THE BEGINNER SERIES / 02

YOUR FIRST BUILD.

A CODEX STARTER GUIDE

From an empty folder to your first working project.

ONE GUIDE. ONE REAL PROJECT.

PLAN / ORGANISE / BUILD / TEST / SHARE

A focus dashboard made with HTML, CSS and JavaScript.

START HERE

What is Codex?

Your AI coding collaborator. You describe the outcome, review the changes and test the result.

IN SIMPLE WORDS

Codex can read a project, write or edit files, explain code and help fix bugs. It is useful even if you have never written code before.

WHAT YOU DO

Choose an idea, explain what you want, approve important changes, test in a browser and decide what to improve next.

WHAT CODEX DOES

Suggest a plan, create and edit files in the selected workspace, investigate errors and describe what changed.

KEEP CONTROL

Codex can make mistakes. Read file changes before accepting them and never paste passwords, tokens or private customer data into prompts.

- A small website is a brilliant first build.
- You do not need to learn everything before you begin.
- Your first version should be simple, testable and easy to change.

PREPARATION

Open your workspace

This guide focuses on the Codex desktop experience. Labels may vary across app versions.

01 / OPEN

Open the Codex app or an available Codex interface and sign in.

02 / CREATE

Make a folder on your computer called mini-focus, in a location you can find again.

03 / SELECT

Open that folder as your project or workspace in Codex.

04 / CHECK

Ask Codex which folder it can access before you start changing files.

YOUR FIRST MESSAGE

Tell me which project folder is open. Do not create or change files yet. Explain in simple terms what you can access.

NO EXTRA SERVICES YET

The project uses plain browser technology. You do not need an API key, account database or paid hosting to start.

PROJECT ANATOMY

Meet your project files

Treat your folder like a neatly organised desk. Every file has a job.

RECOMMENDED STARTING STRUCTURE

```
mini-focus/
  README.md        project overview
  AGENTS.md        guidance for Codex
  index.html       structure and content
  style.css        colours and layout
  script.js        timer and task behaviour
  assets/          future images and icons
```

- index.html tells the browser what appears on the page.
- style.css controls how the page looks.
- script.js makes the timer and task buttons work.
- README.md explains the project to a human reader.
- AGENTS.md gives reusable instructions to Codex.

IMPORTANT

README.md and AGENTS.md are both Markdown files. Neither is required for a browser to run your project.

DOCUMENTATION

README.md vs AGENTS.md

Two small files that make it much easier to return to a project later.

README.md / FOR PEOPLE

Record what the project does, its key features, how to run it and what you want to build next.

SIMPLE README EXAMPLE

```
# Mini Focus
# A tiny focus dashboard.
## Features
- 25-minute focus timer
- Task list and completed count
## Run
Open index.html in a browser.
```

AGENTS.md / FOR CODEX

Keep project-specific coding rules short and useful. Codex can use AGENTS.md as repository instructions when present.

SIMPLE AGENTS EXAMPLE

```
# Project instructions
Use HTML, CSS and vanilla JavaScript.
Do not introduce frameworks or packages.
Keep files small and readable.
Preserve existing working features.
Explain how to test every change.
```

GOOD HABIT

Ask Codex to update README.md when functionality changes. Keep AGENTS.md focused on lasting rules, not temporary tasks.

PROJECT BRIEF

Define the mini build

Our first project is a focus dashboard. The aim is a useful, finished mini website.

THE IDEA

A dark, modern page with a 25-minute focus timer, start/pause/reset buttons, a task list and a completed-task counter.

THE RULES

Use only HTML, CSS and JavaScript. Work on desktop and mobile. Store tasks locally in the browser. Do not use external APIs.

NOT IN VERSION ONE

No user accounts, no cloud syncing, no subscriptions and no complicated dashboard charts. These can wait.

PROMPT 01 / PLAN BEFORE CODING

I want to build a beginner-friendly Mini Focus Dashboard with a timer, task list and completed counter. Use plain HTML, CSS and JavaScript. First propose a simple file structure and 5 build steps. Do not edit files until I approve the plan.

CHECKPOINT

You should receive a plan you understand. If a step seems confusing, ask Codex to explain it before moving on.

PROJECT SETUP

Create the foundation

Now turn your plan into real files. Keep the first version minimal.

PROMPT 02 / CREATE FILES

Create README.md, AGENTS.md, index.html, style.css, script.js and an assets folder. Follow the agreed plan. Build a basic accessible page with a heading, timer display, controls, task input and task list. Do not add functionality yet. Show me exactly which files you created.

WHAT TO LOOK FOR

Your folder should contain the files listed above. The HTML should reference style.css and script.js so the browser loads them.

