

How OpenAI uses Codex



OpenAI 如何 使用 Codex



Contents

Introduction	3
Use Cases	
Code understanding	4
Refactoring and migrations	5
Performance optimization	6
Improving test coverage	7
Increasing development velocity	8
Staying in flow	9
Exploration and ideation	10
Best Practices	11
Looking Ahead	12

目录

介绍 3 用例 代码理解 4 重构与迁移 5 性能优化 6 提高测试覆盖率 7 提高开发速度 8 保持专注状态 9 探索与创意构思 10 最佳实践 11 前瞻 12
--

Introduction

介绍

Codex is used daily across numerous technical teams at OpenAI like Security, Product Engineering, Frontend, API, Infrastructure, and Performance Engineering. Teams are using it to accelerate a range of engineering tasks, from understanding complex systems and refactoring large codebases to shipping new features and resolving incidents under tight deadlines.

Drawing from interviews with OpenAI engineers and internal usage data, we've compiled use cases and best practices that highlight how Codex helps our teams move faster, improve work quality, and manage complexity at scale.

Codex 每天被 OpenAI 的多个技术团队使用，诸如 Security、产品工程、前端、API、基础设施 和 性能工程。团队使用它来加速各种工程任务，从理解复杂系统 和 重构大型代码库 到 发布新功能 和 在紧迫的截止时间内 解决事件。

基于与 OpenAI 工程师的访谈和内部使用数据，我们汇总了用例和最佳实践，突出展示了 Codex 如何帮助我们的团队更快地行动、提高工作质量并在规模上管理复杂性。

Code understanding

Codex helps our teams get up to speed quickly in unfamiliar parts of the codebase when onboarding, debugging, or investigating an incident.

They often use Codex to locate the core logic of a feature, map out relationships between services or modules, and trace data flow through a system. It also helps surface architecture patterns or missing pieces of documentation that would otherwise require significant manual effort to generate.

During incident response, Codex helps engineers ramp into new areas quickly by surfacing interactions between components or tracing how failure states propagate across systems.

Anecdotes from our teams

When I fix a bug, I use Ask mode to see where else in the codebase the same issue might appear.

Performance Engineer, Retrieval Systems

When I’m on-call, I paste the stack trace and ask Codex where the auth flow lives. It jumps straight to the right files so I can triage fast.

Site Reliability Engineer, API Platform

Codex answers my ‘Where would I do this?’ repo questions across Terraform and Python way faster than grep.

DevOps Engineer, Infrastructure Services

Try using Codex for code understanding with these sample prompts:

- ☐ Where is the authentication logic implemented in this repo?
- ☐ Summarize how requests flow through this service from endpoint to response.
- ☐ Which modules interact with [insert module name] and how are failures handled?

代码理解

Codex 帮助我们的团队在入职、调试或调查事故时，迅速熟悉代码库中不熟悉的部分。

他们经常使用 Codex 来定位一个功能的核心逻辑、绘制服务或模块之间的关系，并跟踪数据在系统中的流动。它还可以帮助发现架构模式或缺失的文档，这些内容否则需要大量手工工作来生成。

在事件响应期间，Codex 帮助工程师快速熟悉新的领域，通过呈现组件之间的交互或追踪故障状态如何在系统间传播。

我们团队的轶事

当我修复一个错误时，我使用 Ask 模式 来查看相同问题可能在代码库的其他何处出现。

性能工程师，检索系统

当我值班时，我粘贴堆栈跟踪并请 Codex 指出认证流程 位于何处。它直接跳到正确的文件，这样我就能快速进行分诊/初步诊断。

站点可靠性工程师，API 平台

Codex 回答我的 ‘我会在哪里执行此操作？’ 关于 Terraform 和 Python 的仓库问题，比 grep 快得多。

DevOps 工程师，基础设施服务

尝试使用 Codex 来进行代码理解，使用以下示例提示：

- ☐ 这个代码仓库中的认证逻辑在哪里实现？
- ☐ 总结从入口点到响应，请求如何在此服务中流转。
- ☐ 哪些模块与 insert module name 交互，以及故障如何处理？

Refactoring and migrations

Codex is commonly used to make changes that span multiple files or packages. For example, when engineers are updating an API, changing how a pattern is implemented, or migrating to a new dependency, Codex makes it easy to apply changes consistently.

It’s especially useful when the same update needs to be made across dozens of files, or when the update requires awareness of structure and dependencies that aren’t easily caught with a regex or find-and-replace.

