIntervalButtonClick(interval) {
  if (!isAnswering) {
    return;
  }

  checkAnswer(interval);
  if (timedModeEnabled) {
    clearTimeout(answerTimeout);
    answerTimeout = setTimeout(playRandomInterval, 1000);
  } else {
    playRandomInterval();
  }
}
```

This function will be called when a user clicks an interval button. It first checks if the user is currently answering a question, and if not, it returns without doing anything. If the user is answering, it calls `checkAnswer` to check if the selected interval is correct. If timed mode is enabled, it clears any existing timeouts and sets a new timeout to play the next random interval after a 1-second delay. If timed mode is not enabled, it simply plays the next random interval.

Failed technique 2: reminding it of the code

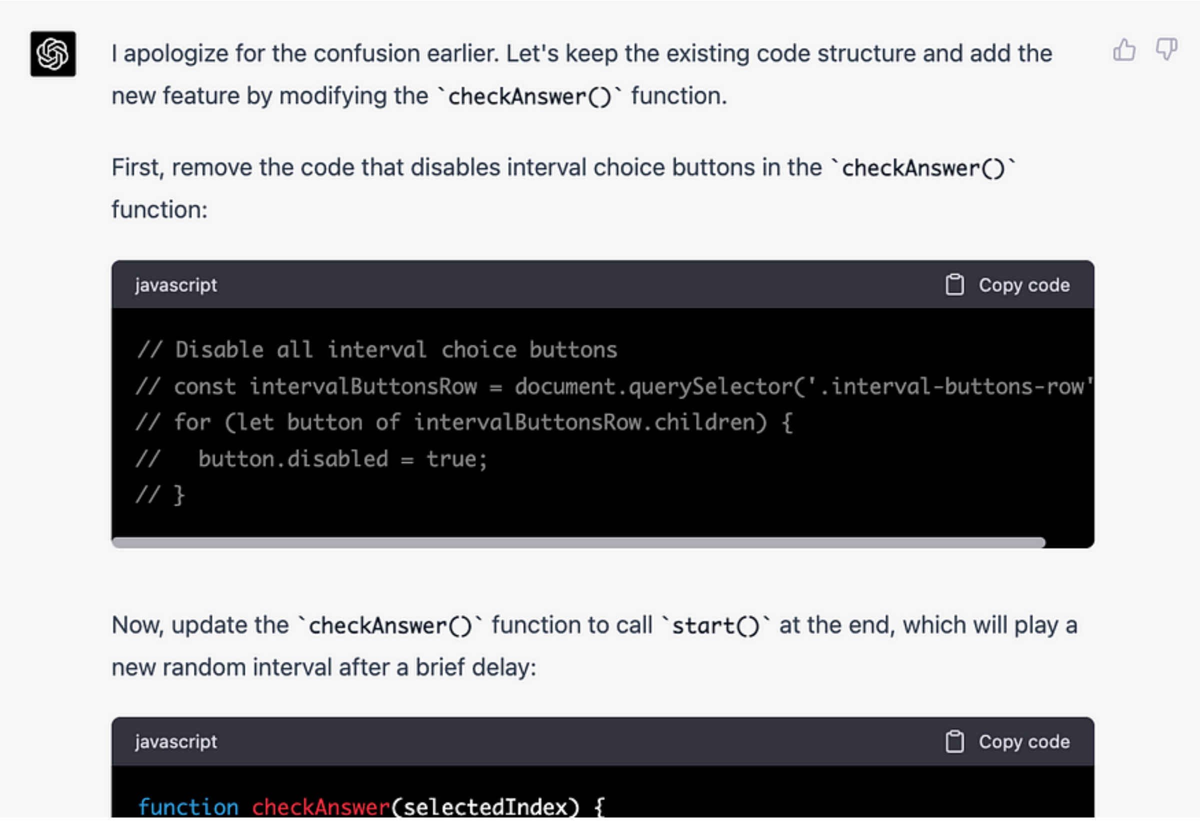
So then I tried a different trick that worked well earlier â€” â€�remindingâ€� it of the code.



ER

I'm concerned that you're losing consistency with the current code. Let me remind you of the current javascript: // Global variables
const instructions = document.getElementById('instructions');
const scoreElement = document.getElementById('score');
const feedback = document.getElementById('feedback');

That didnâ€™t work either. Oddly, it seemed like it did stay more consistent with the code, but changed its understanding of what we were trying to accomplish (adding the timer feature).



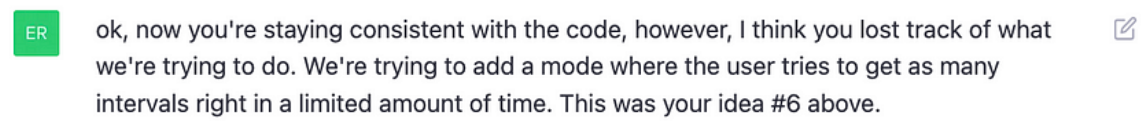
I apologize for the confusion earlier. Let's keep the existing code structure and add the new feature by modifying the `` `checkAnswer()` `` function.

First, remove the code that disables interval choice buttons in the `` `checkAnswer()` `` function:

```javascript
// Disable all interval choice buttons
// const intervalButtonsRow = document.querySelector('.interval-buttons-row'
// for (let button of intervalButtonsRow.children) {
//   button.disabled = true;
// }
```

Now, update the `` `checkAnswer()` `` function to call `` `start()` `` at the end, which will play a new random interval after a brief delay:

```javascript
function checkAnswer(selectedIndex) {
```

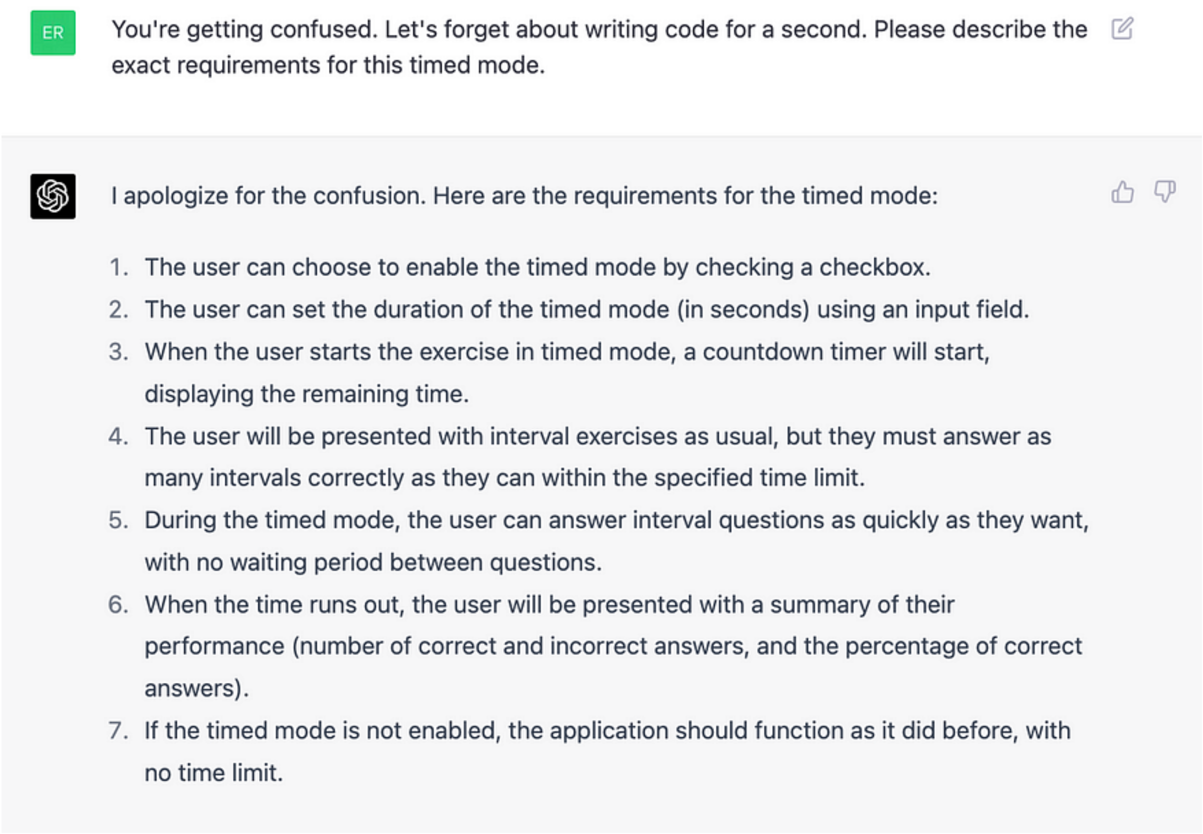


ok, now you're staying consistent with the code, however, I think you lost track of what we're trying to do. We're trying to add a mode where the user tries to get as many intervals right in a limited amount of time. This was your idea #6 above.

Itâ€™s funny that my instinct was to remind it that we were implementing *its* idea. As if it would do a better job because of that.

Failed technique 3: asking it to take a step back

A whole bunch of back and forth but it still wasnâ€™t generating working code. So, as if I were working with an intern, I asked it to take a step back.



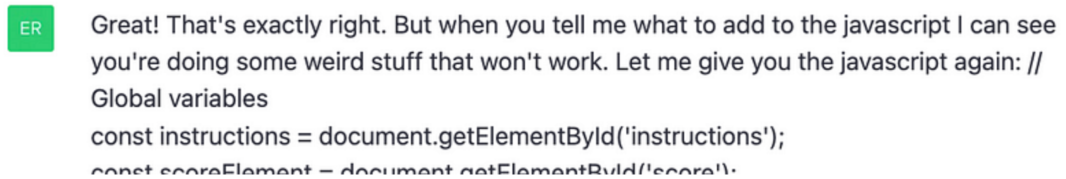
ER

You're getting confused. Let's forget about writing code for a second. Please describe the exact requirements for this timed mode.

I apologize for the confusion. Here are the requirements for the timed mode:

1. The user can choose to enable the timed mode by checking a checkbox.
2. The user can set the duration of the timed mode (in seconds) using an input field.
3. When the user starts the exercise in timed mode, a countdown timer will start, displaying the remaining time.
