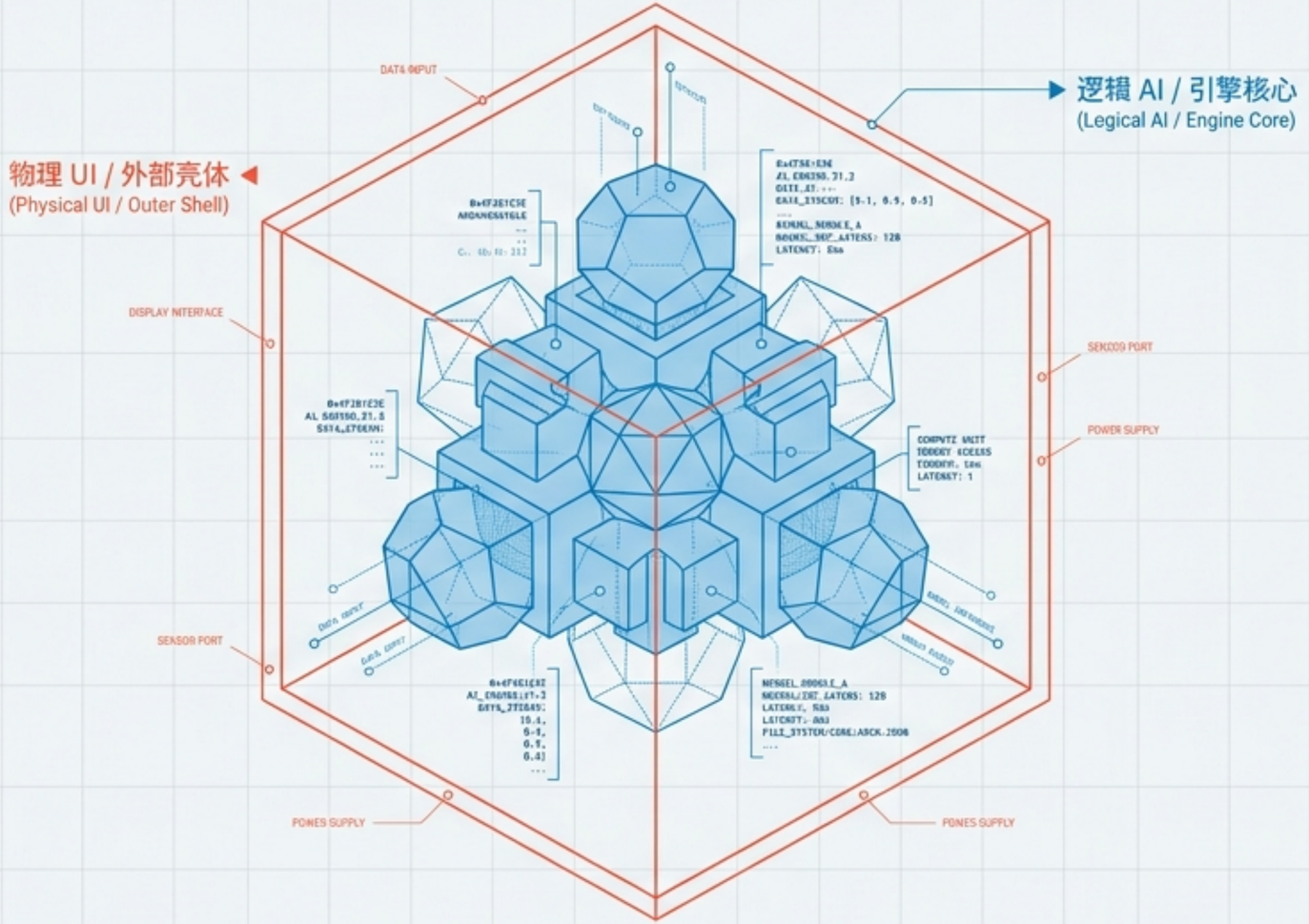
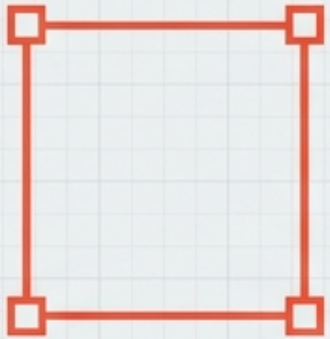
两个仓库，一个灵魂

