WHITE PAPER

Codex-maxxing for long-running work

How Codex helps work continue beyond a single prompt

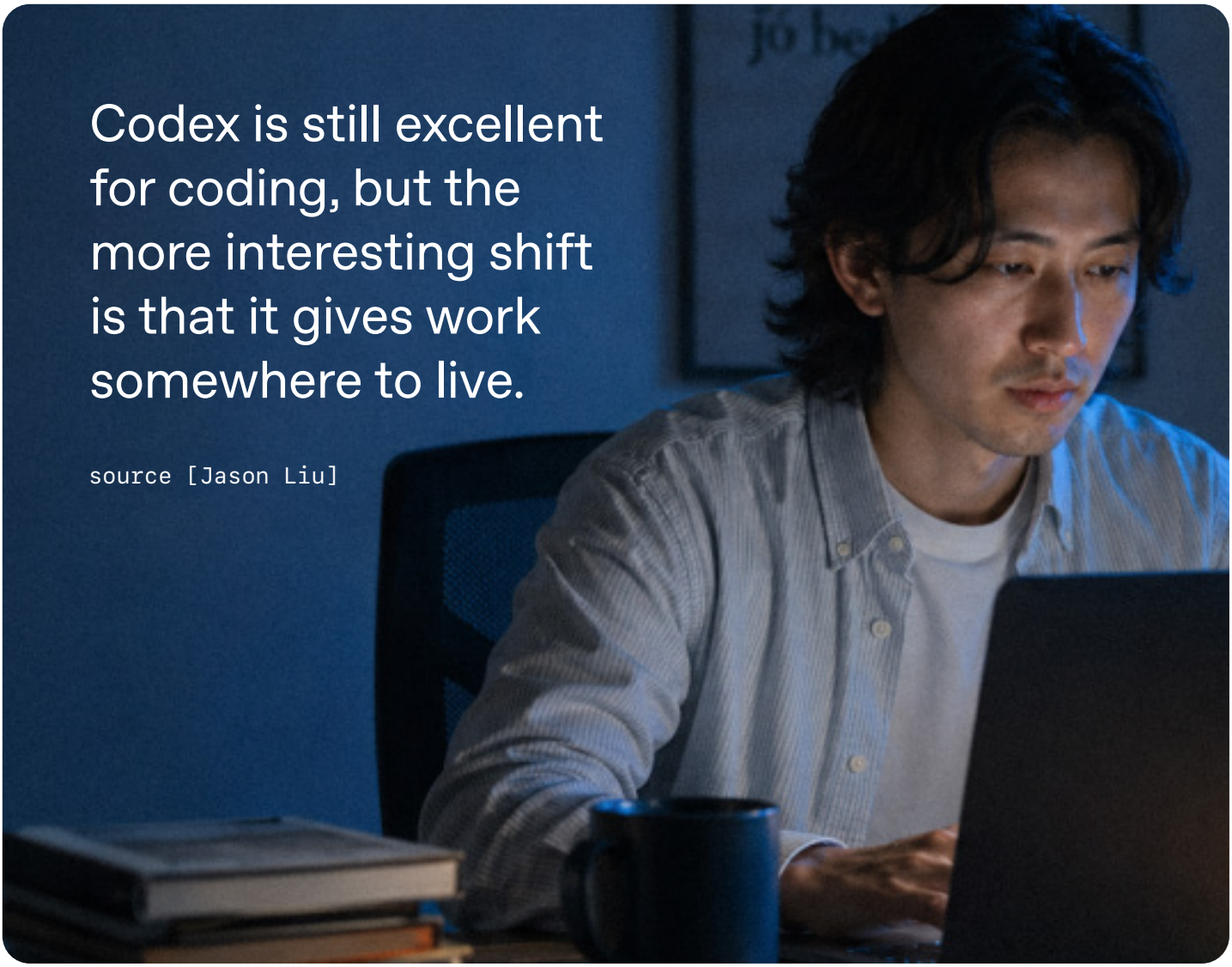
OpenAI

Contents

Codex is built for [coding work]: making diffs, changing repos, reviewing changes, and helping ship code.

But the broader pattern is starting to look different for everyone—not just developers.

When Codex has a durable thread, shared memory, access to tools, a way to recur, and a place to review the output itself, the work can keep moving across more than one prompt.



Creator [Jason Liu] uses Codex as part of his everyday workflow, from presentations and transcripts to spreadsheets, browser tasks, and recurring work.

The guide looks at how Jason uses Codex as a place to gather context, ask for help, review what comes back, and return to the next step without losing the thread.

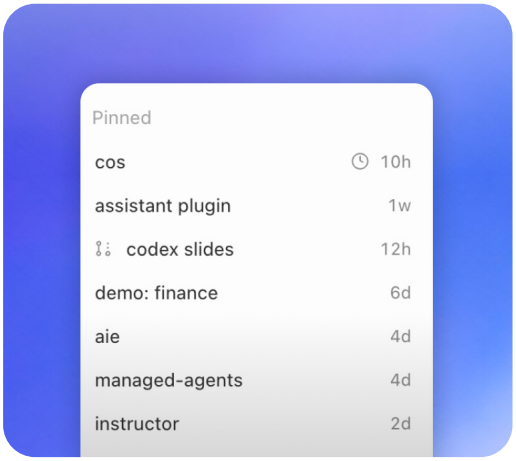
Durable threads

SECTION 01

Giving the work a place to live

For important workstreams, a pinned thread can become the home for the work: the place where context, preferences, old decisions, and open loops accumulate over time.

Use a [durable thread] for work that you expect to revisit:



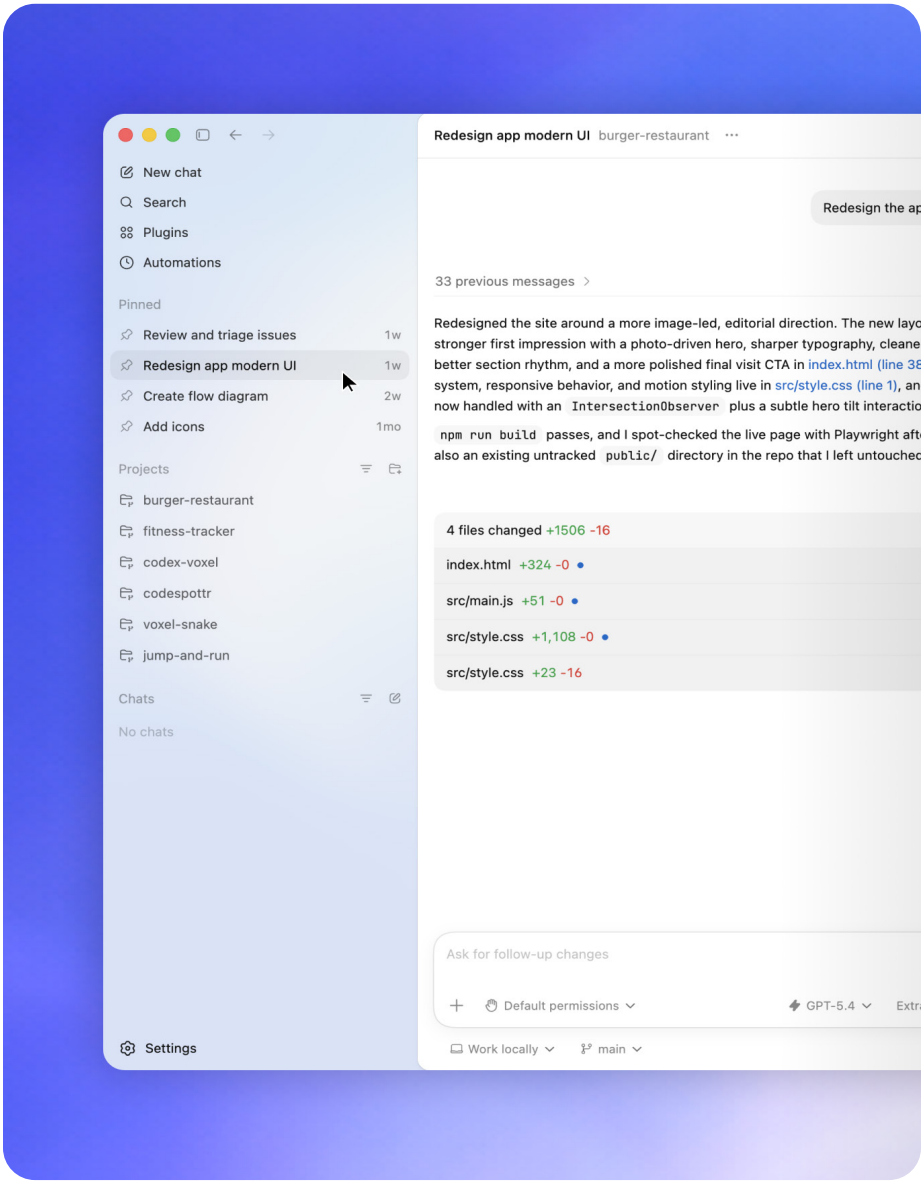
[Chief of Staff]
Messages, follow-ups, meetings, and open loops.

[OpenAI CLI]
Commands, conventions, repo structure, and review preferences.

[Social feedback monitoring]
Recurring questions, useful feedback, and signal from the public conversation.

[Agents SDK]
Examples, docs, recurring questions, and product behavior.

[Codex for open source]
Issues, maintainers, contribution patterns, and release notes.



> Compounding effect of [durable threads]

Durable threads keep history available as the conversation grows and stores them in the memory vault. But there's a tradeoff: long-running threads carry context and may cost more to run than a fresh short thread. For important workstreams, continuity can be worth it and easier to manage.

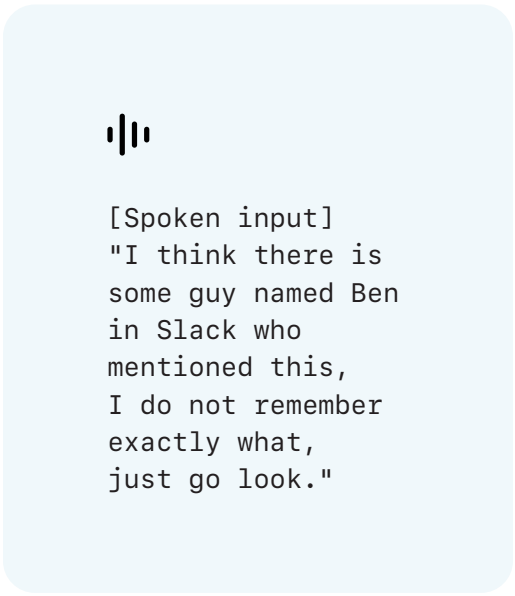
Voice input

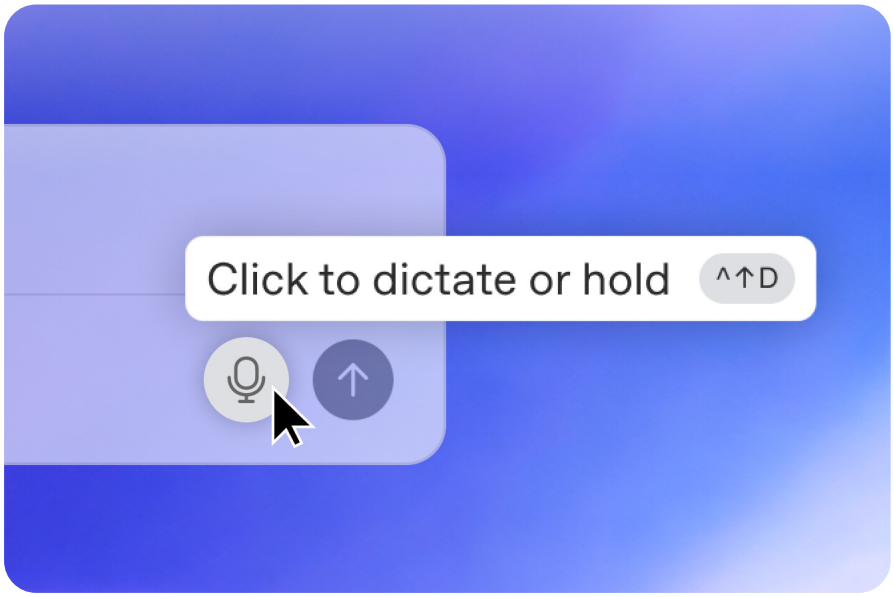
SECTION 02

Put more of your actual thinking into Codex

[Voice input] changes the kind of context Codex gets.

The benefit isn't only speed. It's that spoken input often includes the unedited version of the work: the half-remembered name, the loose direction, the uncertainty, the thing that would be awkward to type but natural to say.





That's a useful instruction because it's how real work often starts. Jason will start recording voice notes as he explores a page an agent made in the browser. When he's finished, he hits Enter so Codex can execute that feedback.

Transcripts work the same way. A call, meeting, hallway conversation, or rough voice note can become starting material. Codex can help turn the raw version into a plan, draft, artifact, or next action.