They’re also using it for code cleanup by breaking up oversized modules, replacing old patterns with modern ones, or preparing code for better testability.

Anecdotes from our teams

Codex swapped every legacy `getUserById()` for our new service pattern and opened the PR. It did in minutes what would’ve taken hours.

Backend Engineer,
ChatGPT Web

To clear launch blockers, I have Codex scan for every instance of the old pattern, summarize the impact in Markdown, then open PRs with the fixes.

Product Engineer,
ChatGPT Enterprise

Try using Codex for refactoring and migrations with these sample prompts:

- ☐ Split this file into separate modules by concern and generate tests for each one.
- ☐ Convert all callback-based database access to async/await.

重构与迁移

Codex 通常用于跨多个文件或包进行更改。例如，当工程师在更新 API、改变模式的实现方式或迁移到新依赖时，Codex 使得一致地应用更改变得容易。

当需要在数十个文件中进行相同的更新，或当更新需要意识到结构和依赖项，而这些内容不容易通过正则表达式或查找并替换捕获时，这一点尤其有用。

他们还将其用于代码清理，例如拆分过大的模块、用现代模式替换旧模式，或为更好的可测试性准备代码。

我们团队的轶事

Codex 交换了每个遗留 `getUserById()` 为我们的新服务模式 `getUserById()` 创建了，并打开了拉取请求。它在几分钟内完成了原本需要数小时的工作。

后端工程师，
ChatGPT Web

为了解除发布阻塞项，我有 Codex 扫描旧模式的每个实例，用 Markdown 总结影响，然后打开带有修复的合并请求。

产品工程师，
ChatGPT 企业版

尝试使用 Codex 进行重构与迁移，以下是一些示例提示：

- ☐ 按关注点将此文件拆分为独立模块并为每个模块生成测试。
- ☐ 将所有基于回调的数据库访问转换为 async/await。

Performance optimization

Codex is used to identify and address performance bottlenecks.

During tuning or reliability efforts, engineers prompt Codex to analyze slow or memory-intensive code paths, such as inefficient loops, redundant operations, or costly queries and suggest optimized alternatives, often resulting in meaningful gains in efficiency and reliability.

Codex is also used to support code health by identifying risky or deprecated patterns that are still in active use. Our teams lean on it to help reduce long-term tech debt and proactively prevent regressions.

Anecdotes from our teams

I use Codex to scan for repeated expensive DB calls. It's great at flagging hot paths and drafting batched queries I can later tune.

Infrastructure Engineer,
API Reliability

Codex is great for spotting performance issues quickly—I save 30 minutes of work by spending 5 minutes on a prompt.

Platform Engineer,
Model Serving

Try using Codex for performance optimization with these sample prompts:

- ☐ Optimize this loop for memory efficiency and explain why your version is faster.
- ☐ Find repeated expensive operations in this request handler and suggest caching opportunities.
- ☐ Suggest a faster way to batch DB queries in this function.

性能优化

Codex 用于识别和解决性能瓶颈。

在调优或可靠性工作中，工程师会提示 Codex 分析慢或占用大量内存的代码路径，例如 低效循环、冗余操作或 代价高的查询，并建议 优化替代方案，通常会带来显著的 效率 和 可靠性 提升。

Codex 还用于通过识别仍然 正在使用 的 有风险或已弃用的模式 来支持 代码健康。我们的 团队 依赖它来帮助减少长期的 技术债务 并主动防止 回归。

我们团队的轶事

我使用 Codex 来 扫描 重复内容 昂贵的数据库调用。它非常擅长标记 关键路径和起草批量查询 I 以后可以调整。

基础设施工程师，API 可 靠性

Codex 非常适合快速发现性能问题——我通过 在一个提示上花 5 分钟节省了 30 分钟的工作。

平台工程师，模型服 务

尝试使用 Codex 进行性能优化，使用以下示例提示：

- ☐ 优化此循环以提高内存效率，并解释为什么你的版本更快。
- ☐ 查找重复的高开销操作，在此请求处理程序中并建议缓存机会。
- ☐ 建议更快的批量数据库查询方法 来优化此函数。

Improving test coverage

Codex helps engineers write tests faster — especially in places where coverage is thin or completely missing.

When working on a bug fix or refactor, engineers often ask Codex to suggest tests that cover edge cases or likely failure paths. For new code, it can generate unit or integration tests based on the function signature and surrounding logic.