2000行代码撑起桌面AI伴侣的极简架构实践

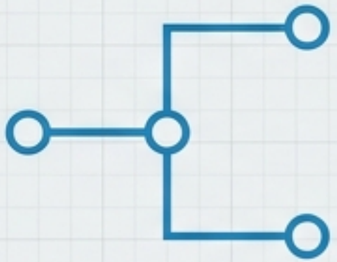


0



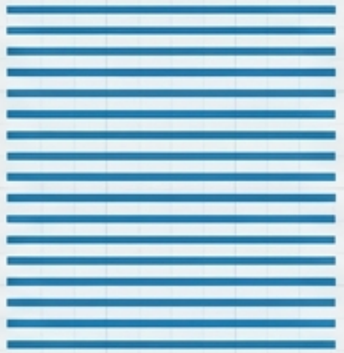
行 Electron 壳
内的 AI 代码

2



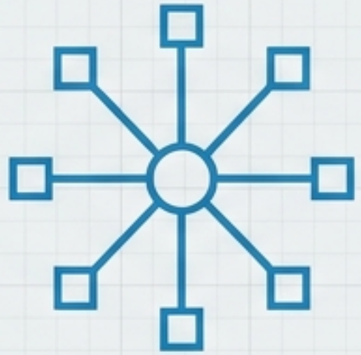
个 npm 依赖构建
完整灵魂引擎

2000



行核心代码驱动
全栈逻辑

7



个异构 AI Agent
同时接入

极简不是为了炫技，而是为了保持“角色”的绝对纯粹。

clawd-on-desk

身体 / The Shell

Tech Stack: Electron + SVG

- 像素动画（12种状态）
- 眼球追踪（20fps，3px偏移）
- 睡眠序列（60秒无操作触发）
- 权限弹窗
- 迷你模式

消耗品（Consumable）| 零 AI 逻辑

HTTP :23456

clawd-soul

大脑 / The Engine

Tech Stack: Node.js + SQLite

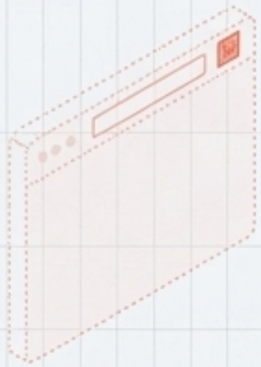
- 视觉捕捉（1920×1080 JPEG q85）
- 性格系统（5种散文原型）
- 情绪引擎
- 记忆重组（23:30 自动做梦）