> A lot of plans get better when the model has access to the messy version of what you think.
>
> source [Jason Liu]

Steering

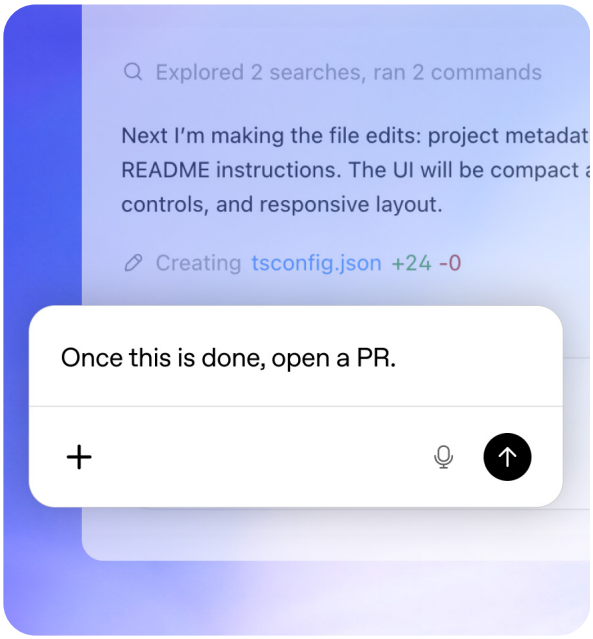
SECTION 03

Shape the queue while Codex works

Voice becomes more useful when it is paired with [steering].

Steering means adding the next instruction while Codex is already working. You can correct direction, add context, approve the next step, or queue up the next action after a tool call. When you are reviewing a site, that might sound like:

```
[Steering]
“Make this smaller.”
“This copy is wrong.”
“The spacing between these two things
feels off.”
“Once this is done, open a PR.”
“Wait for the preview deployment.”
“Show me the preview link before
anything is posted.”
```

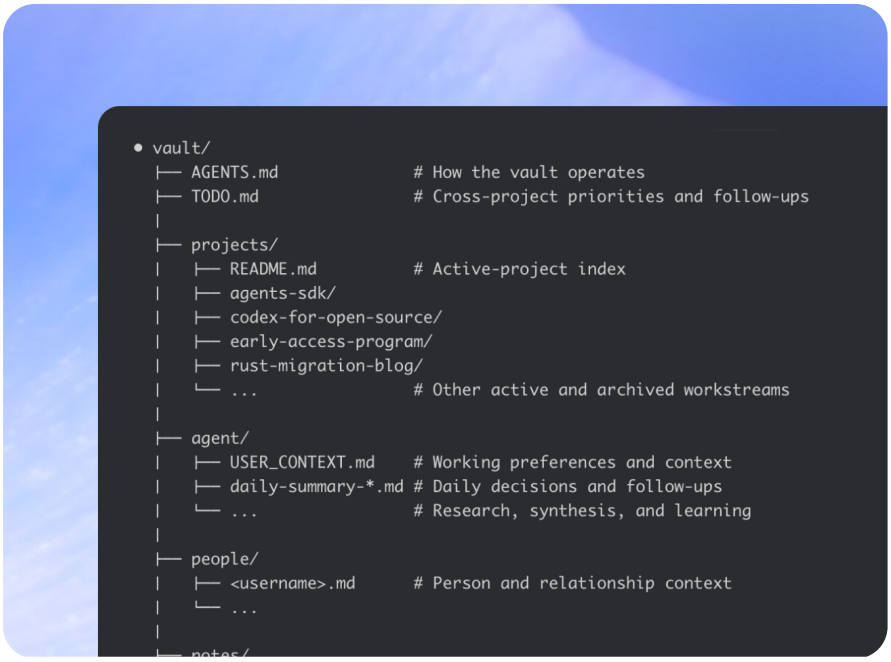


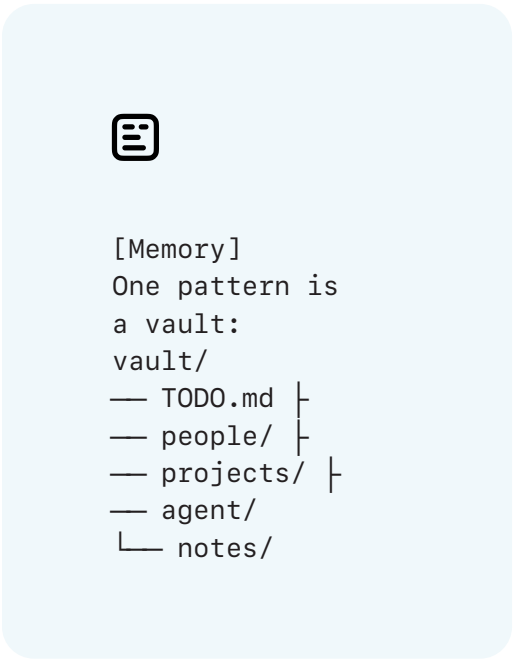
Memory

SECTION 04

Make memory something you can review

Memory acts as a notebook to provide context for actions. As threads last longer, they need [memory] outside the conversation. While message history helps, it's not always enough. Useful context should become something you can open, edit, diff, and reuse.





The vault can hold the rolling context around your work: people, decisions, open loops, daily notes, project state, and the details that would otherwise get lost between threads.

Keep this distinction clear:
Repositories hold code.
The vault holds rolling context around the work.

When the vault lives in GitHub, diffs become a review surface for memory. You can see what Codex thought was important enough to write down.

That review step matters. Long-running threads should not quietly accumulate vague impressions in conversation history. They should record what changed:

```
[Record]
This person prefers this.
This project is waiting on that.
This decision was made.
This loop is closed.
```

```
[Memory instruction]
As people are mentioned,
update the relevant
people notes.

As projects move forward,
update the project page.

As loops close,
mark them closed.

As decisions are made,
write down the decision
and why it matters.
```

Computer and browser use

SECTION 05

Decide what the thread can touch

Once a thread has memory, the next question is practical: [What can it use?]

Connectors extend Codex into the places where work often appears first: Slack threads, inboxes, calendars, docs, and issue trackers.

For this workflow, it helps to separate the surfaces:

[$browser]
Local web surfaces, previews, and annotations

[@chrome]
Signed-in browser sessions and authenticated tabs

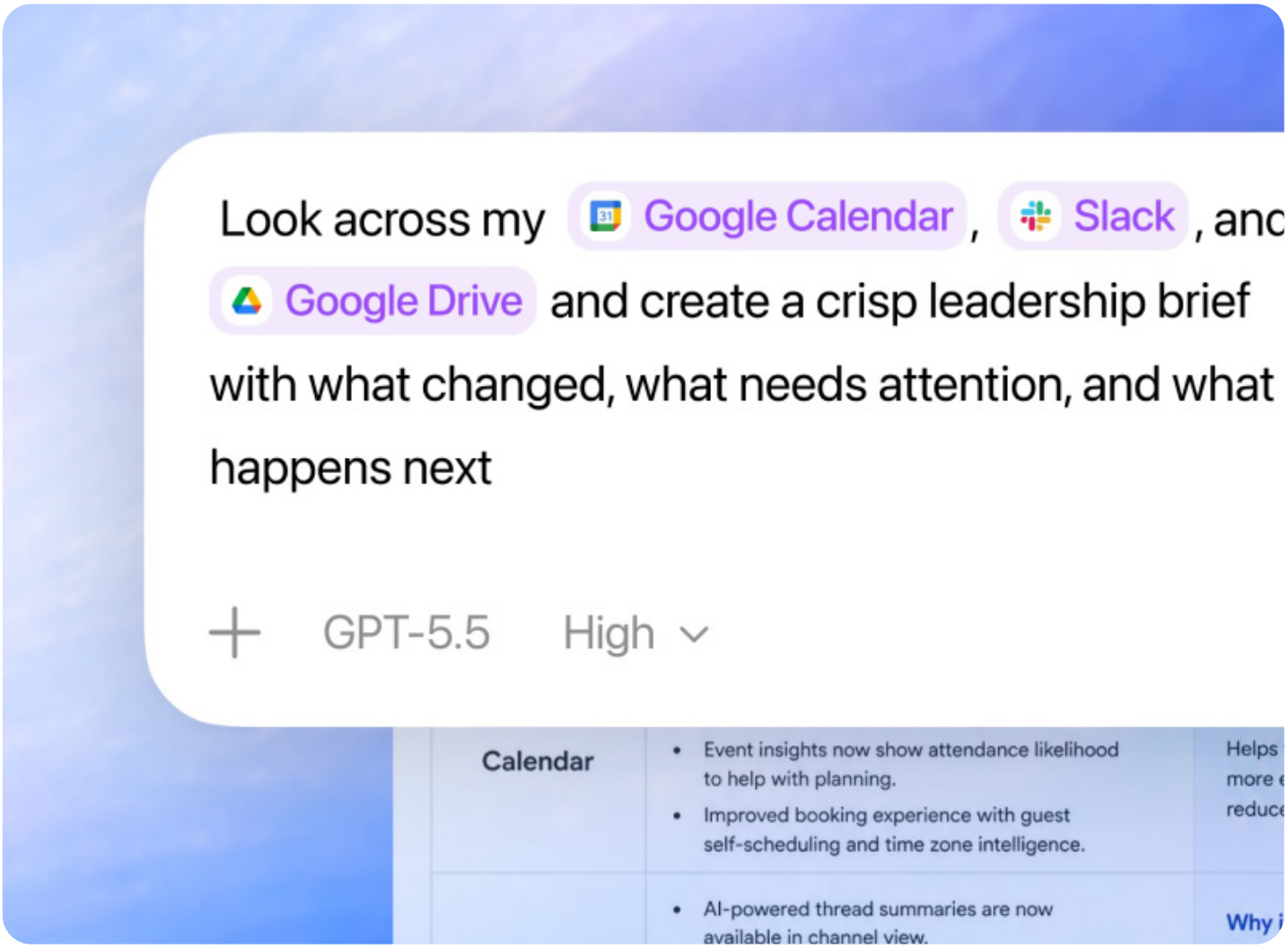
[@computer]
GUI-only work that requires clicking

[Connectors]
Slack, Gmail, Calendar, GitHub, and other work surfaces

[Skills]
Reusable workflows Codex can repeat

If you are iterating on a local app, use the browser surface. If the task depends on logged-in state or multiple authenticated tabs, use Chrome. If the only way to complete the task is through a desktop app, use computer use with clear permissions and review.

Skills make repeated work easier to reuse. Once a workflow succeeds, you can often package the instructions, references, and scripts so Codex doesn't need to be retaught every time.



