How to Make Vibe Coding Actually Work

Hidden Prerequisites, Safer Paths, and Practical Wins

Seth J. Chandler

For Law Students

2025

Today's Journey

- Why vibe coding matters now
- What vibe coding really is
- Ten hidden assumptions that derail beginners
- Guardrails to keep you moving
- A practical path forward
- The broader ecosystem of tools

Why This Matters Now

- Vibe coding = describing what you want to an LLM and iterating
- Gentle on-ramp to computing
- Copy short programs, run them, get results in minutes
- Goal: acquire just enough skill to turn ideas into working tools
- Not about becoming a software engineer

A Connected Path

1. Hidden assumptions that trip up beginners
2. A set of practical guardrails
3. Concrete Week-1 plan
4. Sustainable habits for the long term

What Vibe Coding Is

- **NOT** "magic code without learning"
- A cooperative workflow:
  - You state the outcome
  - LLM proposes code
  - You run it and observe what fails
  - Refine the prompt or code
- Power comes from rapid feedback
- Realistic for absolute beginners with guardrails

What Vibe Coding Is Not

- Not a way to skip all prerequisites
- Not guaranteed to work on first try
- Not a replacement for understanding basics
- Not magic—requires iteration and learning

Ten Hidden Assumptions

The frustration most newcomers feel comes from silent prerequisites.

None is advanced; together they cascade.

Hidden Assumption #1: Terminal Fluency

- Without `cd`, `ls/dir`, `pwd`, nothing starts
- Learn ten most common terminal commands
- Understand relative vs. absolute paths
- Know how to move file names into terminal
- Use arrow keys and tab completion

Hidden Assumption #2: Python Dependencies

- Standard `pip` leads to "dependency hell"
- Solution: Use Poetry or Pipenv
- ModuleNotFoundError = missing dependency
- Use `poetry show --tree` for diagnosis
- Isolated environments with lockfiles

Hidden Assumptions #3-4

Project Layout & Imports

- Files in wrong place cause "module not found"

$PATH Configuration

- Shell must find executables
- Commands fail without proper PATH
- Learn PATH basics for your OS

Hidden Assumptions #5-6

Runtime Selection

- Language must be installed
- Correct version matters

Isolation (Virtual Environments)

- Global installs cause conflicts
- Use Poetry or uv for Python
- Avoid permissions errors

Hidden Assumption #7: Error Reading

- Stack traces have structure
- First actionable line usually near bottom
- Give error messages to LLM
- Use cut-and-paste or screen capture
- Don't panic—errors are normal!

Hidden Assumptions #8-10

Quoting & Spaces

- Unquoted file paths break commands
- Learn drag-and-drop for your OS

Environment Boundaries

- Browser JS $\neq$ Node.js
- CLI code $\neq$ web page

Version Control

- Without Git, experiments feel risky

Guardrails That Keep You Moving

Once you know the pitfalls, adopt guardrails that make vibe coding feel coherent instead of fragile.

Guardrail #1: Start Without Frameworks

- Frameworks choose project structure for you
- Examples: Django, Flask, React, Rails
- Scaffolding creates many files automatically
- Good once you understand parts
- **Week 1 rule: One file, no build step**

Guardrail #2: Defer Docker

- Docker excellent for deployment
- Adds: images, containers, volumes, networks
- More vocabulary, more places to break
- Prefer native installs at first
- Return to Docker when you need it

Guardrail #3: Low-Friction Languages

- **JavaScript in browser:** No installs needed
- **Go:** Strong standard library, single binary
- **Python:** Excellent libraries, use Poetry

Trade-off: JS and Go may have fewer legal-specific packages, but that's OK for first projects.

Guardrail #4: Python with Poetry

Minimal Poetry workflow:

1. `pipx install poetry`
2. `poetry new myproject`
3. `poetry add requests`
4. `poetry run python main.py`

Use Poetry the moment you add any third-party package.

Guardrail #5: Learn Just Enough Git

Five-command subset:

- `git init` (start repository)
- `git add -A` and `git commit -m "message"`
- `git log` (see history)
- `git restore --source -- file` (recover)
- `git branch -c feature-x` (try ideas safely)

Benefits: easy undo, fearless experiments, backup via GitHub

Guardrail #6: Use Study & Learn Mode

- Turn friction into short, targeted lessons
- 90-second PATH explainer
- Readiness checklists for project types
- Just-in-time learning
- Preserve momentum while closing gaps

Your Week 1 Strategy

- Start with one file projects
- Choose low-friction environment
- Avoid frameworks and Docker
- Learn basic terminal commands
- Set up Git repository
- Use Poetry for Python dependencies

First Project Ideas

- HTML page with embedded JavaScript
- Python script to process CSV files
- Go program to organize files
- Simple calculator or converter
- Text analysis tool
- Basic data visualization

When Things Go Wrong

- Read the first clear error message
- Copy error to LLM for help
- Check file locations and imports
- Verify dependencies are installed
- Use Git to revert changes
- Start smaller if needed

Building Confidence

- Keep projects small
- Treat line count as risk indicator
- Celebrate small wins
- Focus on working programs you can run
- Iterate based on feedback
- Don't aim for perfection

Final Counsel

- Reduce operational complexity
- Let LLM help with what it does best
- Build sustainable habits early
- Focus on practical wins
- Confidence comes from working code

The Broader Ecosystem of Tools

There is a LOT more to learn...

And an ever-growing list of sophisticated tools

Advanced Coding Tools

AI-Powered Development:

- Claude Code
- ChatGPT Codex
- Gemini CLI
- GitHub Copilot

Enhanced Editors & IDEs:

- Cursor
- Kiro
- Warp terminal
- Visual Studio Code

Prerequisites for Advanced Tools

- Many tools presuppose decent knowledge of:
  - Visual Studio Code
  - Command line proficiency
  - Git workflows
  - Development environments
  - Package managers
- The ecosystem grows rapidly
- New tools emerge constantly
- Foundation skills remain essential
- Start simple, build up gradually

Remember: Master the basics first, then explore the advanced tools!