持久物（Persistent）| 11个源文件

[2024]
Desktop:
Electron App

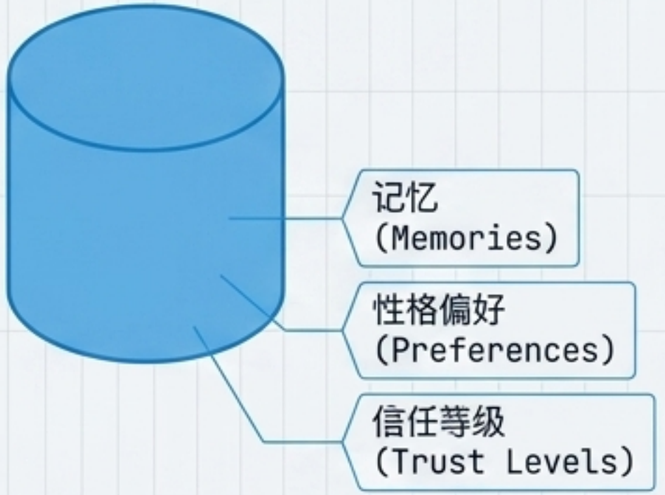
[2025]
Mobile: iOS/Android
Widget

[2026]
Web:
Browser Extension



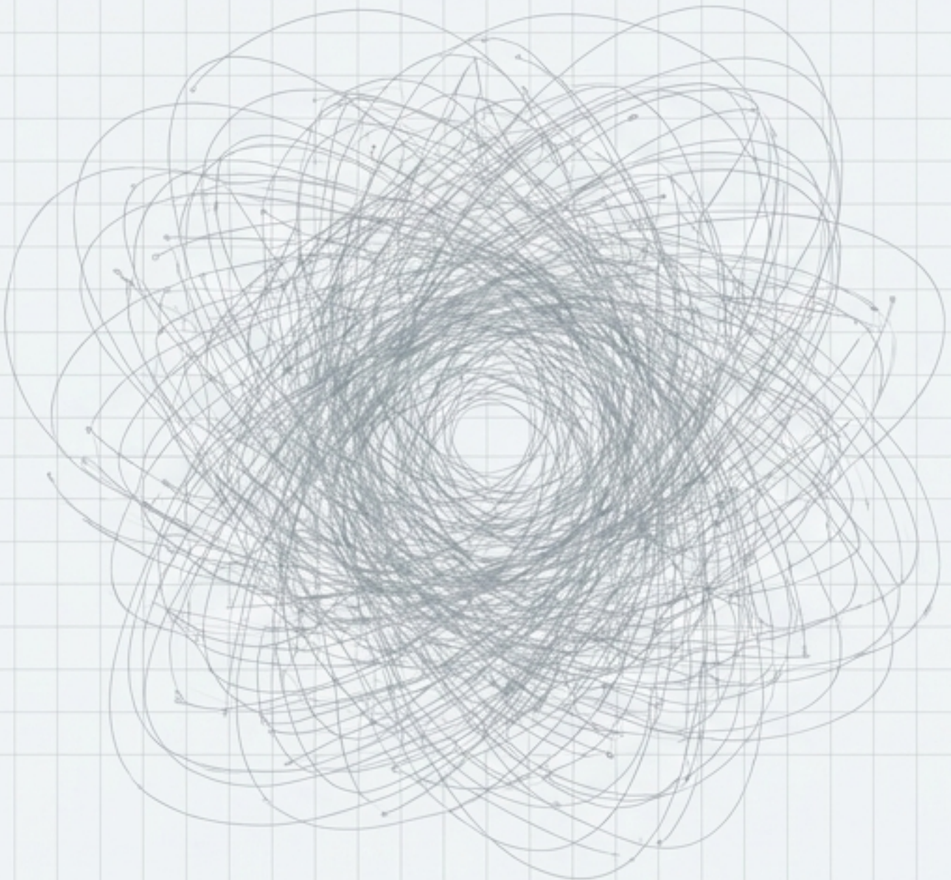
导出一个存档文件，换一台电脑导入，你的宠物还认得你。灵魂引擎是一个标准 HTTP 服务，跑在任何有 Node.js 的机器上。