Remote control

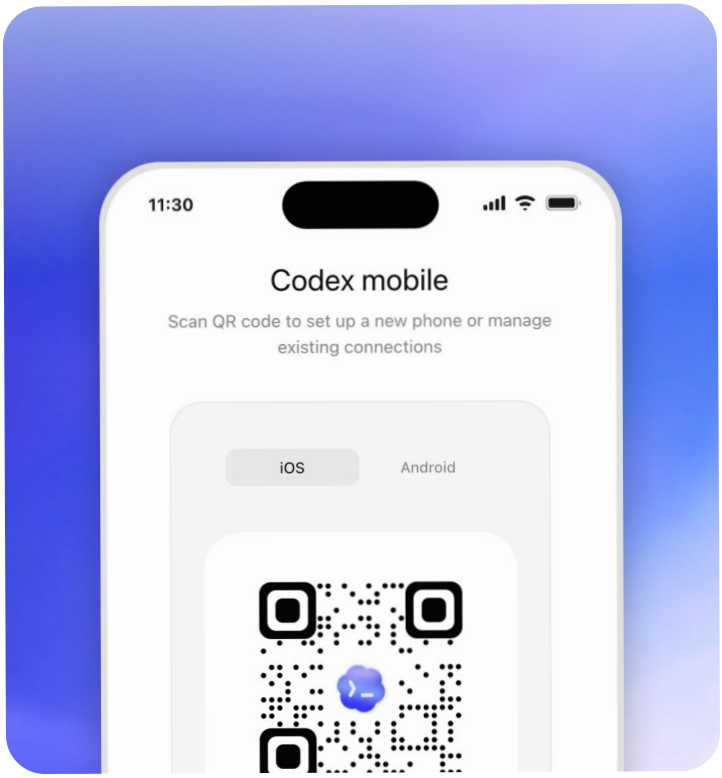
SECTION 06

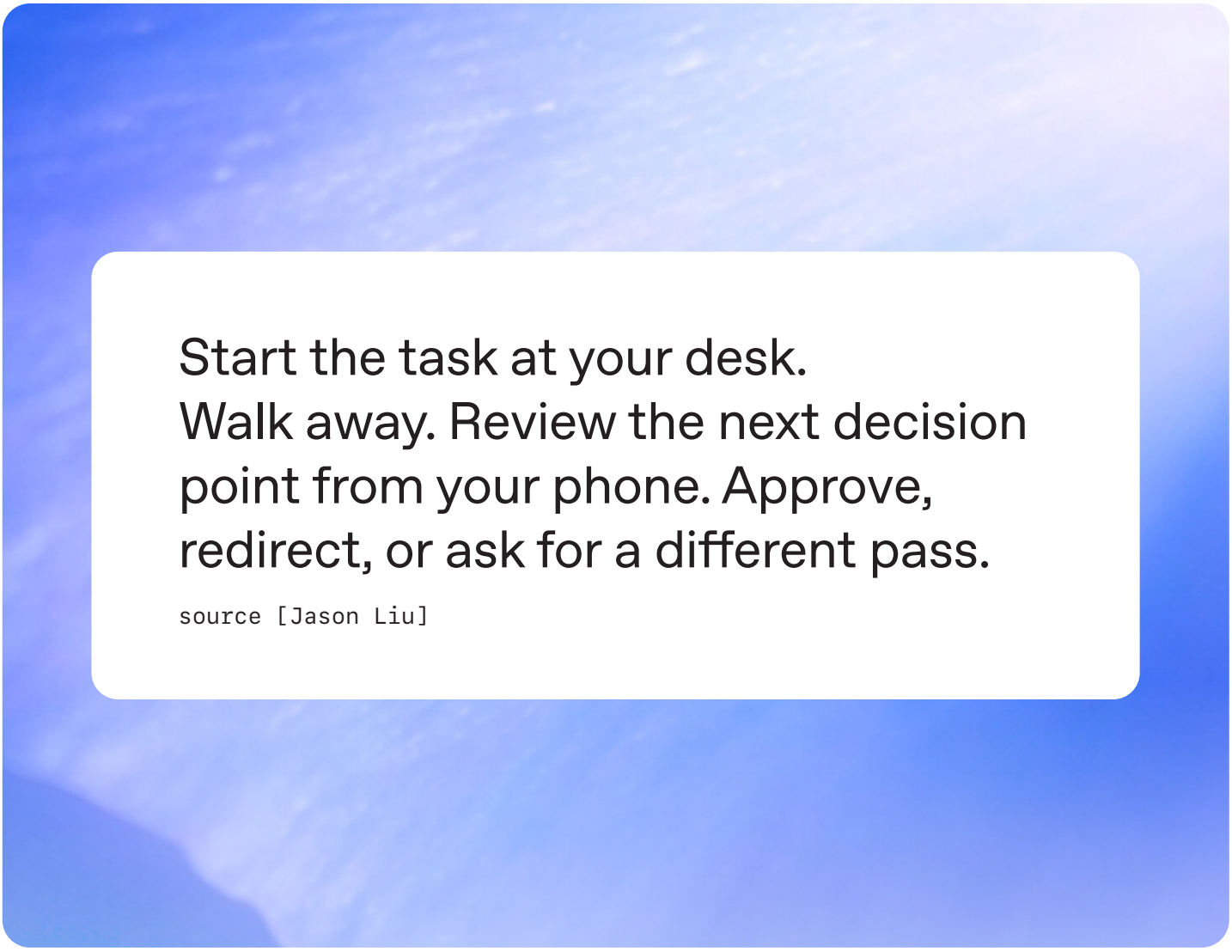
Keep longer loops portable

[Remote control] makes it easier to stay close to long-running tasks.

Codex can keep working from the machine where your files, permissions, and local setup already live. You can check in from another device, review what it found, answer a question, approve the next step, or change direction.

This matters most when Codex is already doing something that takes time.





Remote control is not a reason to skip reviews. It is a way to keep enough attention on the loop to unblock the next move.

Thread automations

SECTION 07

Schedule Codex to return

[Thread automations] are heartbeat-style recurring wake-up calls attached to the current thread. They tell Codex to return to the same conversation on a cadence, preserving the context instead of starting from scratch each time.

A thread can have multiple schedules. It can run until a condition is met. It can adjust cadence as the task changes.

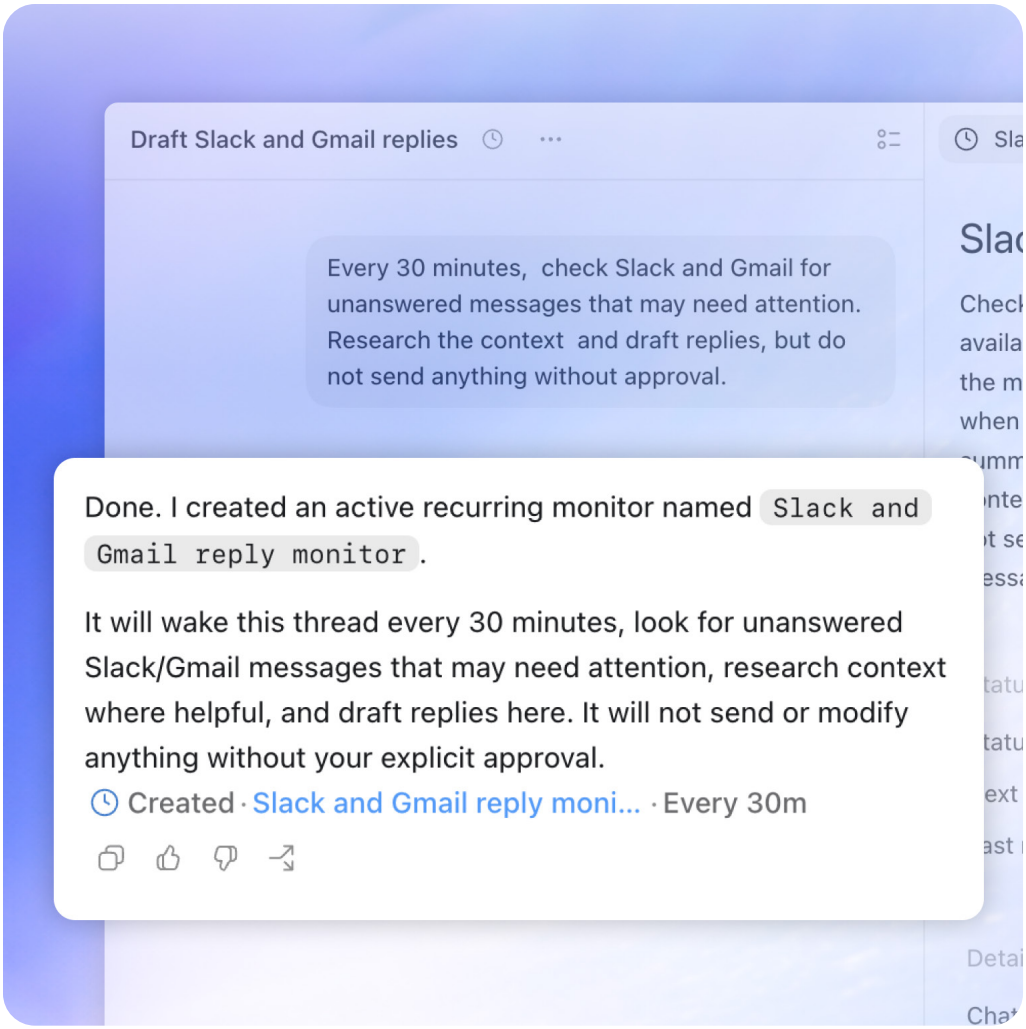
```
[Normal prompt]
Do this now.
```

versus

```
[Thread automation]
Keep checking this
and move it forward when
something changes.
```

That might mean checking a pull request, Slack thread, inbox, deployment, document, customer support thread, or long-running command.

```
[Thread automation]
Every 30 minutes,
check Slack and Gmail
for unanswered messages
that may need attention.
Research the context
and draft replies, but
do not send anything
without approval.
```