Codex is particularly helpful for identifying boundary conditions like empty inputs, max length, or unusual but valid states that are often missed in initial tests.

Anecdotes from our teams

I point Codex at low-coverage modules overnight and wake up to runnable unit-test PRs.

**Frontend Engineer,
ChatGPT Desktop**

When switching mono-repo branches is painful, I have Codex write the tests and kick-off CI while I keep working on my branch.

**Backend Engineer,
Payments & Billing**

Try using Codex for improving test coverage with these sample prompts:

- ☐ Write unit tests for this function, including edge cases and failure paths.
- ☐ Generate a property-based test for this sorting utility.
- ☐ Extend this test file to cover missing scenarios around null inputs and invalid states.

提高测试覆盖率

Codex 帮助工程师更快地编写测试——尤其是在覆盖率薄弱或完全缺失的地方。

在处理修复错误或重构时，工程师经常请 Codex 建议覆盖边界情况或可能的失败路径的测试。对于新代码，它可以根据函数签名和周围逻辑生成单元或集成测试。

Codex 对识别边界条件尤其有帮助，比如空输入、最大长度或不寻常但有效的状态，这些通常在初始测试中被遗漏。

我们团队的轶事

我把 Codex 指向低覆盖率模块过夜，然后醒来就看到可运行的单元测试拉取请求。

**前端工程师，
ChatGPT 桌面版**

当切换单一代码库（mono-repo）分支很痛苦时，我让 Codex 编写测试并启动持续集成，同时我继续在我的分支。

**后端工程师，
支付与账单**

尝试使用 Codex 来提高测试覆盖率，使用这些示例提示：

- ☐ 为此函数编写单元测试，包括边界情况和失败路径。
- ☐ 为此排序工具生成基于属性的测试。
- ☐ 扩展此测试文件以覆盖缺失的围绕空输入和无效的情形状态。

Increasing development velocity

Codex helps teams move faster by accelerating both the start and end of the development cycle.

When kicking off a new feature, engineers use it to scaffold boilerplate — generating folders, modules, and API stubs to get runnable code up quickly without hand-wiring every piece.

As projects approach release, Codex helps meet tight deadlines by handling smaller but essential tasks like triaging bugs, filling in last-mile implementation gaps, and generating rollout scripts, telemetry hooks, or config files.

It’s also used to turn product feedback into starter code. Engineers often paste in a user request or spec and have Codex generate a rough draft they can return to and refine later.

Anecdotes from our teams

I was in meetings all day and still merged 4 PRs because Codex was working in the background.

Product Engineer,
ChatGPT Enterprise

Codex helped ship 3-4 low-priority fixes perfectly that would’ve languished in the backlog, which was super empowering.

Full-Stack Engineer,
Internal Tools

Try using Codex for increasing development velocity with these sample prompts:

- ☐ Scaffold a new API route for POST /events with basic validation and logging.
- ☐ Generate a telemetry hook for tracking success/failure of the new onboarding flow, using this template [insert example of your telemetry code].
- ☐ Create a stub implementation based on this spec: [insert spec or product feedback].

提高开发速度

Codex 通过加速开发周期的开始和结束，帮助团队更快地推进。

在启动一个新功能时，工程师使用它来搭建样板代码 —— 生成文件夹、模块 和 API 存根，以便快速得到可运行代码，而无需手工连接每一部分。

随着项目接近发布，Codex 通过处理诸如缺陷分类（分级）、填补最后阶段实现缺口，以及生成部署脚本、遥测钩子或配置文件等较小但重要的任务来帮助满足紧迫的截止时间。

它还用于将产品反馈转换为起始代码。工程师经常粘贴用户请求或规范，让Codex生成他们之后可以回去完善的草稿。

我们团队的轶事

我一整天都在会议中，但仍然已合并 4 PRs，因为 Codex 在后台运行。

Product Engineer,
ChatGPT 企业版

Codex 帮助发布了 3-4 个低优先级修复，完美地本来会在待办事项积压中被搁置，这让人非常有动力。

Full - Stack Engineer,
Internal Tools

尝试使用 Codex 通过这些示例提示来提高开发速度：

- ☐ 为 POST /events 搭建一个新的 API 路由，包含基本验证和日志记录。
- ☐ 为跟踪新入职流程的成功/失败生成一个遥测钩子，使用此模板 insert example of your telemetry code 。
- ☐ 基于该规范创建一个存根实现： insert spec or product feedback

Staying in flow

Codex helps our engineers stay productive when their schedules are fragmented and filled with interruptions.