Soul Engine Archive

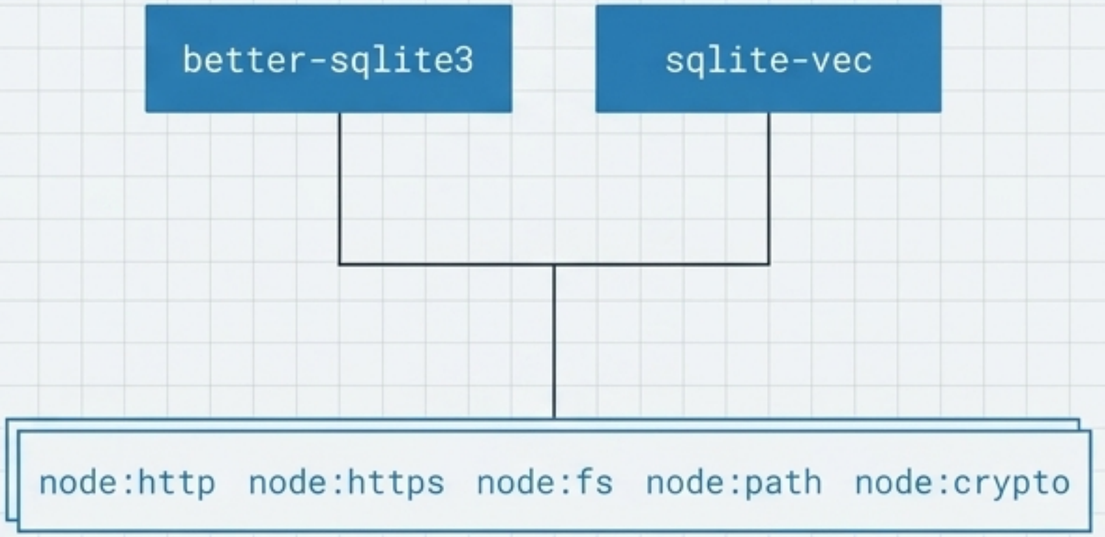


身体会过时。桌面端可能不再流行，Electron 可能被淘汰。

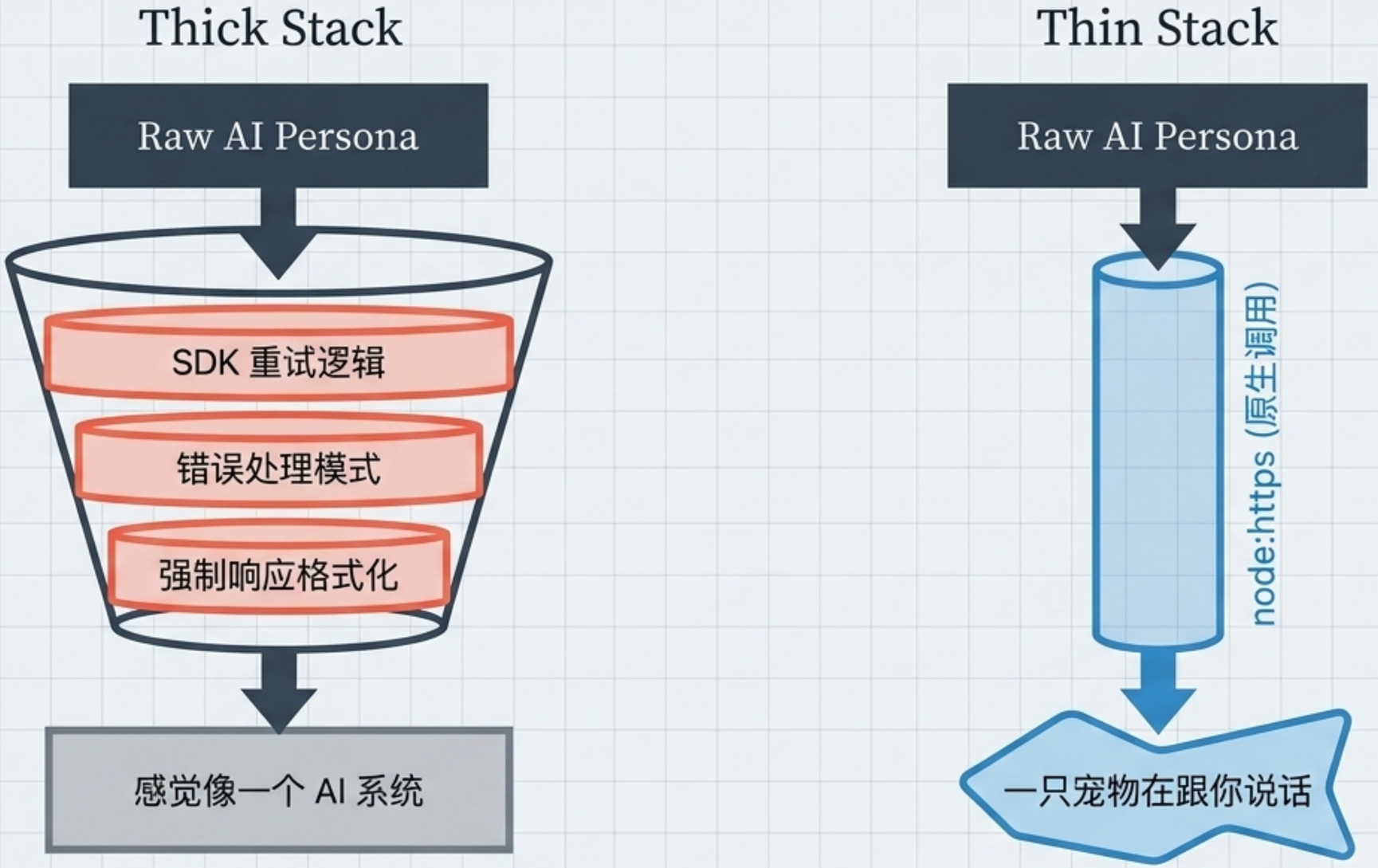