THE IMPORTANT CONNECTIONS

```
<link rel="stylesheet" href="style.css">
<script src="script.js" defer></script>
```

PROMPT 03 / EXPLAIN MY FILES

Walk me through each new file as if I am completely new to coding. Tell me what I can safely edit and which lines connect the files.

CHECKPOINT

Open index.html in a browser. You should see the page, even though the buttons do not work yet.

DESIGN PHASE

Make it look like a product

This is where a basic web page becomes something you would actually want to use.

PROMPT 04 / DESIGN

Style the dashboard with a near-black background, icy cyan highlights, clean typography, subtle borders and comfortable spacing. Make it responsive for phones. Use CSS only, no external UI libraries. Preserve the existing HTML structure where possible.

VISUAL PRIORITIES

Make the remaining time the largest element. Place start, pause and reset below it. Keep the task input obvious. Show the completed count without competing with the timer.

EXAMPLE PALETTE

Background	#070A0F
Panel	#101820
Accent	#77DBFC
Text	#F4F8FC
Muted text	#9CABBA

PROMPT 05 / RESPONSIVENESS

Check the design at approximately 375px phone width and 1280px desktop width. Fix horizontal overflow and cramped controls. Explain what you changed.

CHECKPOINT

Resize the browser. All controls should remain visible without sideways scrolling.

FUNCTIONALITY

Build the focus timer

One clear feature at a time is easier to test and fix.

PROMPT 06 / TIMER

Implement a 25-minute countdown in script.js. Add Start, Pause and Reset controls. Prevent multiple timers from running at once. Keep the displayed time in mm:ss format and stop at 00:00. Change only what is necessary and explain how I can test it.

- Start should count down once per second.
- Pause should stop without losing the remaining time.
- Reset should restore 25:00 and stop the timer.
- Clicking Start twice should not speed up the clock.
- The timer must not display negative time.

PROMPT 07 / TEST THE TIMER

Review the countdown logic for duplicate intervals, reset errors and negative values. Fix any issues, then give me a short manual test checklist.

CHECKPOINT

Run all five checks before you add tasks. A working timer is your first real milestone.

FUNCTIONALITY

Add a task list

Turn the dashboard into a simple tool for planning a focus session.

PROMPT 08 / TASKS

Add a task input and Add button. Allow tasks to be marked complete, unmarked and deleted. Reject empty or whitespace-only tasks. Update the completed-task counter automatically. Keep the UI accessible and easy to use on mobile.

- Add a task by typing and pressing Enter or Add.
- Mark it complete and confirm the counter increases.
- Unmark it and confirm the counter decreases.
- Delete a task and check the list updates.
- Try submitting a blank task. It should not be added.

PROMPT 09 / POLISH

Improve task interactions with clear button labels, keyboard support, visible focus styles and a helpful empty-list message. Do not change the timer behaviour.

CHECKPOINT

Your dashboard now has two working features: a timer and a task list.

SAVE PROGRESS

Keep tasks after refresh

Your browser can remember basic information without a server.

WHAT IS LOCAL STORAGE?

localStorage is a browser feature that stores small amounts of data on the same device and browser. It is suitable for this simple task list.

PROMPT 10 / SAVE TASKS

Use localStorage to save tasks, completion status and the completed count derived from the tasks. Restore them when the page reloads. Handle missing or invalid saved data safely. Do not use a backend or external service.

- Add three tasks.
- Mark one complete.
- Refresh the page.
- Check that the tasks and completion states remain.
- Delete one and refresh again.

KNOW THE LIMIT

Browser storage is not a cloud backup. Clearing site data or changing browser or device can remove it. Do not store sensitive information.

TROUBLESHOOTING

When something breaks

An error is information. Give Codex useful detail instead of writing only “fix it”.

WEAK PROMPT

It does not work.

BETTER PROMPT / SPECIFIC BUG

The Start button responds, but the timer counts down twice as fast after I click Start two times. Expected: one second per tick. Actual: two seconds per tick. Please inspect script.js, identify the cause, make the smallest safe fix and explain how to retest.

IF THE PAGE IS BLANK

Tell Codex what browser you are using, which file you opened and what you see. Ask it to check file paths and browser console errors.

IF STYLING DISAPPEARS

Ask Codex to check the stylesheet link, file names and any recent changes to CSS selectors.

IMPORTANT

Never blindly approve a large rewrite for a tiny bug. Ask for the smallest fix and test the existing features again.

WORKING SMARTER

The repeatable Codex loop

Use the same small cycle for every new feature, not one giant prompt for everything.

01 DESCRIBE

Say what should happen and what should stay unchanged.

02 LET IT PLAN

Ask for a short approach before major changes.