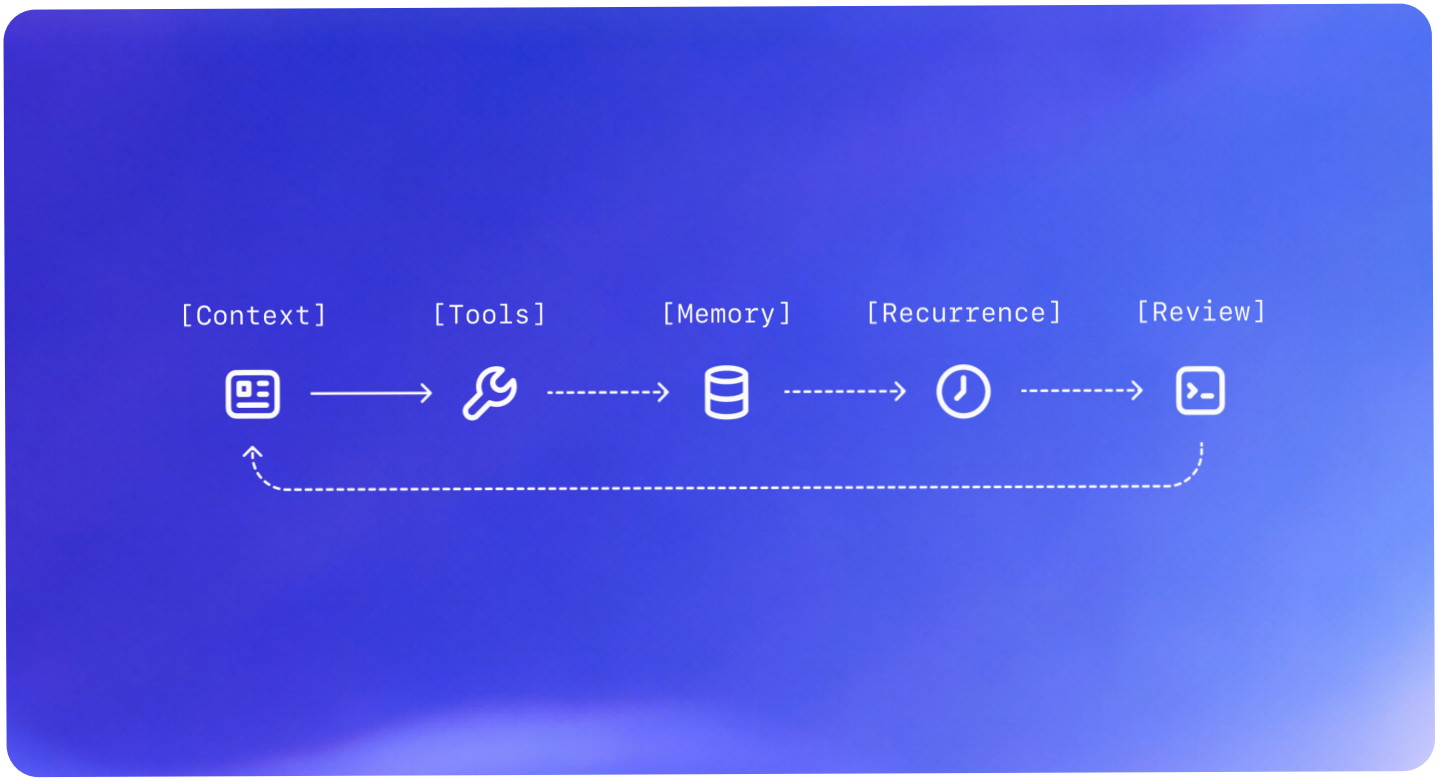


Three examples of loops

SECTION 08

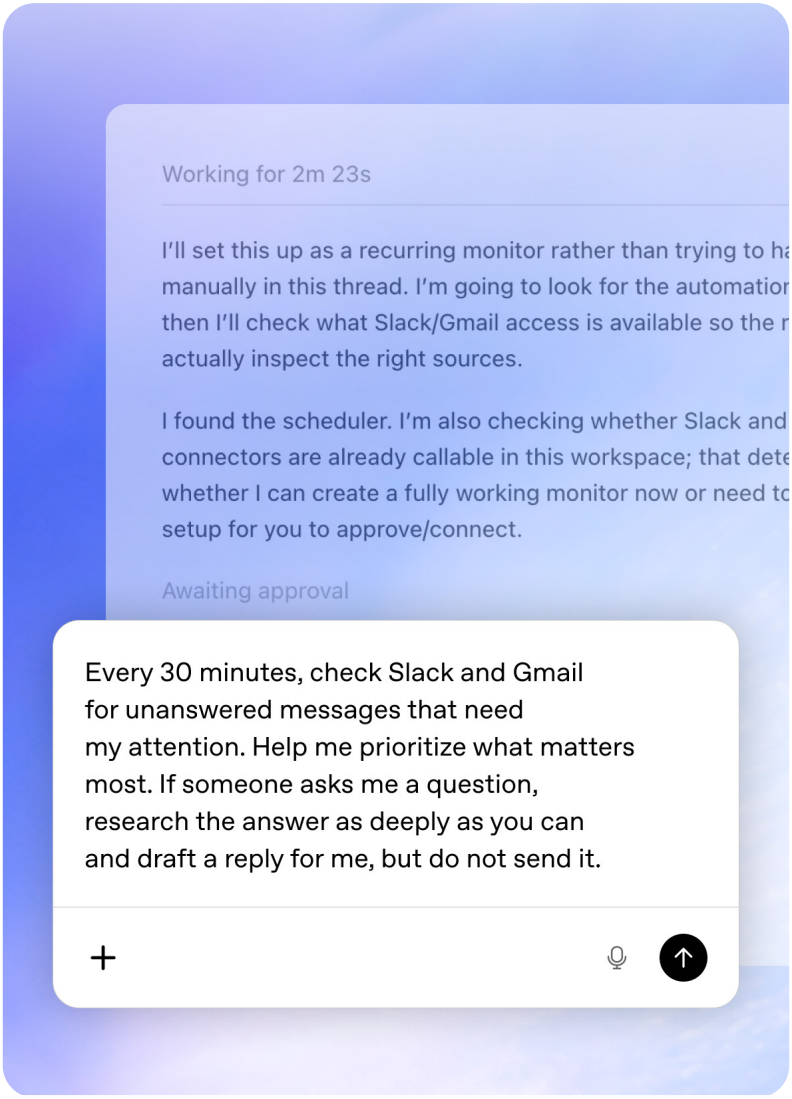
Real ways to apply Codex in your workflows

The power comes from the [loop] across features: context, tools, memory, recurrence, and review.



Loop 1 [Chief of Staff]

Codex checks Slack and Gmail on a schedule, finds messages that may need attention, searches for the context behind them, and drafts replies. The user still decides what gets sent.