It’s used to capture unfinished work, turn notes into working prototypes, or spin off exploratory tasks that can be revisited later. This makes it easier to pause and resume work without losing context, especially when they’re on call or have a lot of meetings.

Anecdotes from our teams

If I spot a drive-by fix, I fire a Codex task instead of swapping branches and review its PR when I’m free.

Backend Engineer,
ChatGPT API

I routinely forward Slack threads, Datadog traces, issues and more to Codex so I can stay focused on high priority work.

API Engineer,
Infrastructure Observability

Try using Codex for staying in flow with these sample prompts:

- ☐ Generate a plan to refactor this service and split it into smaller modules.
- ☐ Stub out the retry logic and add a TODO — I’ll fill in the backoff logic later.
- ☐ Summarize this file so I can pick up where I left off tomorrow.

保持专注状态

Codex 帮助我们的 engineers 在日程安排零碎且充满 interruptions 时保持高效。

它用于记录未完成的工作，将笔记转化为工作原型，或拆分出可在以后重访的探索性任务。这使得在不丢失上下文的情况下暂停和恢复工作变得更容易，尤其是在他们值班或有很多会议时。

我们团队的轶事

如果我发现临时修复，我会发起一个 Codex 任务，而不是切换分支，并在有空时审查它的拉取请求。

后端工程师，
ChatGPT API

我经常将 Slack 线程、Datadog 跟踪、问题等转发到 Codex，以便我可以专注于高优先级的工作。

API 工程师，基础设施可观测性

尝试使用 Codex 通过这些示例提示保持专注状态：

- ☐ 生成一个重构该服务并将其拆分为更小模块的计划。
- ☐ 为重试逻辑写出存根并添加 待办事项 — 我稍后会补充退避逻辑。
- ☐ 总结此文件，以便我明天从中断处继续。

Exploration and ideation

Codex is also useful for open-ended work like finding alternative solutions or validating design decisions. You can prompt for different ways of solving a problem, explore unfamiliar patterns, or pressure-test assumptions. This helps surface tradeoffs, expand design options, and sharpen implementation choices.

It’s also used to identify related bugs. Given a known issue or deprecated method, Codex can identify similar patterns elsewhere in the code, making it easier to catch regressions or finish cleanup work.

Anecdotes from our teams

Codex helps me solve the cold-start problem — I paste a spec and docs and it scaffolds code or shows me what I forgot.

Product Engineer,
ChatGPT Desktop

After I fix a bug I ask Codex where similar bugs might lurk, then spin follow-up tasks.

Performance Engineer,
Retrieval Systems

Try using Codex for exploration and ideation with these sample prompts

- ☐ How would this work if the system were event-driven instead of request/response?
- ☐ Find all modules that manually build SQL strings instead of using our query builder.
- ☐ Rewrite this in a more functional style, avoid mutation and side effects.

探索与创意构思

Codex 对于开放式工作也很有用，例如寻找替代方案或验证设计决策。你可以提示不同的解决问题的方法，探索不熟悉的模式，或对假设进行压测。这有助于呈现权衡、扩展设计选项并完善实现选择。

它也用于识别相关错误。针对已知问题或已弃用的方法，Codex 可以识别代码中其他地方的类似模式，从而更容易捕捉回归或完成清理工作。

我们团队的轶事

Codex 帮助我解决冷启动问题——我粘贴规范和文档，它搭建代码脚手架或显示我忘记的内容。

产品工程师，
ChatGPT 桌面版

修复错误后，我会询问 Codex 相似错误可能出现的位置，然后衍生后续任务。

性能工程师，检索系统

尝试使用 Codex 进行探索与创意构思，参考这些示例提示

- ☐ 如果系统是事件驱动而不是请求/响应，这将如何工作？
- ☐ 查找所有手动构建 SQL 字符串而不是使用我们查询构建器的模块。
- ☐ 以更函数式的风格重写，避免可变性和副作用。

Best practices

Codex works best when it’s given structure, context, and room to iterate. Here are some of the habits OpenAI teams are cultivating to get consistent value out of it in day-to-day work.