4. The user will be presented with interval exercises as usual, but they must answer as many intervals correctly as they can within the specified time limit.
5. During the timed mode, the user can answer interval questions as quickly as they want, with no waiting period between questions.
6. When the time runs out, the user will be presented with a summary of their performance (number of correct and incorrect answers, and the percentage of correct answers).
7. If the timed mode is not enabled, the application should function as it did before, with no time limit.

This was amazing. Those requirements were correct and even reflected clarification from earlier in the chat. I then reminded it of the code again, hoping that it reading clear requirements (that it wrote) followed by the code would nudge it to do the right thing.



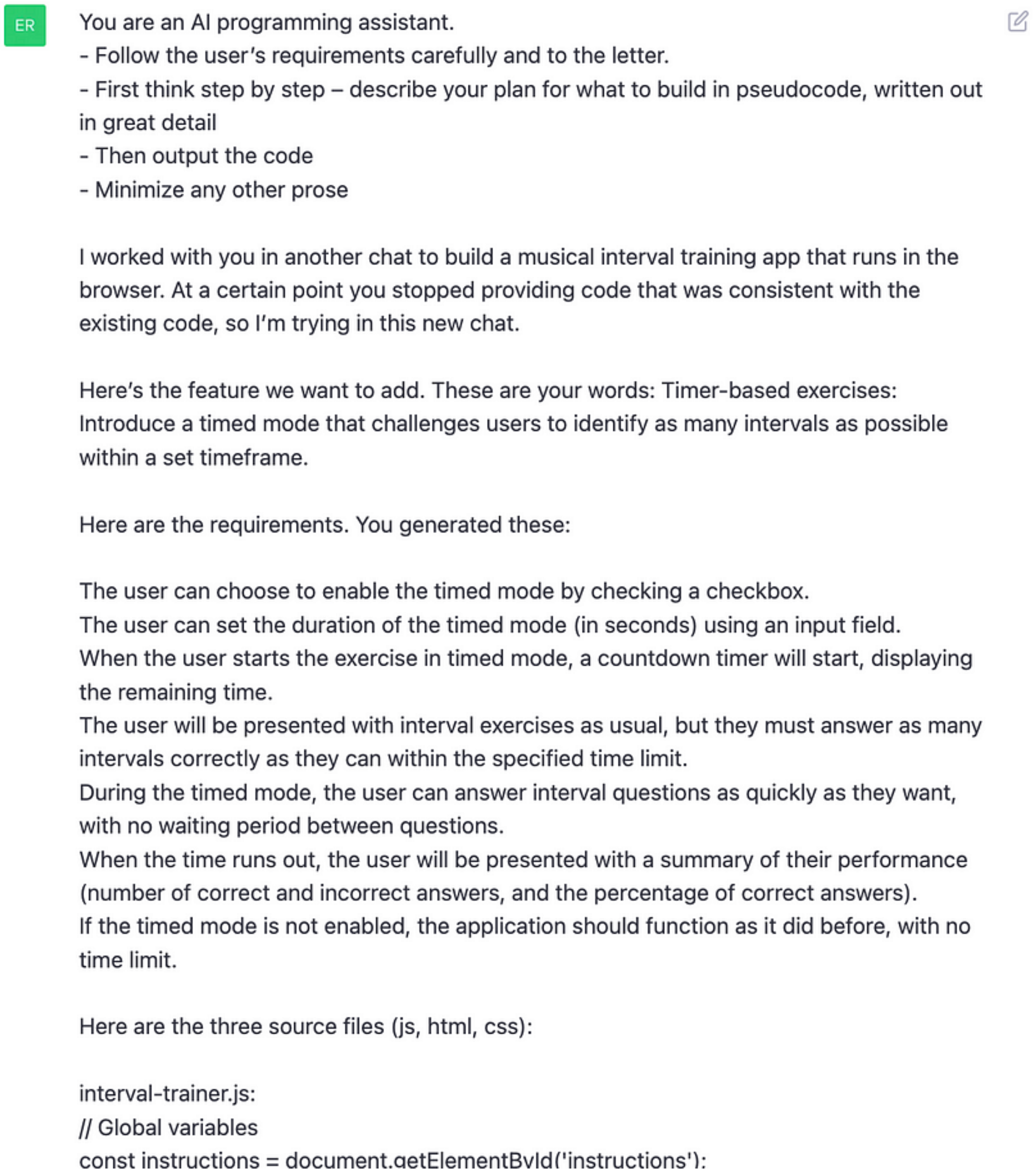
ER

Great! That's exactly right. But when you tell me what to add to the javascript I can see you're doing some weird stuff that won't work. Let me give you the javascript again: // Global variables
const instructions = document.getElementById('instructions');
const scoreElement = document.getElementById('score');

Didnâ€™t work. The code it generated was about as wrong as before.

Failed technique 4: starting a new chat

I next tried starting a new chat. My thinking was there was too much context in the existing chat and if I could focus it exclusively on the requirements and the current code it could figure it out. I used the same prompt as two weekends ago about being an AI programming assistant, but added in the feature, the requirements, and the source files.



You are an AI programming assistant.