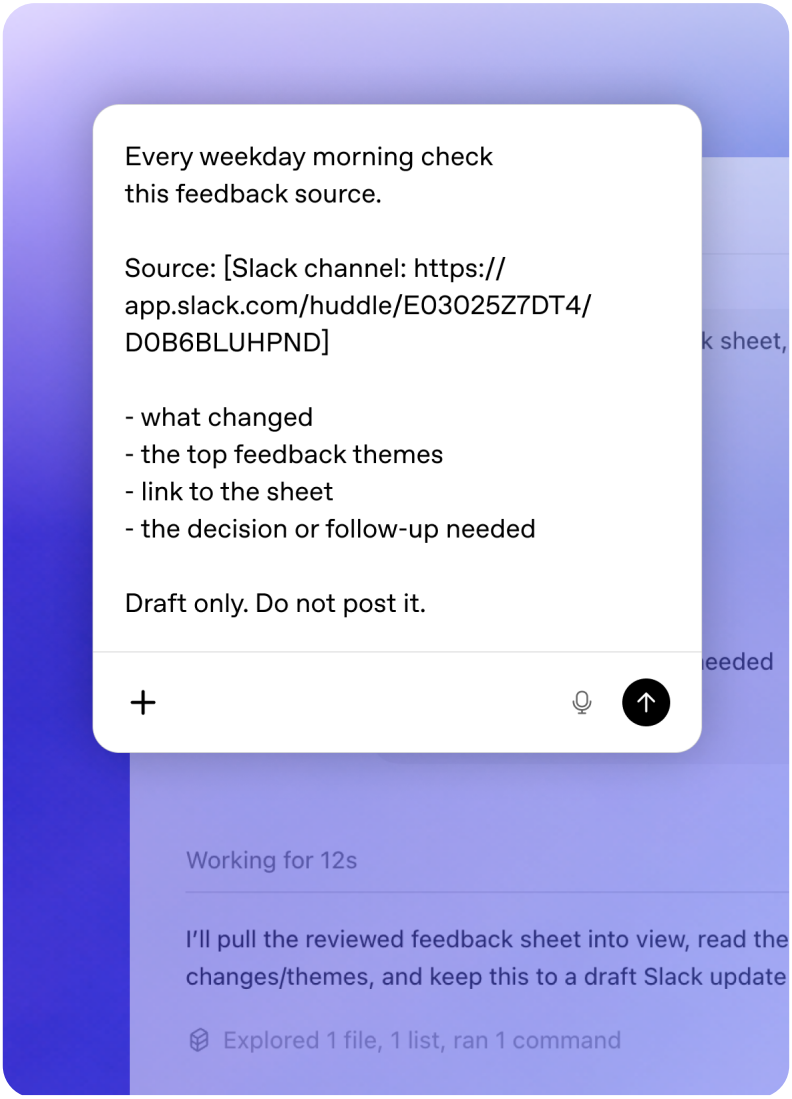



```
Loop 1
[Chief of Staff]

[What Codex prepares]
Open messages
Relevant context
Draft replies
Questions that
need judgment

[You decide]
Approval
Tone
Timing
Final decision
```

Loop 2 [Monitor for feedback]

Codex watches a Slack thread for animation feedback, updates the Remotion project, re-renders the piece, and prepares the revision for review.

The loop crosses tools: Slack for feedback, Remotion for rendering, and computer use when the upload or review surface requires a GUI.

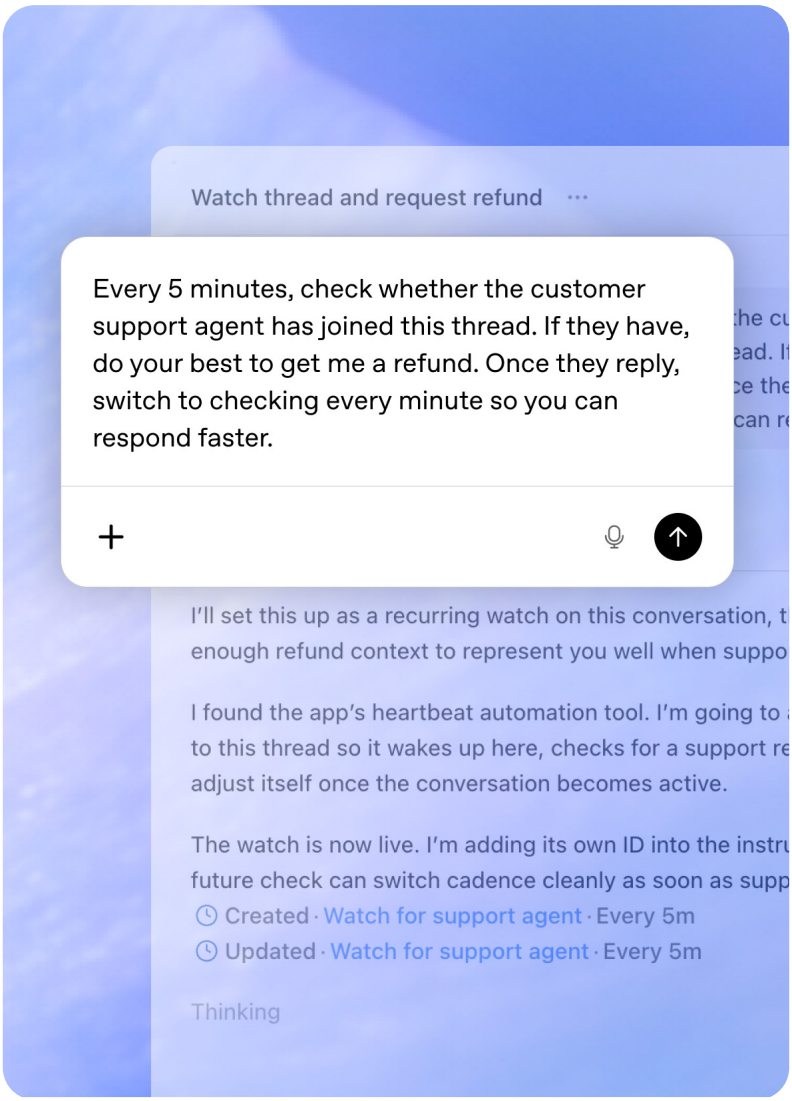



```
Loop 2
[Monitor for feedback]

[What Codex prepares]
Feedback summary
Updated render
Revision notes
Next review link

[You decide]
Creative judgment
Final approval
Publishing decision
```

Loop 3 [Get a refund]

Codex checks whether a customer support agent has joined, then prepares the next response when the conversation changes.

The task can continue while the user is away, but the action stays bounded.




```
Loop 3
[Get a refund]

[What Codex prepares]
Status checks
Draft responses
Useful evidence
Recommended next step

[You decide]
Consent
Approval
Any irreversible action
```

Goals

SECTION 09

Set goals Codex can verify

A weak goal asks Codex to implement a plan.

