Build Beautifully

buildbeautifully.lovable.app

Session 1 Companion · Keep this open. Refer back anytime.

If you get lost or fall behind — just ask Claude directly.

Claude always knows where you are in the course. These three phrases will get you back on track instantly:

"Claude, where am I in the course?" *"Catch me up to [lesson or project name]."* *"Explain that in plain English."*

TODAY'S MAP

Phase	What you're doing	~Time
Module 0	Seven mini-lessons in the terminal — type each `/lesson` command when prompted	~30 min
Project 1	Building your first app! Deploy it live.	~35 min
Project 2	Build YOUR animated career journey landing page — your story, deployed to a real URL.	~35 min

HOW TO START A CLAUDE CODE SESSION IN A FOLDER

Claude Code always runs *inside* a folder. The folder you start it in is where it creates files and reads your `CLAUDE.md`. There are two ways to get there:

MAC — OPTION A: RIGHT-CLICK

Open **Finder**, navigate to your project folder, then **right-click the folder → "New Terminal at Folder."**

The terminal opens already inside that folder. Then just type:

```
claude
```

Don't see that option? Go to System Settings → Privacy & Security → Developer Tools and enable Terminal.

MAC — OPTION B: CD COMMAND

Open Terminal, then type `cd` followed by the path to your folder:

```
cd ~/build-beautifully
claude
```

How to find the path: drag your folder from Finder into the Terminal window after typing `cd` — the full path auto-fills. Or right-click the folder and hold Option — you'll see "Copy [folder] as Pathname."

WINDOWS — OPTION A: RIGHT-CLICK

Open **File Explorer**, navigate to your project folder, then **right-click inside the folder (on empty space) → "Open in Terminal."**

The terminal opens already inside that folder. Then type:

```
claude
```

If you see PowerShell instead of Command Prompt, that's fine — both work.

WINDOWS — OPTION B: CD COMMAND

Open Terminal or PowerShell, then type `cd` followed by the path:

```
cd C:\Users\YourName\build-beautifully
claude
```

How to find the path: in File Explorer, click the address bar at the top — the full path appears. Copy it, then paste it after `cd`.

Once you're in the right folder and Claude Code is running, you're ready. Claude will read your `CLAUDE.md` automatically and know what today's session is about.

LESSON-BY-LESSON REFERENCE

LESSON 1

Welcome & Your First File

`/lesson-1-welcome`

- You type a sentence — Claude builds a real HTML file. That's the whole model.
- Files Claude creates live in your **current folder** — visible in Finder or File Explorer like any normal file.
- To open a file: double-click it in Finder/File Explorer, *or* open a second terminal tab and type `open filename.html` (Mac) / `start filename.html` (Windows).
- The full file path is always shown — use it if double-click doesn't work.

LESSON 2

The CLAUDE.md File

`/lesson-2-claude-md`

- Claude reads `CLAUDE.md` every time you start it in that folder — it's how it already knows what you're working on.
- **Keep it short.** Store stable facts: tools you use, colors, how things are structured.
- **True-over-time** (belongs here): "We use Next.js and Tailwind." **Right-now stuff** (belongs in chat): "Currently fixing the header bug."
- Create one instantly: type `/init` or ask Claude in plain English.
- Outdated `CLAUDE.md` is worse than none — update it when things fundamentally change.

LESSON 3

Slash Commands

`/lesson-3-slash-commands`

- Slash commands are shortcuts you type *into Claude Code*, not into the terminal window.
- **Built-in commands:**
  - `/help` — full list of available commands
  - `/clear` — fresh start (files stay, conversation memory resets)
  - `/model` — switch Claude models
  - `/context` — check how full Claude's memory is
  - `/agents` — see your subagents
- **Custom commands** (like `/lesson-1-welcome`) are files in `.claude/commands/` — anyone can write their own.

LESSON 4

Plan Mode

`/lesson-4-plan-mode`

- Press **Shift + Tab** to toggle plan mode on/off. You'll see "plan mode on" in the corner.
- In plan mode: Claude can *think and propose*, but **cannot write files or run commands.**
- Use it before starting anything new — read the plan, approve it, *then* build.
- **The 95% rule:** type *"Use the AskUserQuestion tool to ask me clarifying questions until you're 95% sure of what I want."*
- Bad output is almost always a **clarity problem**, not a skill problem.