03 REVIEW

Inspect the files or diff. Look for unexpected edits.

04 TEST

Try the happy path, an edge case and a failure case.

05 RECORD

Update README.md and commit a working checkpoint.

REUSABLE CHANGE PROMPT

Add [FEATURE]. Preserve [EXISTING FEATURES]. Keep changes minimal. Explain which files you changed, why, and how to test the result.

QUALITY CONTROL

Test like a beginner pro

Use a simple test sheet before calling the project finished.

01	Timer starts, pauses and resets correctly.
02	Repeated Start clicks do not create a faster timer.
03	Tasks can be added, completed, unmarked and deleted.
04	Blank tasks do not get added.
05	The counter always matches completed tasks.
06	Tasks survive a page refresh in the same browser.
07	All buttons work by keyboard and have readable labels.
08	The layout works at phone and desktop sizes.

PROMPT 11 / FINAL AUDIT

Review the project against this checklist. Report any failures by severity. Fix only reproducible problems. Update README.md with the features and instructions to run locally.

VERSION CONTROL

Save your milestones with Git

Git is a history of code changes. GitHub can host a remote copy of that history.

WHY BOTHER?

A commit is a named snapshot of your project. If an experiment goes wrong, Git makes it easier to understand and recover from changes.

OPTIONAL COMMAND-LINE BASICS

```
git init
git add .
git commit -m "Build first focus dashboard"
git status
```

PROMPT 12 / HELP WITH GIT

Explain whether Git is already set up in this folder. If not, guide me through creating my first commit. Explain every command before running it and ask before connecting to any remote service.

SAFETY FIRST

Check that no passwords, API keys, personal data or large private files are being tracked. Never commit secrets. Add a .gitignore file when relevant.

AFTER YOU CHANGE A FEATURE

Test the app, review the changes and make another descriptive commit. Small commits are easier to understand than one huge final commit.

SHARING YOUR SITE

Publish when you are ready

A static HTML/CSS/JS project can be hosted without running a server yourself.

OPTION A / GITHUB PAGES

A common choice for static sites. Create or connect a GitHub repository, configure Pages for the correct branch and folder, then open the public URL.

OPTION B / STATIC HOST

Services such as Netlify or Cloudflare Pages may host a static folder or a connected repository. Check each provider's current setup and pricing.

PROMPT 13 / DEPLOYMENT PLAN

My project is a static website with index.html, style.css and script.js. Explain the simplest current way to publish it with GitHub Pages. List exactly what I should verify before making it public. Do not publish anything without my approval.

PUBLIC MEANS PUBLIC

Review the code and README for private information before uploading. localStorage data on your computer is not automatically published with your website.

FINAL CHECK

Open the public website on another device, test the timer and tasks and make sure links and styles load correctly.

PROMPT LIBRARY

Copy these anytime

Four small prompts you can keep using after this guide.

START A NEW PROJECT

Help me turn [IDEA] into the smallest useful version. First define the core features, simple folder structure, risks and a step-by-step plan. Do not edit files yet.

UNDERSTAND EXISTING CODE

Read the current project and explain the files, dependencies and user flow in plain English. Identify the main entry point. Do not change code.

FIX A SPECIFIC PROBLEM

Expected: [EXPECTED]. Actual: [ACTUAL]. Steps to reproduce: [STEPS]. Find the root cause, make the smallest safe fix and give me a retest checklist.

END A WORK SESSION

Summarise what we changed, what works, what still needs testing and the next 3 steps. Update README.md only if it is now inaccurate.

THE TRICK

Good prompts include the goal, context, constraints and how success will be checked. Fancy wording is optional.

FINISH LINE

You built something.

You now have a repeatable method, not just a single web page.

YOUR FINAL CHECKLIST

A project folder. A useful README.md. Clear Codex guidance. A functioning timer. A task list. Browser persistence. Manual tests. A saved milestone.

YOUR NEXT CHALLENGE

Try one carefully scoped addition: a choice of 25- and 50-minute sessions, a small sound when the timer finishes, or a daily task reset. Add only one at a time.

NEXT-BUILD PROMPT

Suggest 5 small improvements to my Mini Focus Dashboard. Rank them by usefulness and complexity. Recommend one beginner-friendly change, plan it first and keep existing features intact.

ONE LAST REMINDER

Review code changes, test in the browser and keep backups. Codex is a collaborator, not a replacement for your judgement.

BUILD SMALL. TEST OFTEN. MAKE IT YOURS.

PAL / PIXELS DESIGN x CODE x CURIOSITY

REFERENCE / OpenAI Codex documentation:

developers.openai.com | Codex quickstart and AGENTS.md guidance