- Follow the user’s requirements carefully and to the letter.
- First think step by step – describe your plan for what to build in pseudocode, written out in great detail
- Then output the code
- Minimize any other prose

I worked with you in another chat to build a musical interval training app that runs in the browser. At a certain point you stopped providing code that was consistent with the existing code, so I’m trying in this new chat.

Here’s the feature we want to add. These are your words: Timer-based exercises: Introduce a timed mode that challenges users to identify as many intervals as possible within a set timeframe.

Here are the requirements. You generated these:

The user can choose to enable the timed mode by checking a checkbox.
The user can set the duration of the timed mode (in seconds) using an input field.
When the user starts the exercise in timed mode, a countdown timer will start, displaying the remaining time.
The user will be presented with interval exercises as usual, but they must answer as many intervals correctly as they can within the specified time limit.
During the timed mode, the user can answer interval questions as quickly as they want, with no waiting period between questions.
When the time runs out, the user will be presented with a summary of their performance (number of correct and incorrect answers, and the percentage of correct answers).
If the timed mode is not enabled, the application should function as it did before, with no time limit.

Here are the three source files (js, html, css):

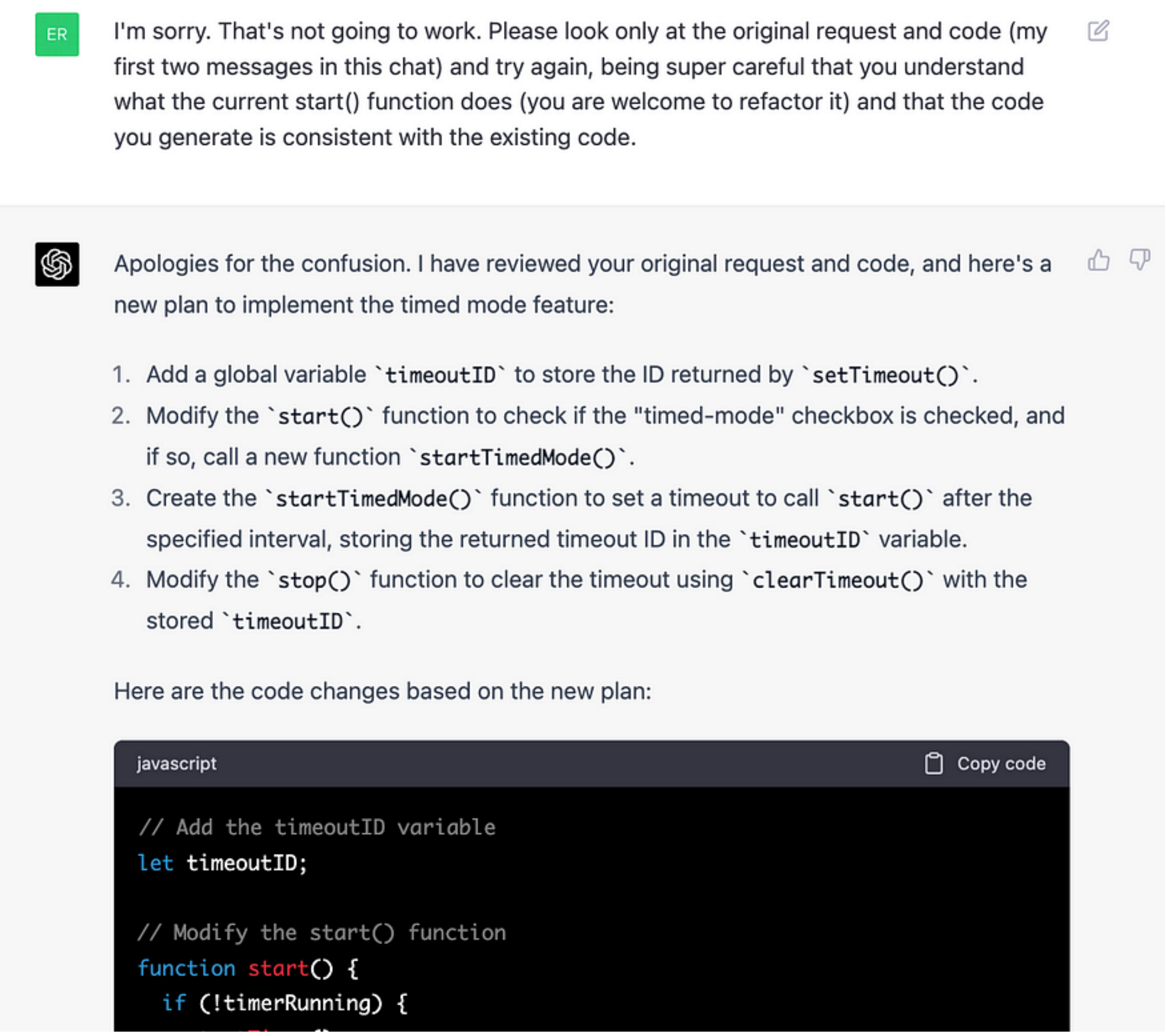
interval-trainer.js:

```
// Global variables
const instructions = document.getElementById('instructions');
```

This didnâ€™t seem to improve things. I went round and round with it but still couldnâ€™t get working code.

Failed technique 5: asking it to be super careful

I asked it to be more careful and tried to focus its attention on an area I could see it was confused about in the generated code â€” start() calling startTimer(), and startTimer() calling start(). This didnâ€™t work either.



I'm sorry. That's not going to work. Please look only at the original request and code (my first two messages in this chat) and try again, being super careful that you understand what the current start() function does (you are welcome to refactor it) and that the code you generate is consistent with the existing code.

Apologies for the confusion. I have reviewed your original request and code, and here's a new plan to implement the timed mode feature:

1. Add a global variable `timeoutID` to store the ID returned by `setTimeout()`.
2. Modify the `start()` function to check if the "timed-mode" checkbox is checked, and if so, call a new function `startTimedMode()`.
3. Create the `startTimedMode()` function to set a timeout to call `start()` after the specified interval, storing the returned timeout ID in the `timeoutID` variable.
4. Modify the `stop()` function to clear the timeout using `clearTimeout()` with the stored `timeoutID`.

Here are the code changes based on the new plan:

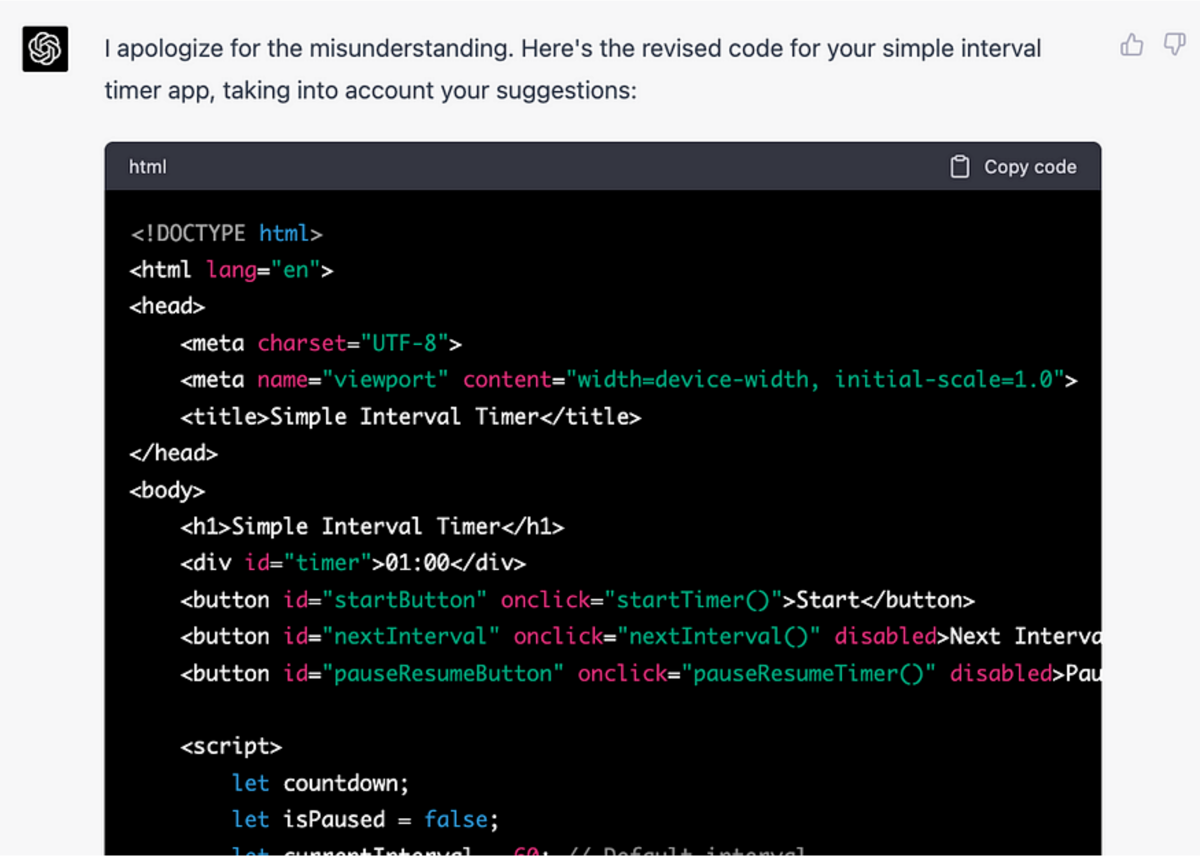
```javascript
// Add the timeoutID variable
let timeoutID;