LESSON 5

Plan → Build → Iterate

`/lesson-5-plan-build-iterate`

- **PLAN** — agree on what you want before code gets written.
- **BUILD** — Claude writes the code. Usually fast: minutes, not hours.
- **ITERATE** — look at the result, say what to change, repeat. Don't accept the first version.
- You can drag a screenshot into the terminal and tell Claude what to change.
- **YOLO mode** (`--dangerously-skip-permissions`): skip permission checks. Fine in a sandbox. **Never** on anything live or important.

LESSON 6

Skills & Plugins

`/lesson-6-skills-and-plugins`

- A **skill** is a packaged Claude expert for one specific task. Think: a spell.
- A **plugin** is a bundle of related skills + commands. Think: a spellbook.
- Slash commands = *you* trigger them manually. Skills = Claude triggers them automatically when it detects you need one.
- Where to get skills: build your own with `/skill`, download from GitHub, or get the Build Beautifully curated list.

LESSON 7

Agents & Subagents

`/lesson-7-subagents`

- **Agent** = the Claude you're talking to right now. It's the brain — decides what to do, what to build, when to call in specialists.
- **Subagent** = a specialist worker Claude sends out for one focused job. Faster and cheaper because it only knows what it needs.
- **Parallel subagents**: same job done across many things at once (e.g. 10 job descriptions processed simultaneously).
- **Specialized subagent**: one expert for one rich task (e.g. `resume-parser` — reads your PDF and turns it into structured data).
- See your agents with `/agents`. Create a new one by asking Claude in plain English.

COMMON TROUBLESHOOTING

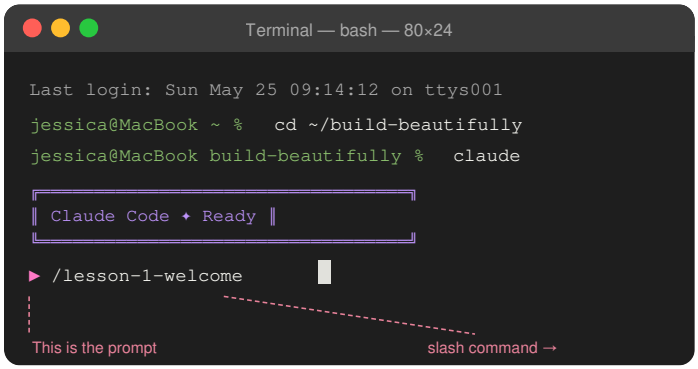
What's happening	Fix
"Claude doesn't seem to understand the commands being shown on screen and doesn't seem to know what to do"	Claude Code probably isn't operating in the right folder. Did you `cd` into your project folder *before* running `claude`? If not, quit, navigate to the right folder, and restart.
File won't open in browser	Ask Claude: *"Give me the exact terminal command to open my [filename] file."* It'll give you the full path — no guessing.
Claude feels forgetful or confused	Type `/context`. If it's over 40%, ask Claude to summarize what you've decided and what's left to do. Then type `/clear`, open a fresh conversation, and paste that summary back in. Your files are safe — only the conversation memory resets.
Something just broke / error message	Tell Claude exactly what happened in plain English — copy and paste the error if you can. It can fix almost anything if you show it what went wrong.
You're behind the rest of the room	Type: *"Catch me up to [lesson or project name]."* Claude will fast-forward you to the right point and keep going.
Terminal commands not working on Windows	Mac uses `open filename.html` to open files; Windows uses `start filename.html`. Everything else is the same.

GLOSSARY

Word	Plain English
`Terminal`	The window where you type commands. See the illustration →
`cd`	"Change directory." How you navigate into a folder in the terminal.
`npm run dev`	Starts a local preview of your site at `localhost:3000`.
`localhost:3000`	A preview URL that only works on *your* computer. Not shareable until you deploy.
`GitHub`	Cloud storage for your code — a folder in the sky that remembers every version.
`Vercel`	Takes your code from GitHub and turns it into a live website anyone can visit.
`Repo`	Short for "repository." Your project's home on GitHub.
`Deploy`	Putting your site on the internet so anyone in the world can open it.
`Commit`	Saving a named snapshot of your code to GitHub.
`Push`	Uploading your latest commits from your laptop to GitHub.

WHAT IS THE TERMINAL?

The terminal is a text window where you give your computer direct instructions by typing commands. On a Mac it's called **Terminal**. On Windows it's **Command Prompt** or **PowerShell**. This is where you'll run `claude`.



The green text is your prompt (shows your name + folder). The white text is what you type. The blinking block is your cursor.