传统的 AI 封装



极简的 package.json



2 个依赖意味着只有 2 个东西能出问题。整个引擎代码一个下午就能读完。

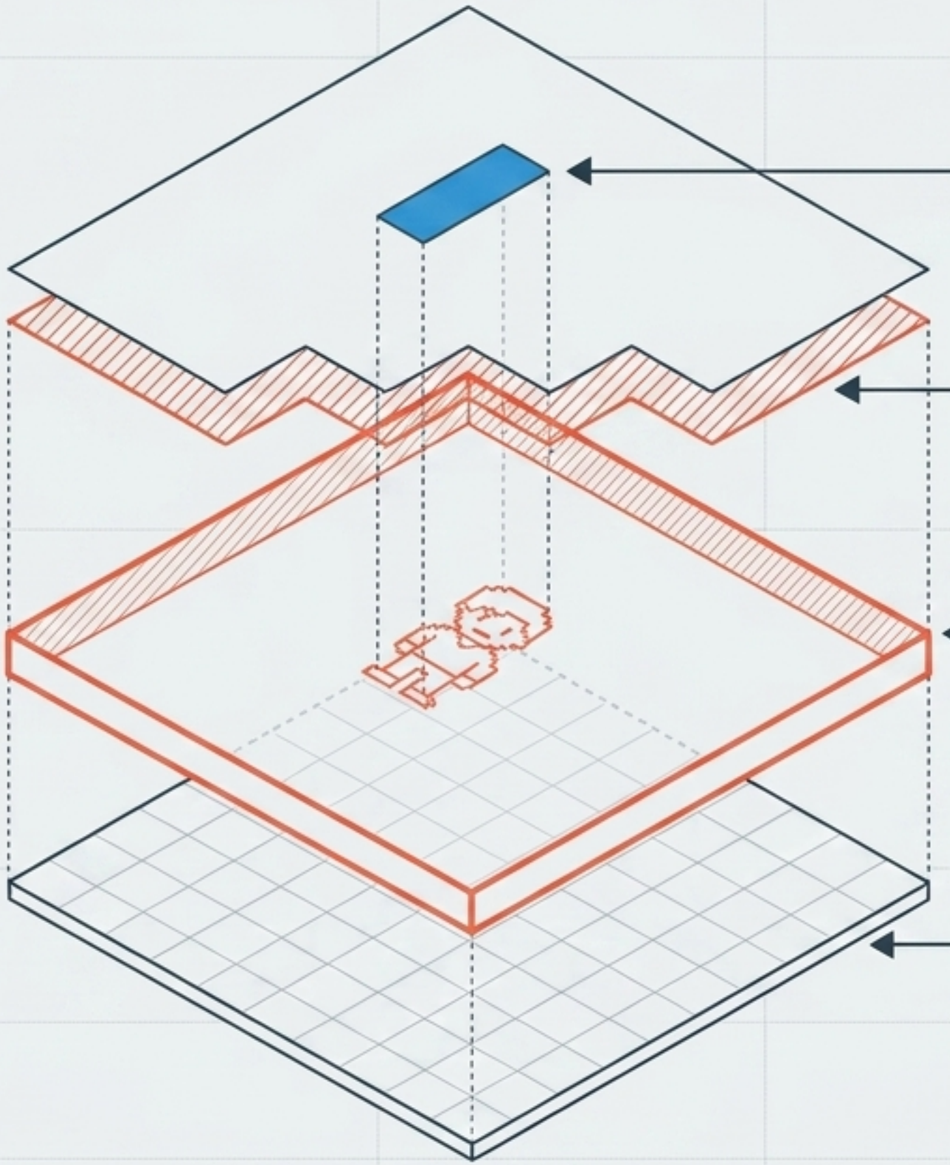


“每多一层 SDK 抽象，角色就多一个泄漏的缝隙。栈越薄，角色越纯。”