A [stronger goal] gives Codex something to test against: expected behavior, review criteria, constraints, or a clear definition of done.

```
[Weak goal]
“Implement
the plan
in this
Markdown
file.”
```

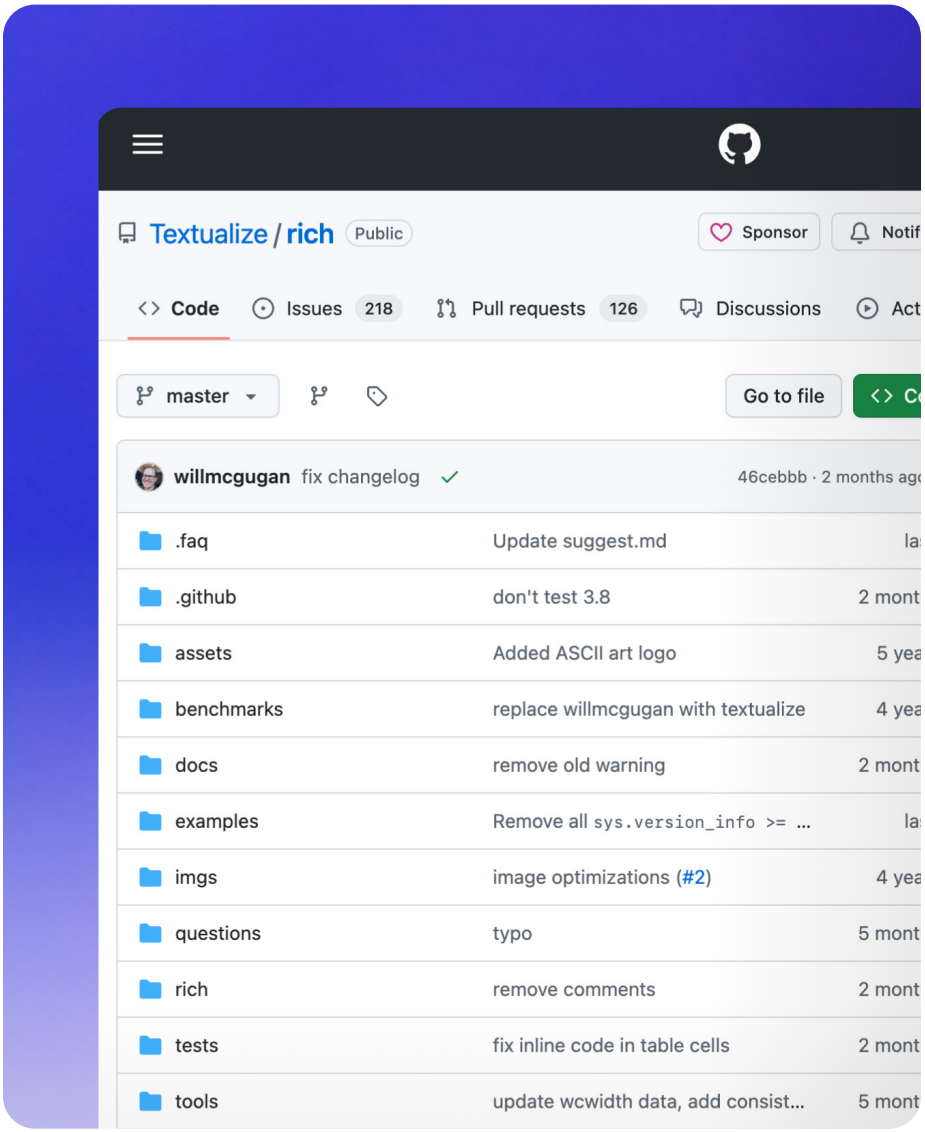
versus

```
[Strong goal]
“Port this library, keep the public
API compatible, and use the original
unit tests as the success check.
The work is ready for review when
the same tests pass and the
differences are documented.”
```

The Rich-to-Rust example makes this concrete. The goal was not only to port the library. The goal was to port it in a way that could pass the original unit tests.

That test suite gave the run a real standard. The work was not ready until the new implementation passed the same tests.

Migrating [Rich into Rust]



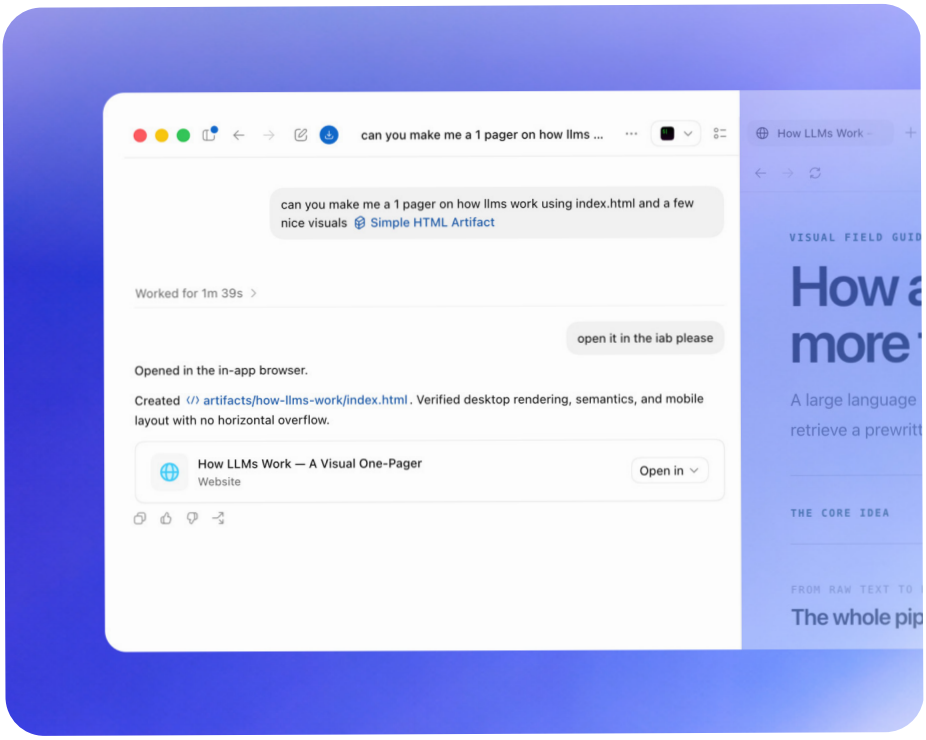
Side panel

SECTION 10

Make the artifact part of the loop

The [side panel] is easy to understand as a preview surface, but that undersells it. It's where Codex becomes more than a chat interface. You can inspect the same object Codex is acting on, leave comments, review changes, and keep the artifact inside the thread.

[Side panel]



[Inspect artifacts]
Markdown, spreadsheets, CSVs, PDFs, and slides can all live there. Markdown can be reviewed with comments. Spreadsheets can render formulas and support cell edits. CSVs become tables instead of raw text. PDFs render directly. Slides can be created and reviewed without leaving the app.

[Operate web surfaces]
The in-app browser makes small web artifacts more useful: index.html, Storybook, Remotion Studio, Slidev, Streamlit, and Jupyter. The smallest version is often enough. A single index.html file with JavaScript and CSS can become a live surface for interaction and review.

[Review changes]
You and Codex can look at the same object while the work is still moving. Comments become instructions. The artifact becomes context.

The side panel is where Codex stops being only a chat app and starts becoming the place the work happens.

source [Jason Liu]

Jason Liu's practice shows how quickly Codex can [build systems] for work.

With new ways to move projects forward, tasks become easier to pick up, review, and continue without losing context. People can spend less time restarting and more time building on what's already in motion.

OpenAI