Start with Ask Mode	For large changes, start by prompting Codex for an implementation plan using Ask mode, which then becomes the input for follow-up prompts when you switch to Code Mode. This two-step flow keeps Codex grounded and helps avoid errors in its output. Codex works best with well-scoped tasks that would take you or a teammate about an hour to complete or a few hundred lines of code to implement. As models improve, expect the size of the tasks it can take on to increase.
Iteratively improve Codex’s development environment	Setting a startup script, environment variables, and internet access significantly reduces Codex’s error rate. As you run tasks, look for build errors that can be corrected in Codex’s environment configuration. This may take a few iterations, but gives significant efficiency gains in the long run.
Structure your prompt as if you are writing a Github Issue	Codex responds better when prompts mirror how you’d describe a change in a PR or issue. That means including file paths, component names, diffs, and doc snippets when relevant. Prompting with patterns like “Implement this the same way it’s done in [module X]” improves results.
Use the Codex task queue as a lightweight backlog	Fire off tasks to capture tangential ideas, partial work, or incidental fixes. There’s no pressure to generate a full PR in one go. Codex works well as a staging area you can return to when you’re back in focus.

最佳实践

Codex 在被赋予结构、上下文和迭代空间时效果最佳。以下是 OpenAI 团队在日常工作中培养的一些习惯，以便从中持续获得价值。

Start with Ask Mode	对于大型变更，首先使用 Ask 模式提示 Codex 制定实现计划，然后在切换到 Code Mode 时将该计划作为后续提示的输入。这一两步流使 Codex 保持扎实并有助于避免其输出中的错误。Codex 最适合范围明确的任务，这类任务大约需要你或队友一小时完成或由几百行代码实现。随着模型的改进，预计它能承担的任务规模会增大。
Iteratively improve Codex’s development environment	Setting a startup script, environment variables, and internet access significantly reduces Codex’s error rate. As you run tasks, look for build errors that can be corrected in Codex’s environment configuration. This may take a few iterations, but gives significant efficiency gains in the long run.
Structure your prompt as if you are writing a Github Issue	Codex responds better when prompts mirror how you’d describe a change in a PR or issue. That means including file paths, component names, diffs, and doc snippets when relevant. Prompting with patterns like “Implement this the same way it’s done in [module X]” improves results.
Use the Codex task queue as a lightweight backlog	Fire off tasks to capture tangential ideas, partial work, or incidental fixes. There’s no pressure to generate a full PR in one go. Codex works well as a staging area you can return to when you’re back in focus.

Use AGENTS.md to supply persistent context

Maintain an AGENTS.md file to help Codex operate more effectively in your repo across prompts. These files typically include naming conventions, business logic, known quirks, or dependencies Codex can’t infer from the code alone. Learn more on structuring your [AGENTS.md](#) file in the [docs](#).

Leverage “Best of N” to improve output

The Best-of-N feature lets you simultaneously generate multiple responses for a single task to quickly explore multiple solutions and pick the best one. For more complicated tasks, you can review several iterations and combine parts of different responses to get a stronger result.

Looking ahead

Codex is still in research preview, but it’s already making a real impact in how we build, helping us move faster, write better code, and take on work that would’ve otherwise never been prioritized.

We’re excited by the potential ahead — as our models get better and Codex becomes more deeply integrated into our workflows, we’re looking forward to unlocking even more powerful ways to develop software with it. We’ll continue to share what we learn along the way.

使用 AGENTS.md 提供持久化上下文

维护一个 AGENTS.md 文件，以帮助 Codex 在你的 repo 中跨提示更有效地运行。这些文件通常包括 命名约定、业务逻辑、已知问题，或依赖项 Codex 无法仅从代码中推断出来。请在 docs 中了解有关构建你的 AGENTS.md 文件的更多信息。

利用 “Best of N” 来改进输出

Best-of-N 功能允许你为单个任务同时生成多个响应，以快速探索多个解决方案并挑选最佳的那个。对于更复杂的任务，你可以审阅多个迭代并组合不同响应的部分以获得更强的结果。

前瞻

Codex 仍处于研究预览阶段，但它已经在我们的构建方式上产生了切实影响，帮助我们更快地推进、编写更好的代码，并承担那些原本不会被优先处理的工作。

我们对前景感到兴奋——随着我们的模型不断改进且 Codex 更深入地整合到我们的工作流中，我们期待着解锁用它来开发软件的更强大方式。我们将继续在此过程中分享我们的所学。

OpenAI

OpenAI