Windows 兼容性问题不是技术问题，是考古问题。靠 ALT 键 trick、koffi FFI 调用和 PowerShell 辅助进程才勉强解决。

WS_EX_NOACTIVATE

The Windows Hack X-Ray

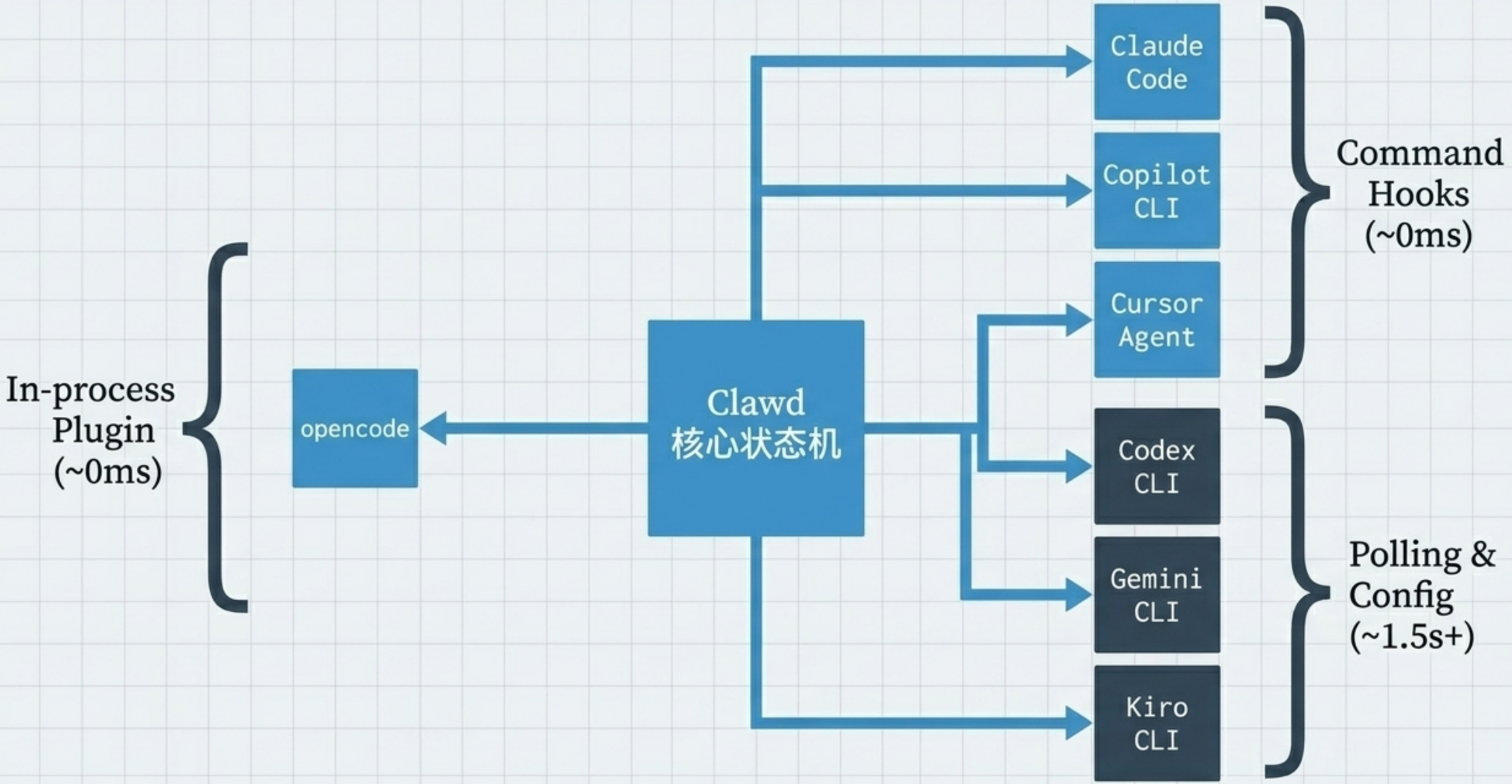


输入窗口
抢占焦点，接收指针事件

Windows 拖拽 Bug
(WS_EX_NOACTIVATE 幽灵死区)

渲染窗口
setIgnoreMouseEvents(true)