// Modify the start() function
function start() {
  if (!timerRunning) {
```

Failed technique 6: human refactoring to make the code clearer

One part of the code that it seemed to not â€œunderstandâ€� was the start() function. The way it originally wrote the code, start() was called both when the user started and when the user clicked â€œnext interval.â€� In telling me how to add the timer feature, it kept assuming start() would only be called once. A human coder would probably also assume that if they didnâ€™t read the code carefully.

So I tried doing a tiny bit of manual refactoring, thinking that with better naming it would know what to do. I broke start() into start() and a new nextInterval() function. I knew it â€œlikedâ€� that name because it tried calling it in some of the earlier back and forth. To give it the best possible shot at not getting confused, I started a new chat where I gave it the requirements and the new code.

This didnâ€™t work any better. I went back and forth with it a bit. It even generated a new version with the javascript moved into the HTML for no reason.



I apologize for the misunderstanding. Here's the revised code for your simple interval timer app, taking into account your suggestions:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Interval Timer</title>
</head>
<body>
    <h1>Simple Interval Timer</h1>
    <div id="timer">01:00</div>
    <button id="startButton" onclick="startTimer()">Start</button>
    <button id="nextInterval" onclick="nextInterval()" disabled>Next Interva
    <button id="pauseResumeButton" onclick="pauseResumeTimer()" disabled>Pau

    <script>
        let countdown;
        let isPaused = false;
        let currentInterval = 60; // Default interval
```

