Codex prompt cookbook

Explore and create

OUTCOME + CONTEXT + BOUNDARIES + A RESULT YOU CAN CHECK

Attach the relevant material. Replace every bracketed field before sending.

01 Understand an unfamiliar project

Get your bearings before choosing what to change.

Explore [project folder] and explain how [feature or request] works. Identify the entry points, key files, data flow and relevant checks, with file references I can follow. Read the project instructions. Do not edit files yet. Flag anything you could not inspect and finish with the next useful investigation.

Check: Follow two file references and confirm the described flow.

02 Research a decision

Compare current options using evidence you can revisit.

Research [decision] for [audience and use case]. Compare [options] on [criteria] within [budget or constraints]. Use current primary sources and link the facts that affect the decision. Separate published facts from your analysis. Deliver a short recommendation, a comparison table and unresolved questions. Do not purchase or contact vendors.

Check: Open the decisive sources and check dates, terms and assumptions.

03 Create a finished document

Turn supplied material into a file for a specific reader.

Use [source files] to create [document type] for [audience]. Lead with [main decision or message], keep it to [length], and match [style reference]. Preserve the supplied figures and cite their sources. Flag gaps instead of inventing details. Save the finished file in [folder], inspect its rendered layout and report any unresolved problems.

Check: Check the source figures and inspect the finished file.

Original Kingy.ai examples. Adapt the scope; tool access and permissions still apply.

Read the full illustrated guide on Kingy.ai

OpenAI prompting Codex best practices Checked Oct 16, 2026

Companions: Quick start Advanced Troubleshooting Prompts Shortcuts

Codex prompt cookbook

Build and repair

OUTCOME + CONTEXT + BOUNDARIES + A RESULT YOU CAN CHECK

Name the behavior, reproduction steps and constraints that matter.

04 Fix a reproducible bug

You can describe what fails and what should happen.

Fix [bug] in [project]. Reproduce it using [steps]; expected behavior is [expected], but I see [actual]. Inspect the relevant code and explain the cause briefly. Make a focused fix while preserving [API or behavior constraint]. Add a regression test if useful, rerun the reproduction and relevant checks, and report commands, results and remaining gaps.

Check: Repeat the original reproduction after the fix.

05 Implement a scoped feature

You know the intended behavior and acceptance checks.

Implement [feature] in [project or route]. It should [user-visible behavior] and handle [important edge cases]. Follow existing patterns and preserve [contracts or files]. Done means [acceptance checks]. Make reasonable implementation choices, run the relevant checks and show the working result. Report what changed and what you could not verify. Keep deployment outside this task.

Check: Exercise the main path and the named edge cases.

06 Refactor without changing behavior

Improve structure while keeping the public contract stable.

Refactor [module] to [specific structural goal]. Preserve [public API, outputs and compatibility]. Read the callers and existing tests before editing. Keep unrelated work intact and avoid broad cleanup. Verify the preserved behavior with the relevant checks, inspect the final diff and explain any compatibility risk or verification gap.

Check: Compare the public contract and behavior before and after.

Original Kingy.ai examples. Adapt the scope; tool access and permissions still apply.

Read the full illustrated guide on Kingy.ai

OpenAI prompting Codex best practices Checked Oct 16, 2026

Companions: Quick start Advanced Troubleshooting Prompts Shortcuts

Codex prompt cookbook

Check and improve

OUTCOME + CONTEXT + BOUNDARIES + A RESULT YOU CAN CHECK

Choose a comparison or success condition before asking for a review.

07 Review a change for real defects

Inspect a diff before you accept or share it.

Review [working changes, commit or PR] against [base or intended behavior]. Focus on correctness, regressions, data handling and missing coverage. For each actionable finding, give severity, file and line, trigger and impact. Separate confirmed defects from questions. Do not edit files. If you find no actionable defects, say so and list the checks you could not perform.

Check: Confirm each finding against the current diff and trigger.

08 Add tests that catch a failure

A behavior needs protection against a meaningful regression.

Add focused tests for [behavior] in [file or module], following this project's conventions. Cover [normal case], [edge case] and [failure case]. Test observable behavior rather than copying implementation details. Run the relevant suite and explain which regression each test would catch. Do not alter production behavior merely to make a failing expectation pass.

Check: Check that the tests exercise the stated behavior and edge cases.

09 Verify a browser journey

Check a real interface, including its mobile layout.

Verify [journey] at [URL] on desktop and mobile. Use [safe test data] and follow [steps]. Check the expected result, keyboard access, errors, visual layout and horizontal overflow at [widths]. Report observations and retain useful evidence. Do not send real messages, submit payments or change live records; stop before any such step and describe what remains untested.

Check: Match the evidence to the stated widths and expected result.

To revise: Keep [what works]. Change [specific issue]. Recheck [success condition].

Read the full illustrated guide on Kingy.ai

OpenAI prompting Codex best practices Checked Oct 16, 2026

Companions: Quick start Advanced Troubleshooting Prompts Shortcuts

Codex prompt cookbook

Coordinate and reuse

OUTCOME + CONTEXT + BOUNDARIES + A RESULT YOU CAN CHECK

Use advanced tools only when available and useful for the task.

10 Coordinate a larger implementation

Independent work can help a clearly defined change.

Deliver [change] in [repository] with [acceptance checks]. Inspect the starting state and preserve existing edits. Plan the dependencies, then implement. Where tools support it, use scoped subagents for independent work and separate worktrees for parallel editing. Give each lane clear ownership, integrate once, and verify the combined result. Report delivery state and remaining risks; do not deploy.

Check: Verify the integrated result, not only each lane's output.

11 Package a workflow as a skill

A task has worked manually and you need to repeat it.

Create a local skill for [repeatable job] in [skill folder]. Use [successful workflow] as the reference. Define when to use it, required inputs, steps, outputs, checks and failure handling. Keep the instructions concise and add scripts only where they improve reliability. Try [representative task], report the result and leave account connections, scheduling and publication outside this task.

Check: Try a realistic input and inspect the output and failure behavior.

12 Prepare a reliable continuation

Another session needs to continue from the actual state.

Prepare a continuation note for [task] in [file]. Record the objective, decisions, project path, branch or worktree, changed files, completed checks, failures, external actions and current delivery state. Link the supporting evidence and list the next concrete steps. Distinguish verified facts from assumptions. Keep secrets out. Do not perform new external writes while preparing the note.

Check: Confirm the note matches current files and external state.

Original Kingy.ai examples. Adapt the scope; tool access and permissions still apply.

Read the full illustrated guide on Kingy.ai

OpenAI prompting Codex best practices Checked Oct 16, 2026

Companions: Quick start Advanced Troubleshooting Prompts Shortcuts