User Desktop

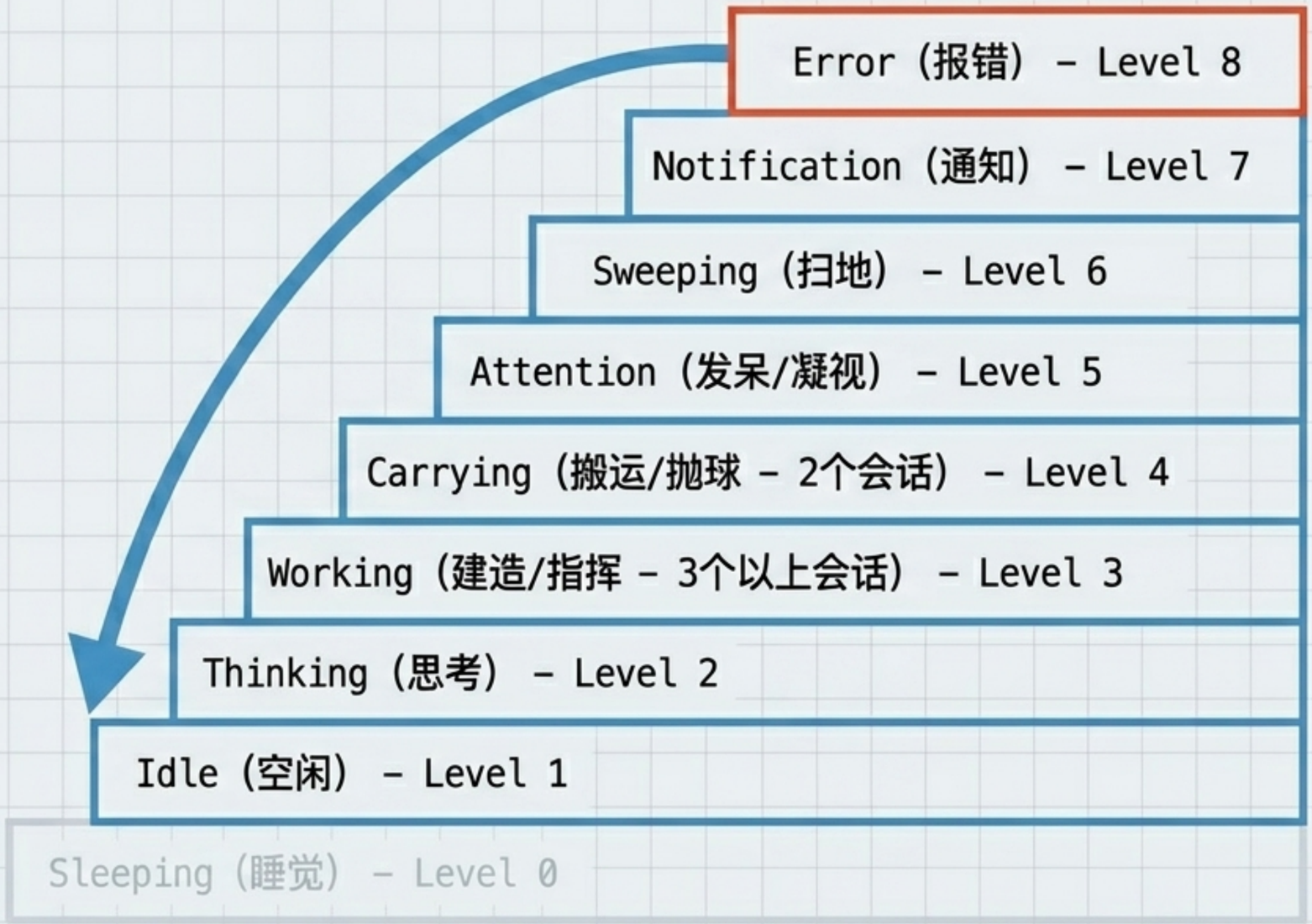


程序员桌面上同时跑多个 AI 是常态。宠物必须认识你所有的工具。

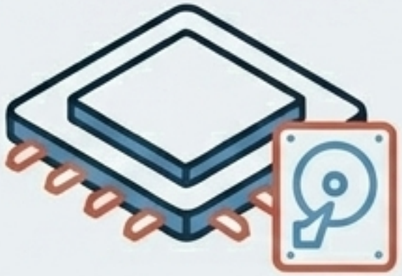
状态独立追踪。高优先级状态无缝抢占低优先级。

高优先级状态（如Error、Notification）可立即打断并抢占当前正在进行的低优先级状态。状态切换通过抢占而非等待完成实现。

Preemptive State Waterfall



System Architecture Principles

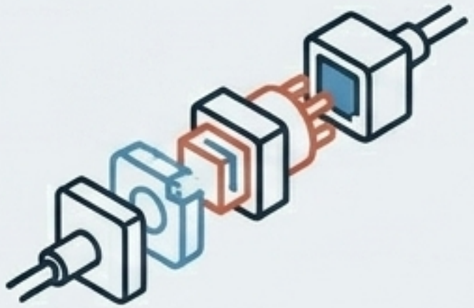


100% 本地

截屏在内存里分析完即销毁。无云端，无账号，无遥测。

MIT 协议

数据属于用户。随时自由导出/导入你的的“灵魂存档”。
20 位开源贡献者。



极低 Hack 门槛

新主题 = 1个SVG + 7个动画。
新人格 = 1份散文档案。
新Agent = 1个HTTP调用。

一个小生物住在你屏幕上，看你在干什么，记得你是谁，但不帮忙。

用最少的代码量（2000行/11个文件），做出“有灵魂的感觉”。

散文塑造性格

三层记忆与做梦机制

反常识的“无用论”

极简双仓架构