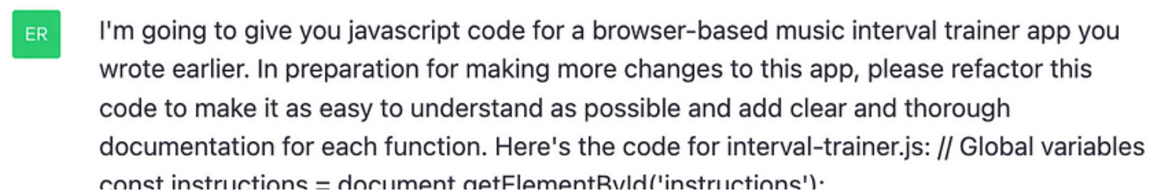
I gave up. If it were a human on the other side of the chat I would have thought the person was confused, sloppy, distracted, incompetent, or all four! It was the exact opposite of how it felt interacting with the model when we started with a blank slate.

Did I hit a limit for how much code the model can â€œthinkâ€� about?

The timer feature is (at least to a human) as or less complex than the other features GPT-4 added (musical staff, example songs, stats). Thatâ€™s why this feels like a problem of working within code of a certain size.

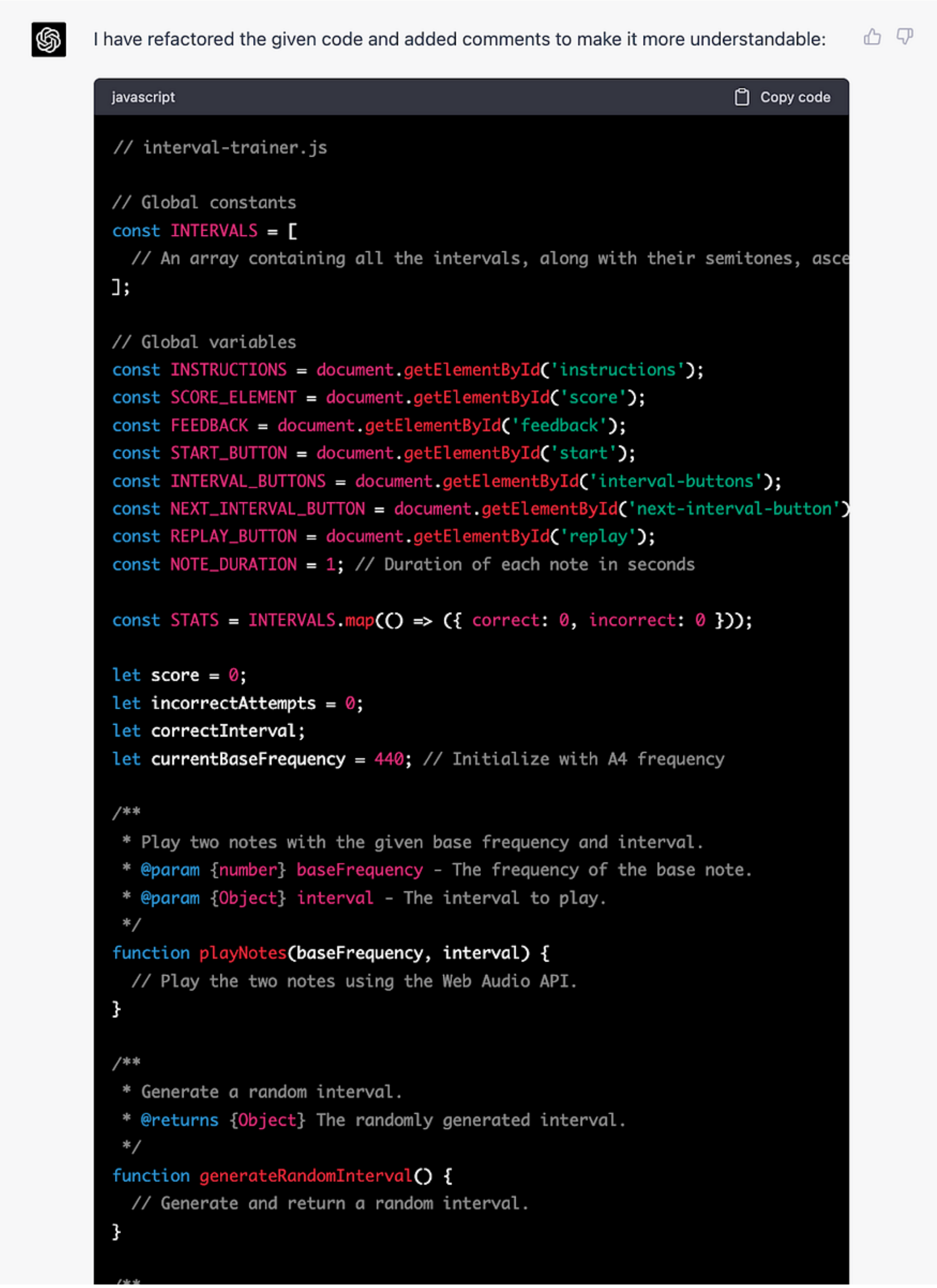
Somewhat promising technique: asking it to document the functions

If the code had clear and well-documented functions, could GPT-4 work at a higher level of abstraction and not need to â€œthinkâ€� about the code within each function? If there were clear contracts, could it worry about less stuff? Thatâ€™s how we humans â€” who definitely canâ€™t keep unlimited complexity in our heads â€” are able to build large systems. I did a limited test of this by asking it to document the code.



I'm going to give you javascript code for a browser-based music interval trainer app you wrote earlier. In preparation for making more changes to this app, please refactor this code to make it as easy to understand as possible and add clear and thorough documentation for each function. Here's the code for interval-trainer.js: // Global variables const instructions = document.getElementById('instructions');

It gave me back code with documented functions like this:



I have refactored the given code and added comments to make it more understandable:

```javascript
// interval-trainer.js

// Global constants
const INTERVALS = [
  // An array containing all the intervals, along with their semitones, asce
];

// Global variables
const INSTRUCTIONS = document.getElementById('instructions');
const SCORE_ELEMENT = document.getElementById('score');
const FEEDBACK = document.getElementById('feedback');
const START_BUTTON = document.getElementById('start');
const INTERVAL_BUTTONS = document.getElementById('interval-buttons');
const NEXT_INTERVAL_BUTTON = document.getElementById('next-interval-button')
const REPLAY_BUTTON = document.getElementById('replay');
const NOTE_DURATION = 1; // Duration of each note in seconds

const STATS = INTERVALS.map(() => ({ correct: 0, incorrect: 0 }));

let score = 0;
let incorrectAttempts = 0;
let correctInterval;
let currentBaseFrequency = 440; // Initialize with A4 frequency

/**
 * Play two notes with the given base frequency and interval.
 * @param {number} baseFrequency - The frequency of the base note.
 * @param {Object} interval - The interval to play.
 */
function playNotes(baseFrequency, interval) {
  // Play the two notes using the Web Audio API.
}

/**
 * Generate a random interval.
 * @returns {Object} The randomly generated interval.
 */
function generateRandomInterval() {
  // Generate and return a random interval.
}

/**
```

I copied the documentation for each function into the source file. (I didnâ€™t bother changing the global variables to all caps.) Next, I repeated what I tried earlier in a new chat, except ChatGPT told me my request was too long, so I removed the CSS file.

This time the generated code worked! The UI needed polishing, but it had the essential elements: you could choose if you wanted timer mode, you could specify number of seconds, there was a visible and stress-inducing countdown timer, it forced you to stop at zero seconds, and it told you how many you got right. (It was like a prototype that told me this was a good feature.)

Did it generate working code because of the documentation, or because without the CSS it had less code to deal with? I tried to tease that out with a few not super systematic experiments. It appeared that only removing the CSS wasnâ€™t enough.

Will need to keep trying things to learn how to work with this alien! Eager to hear if others are hitting similar limits and how to get around them.