“宁愿失业，也不再用AI写一行代码！”用了18个月AI后，20年老兵宣布停用：要重获手搓代码的快乐

CSDN 2026年8月17日 14:30 江苏

整理 | 郑丽媛

出品 | CSDN（ID：CSDNnews）

如果有人告诉你，AI 能让程序员写代码快上几倍，你大概不会觉得奇怪。

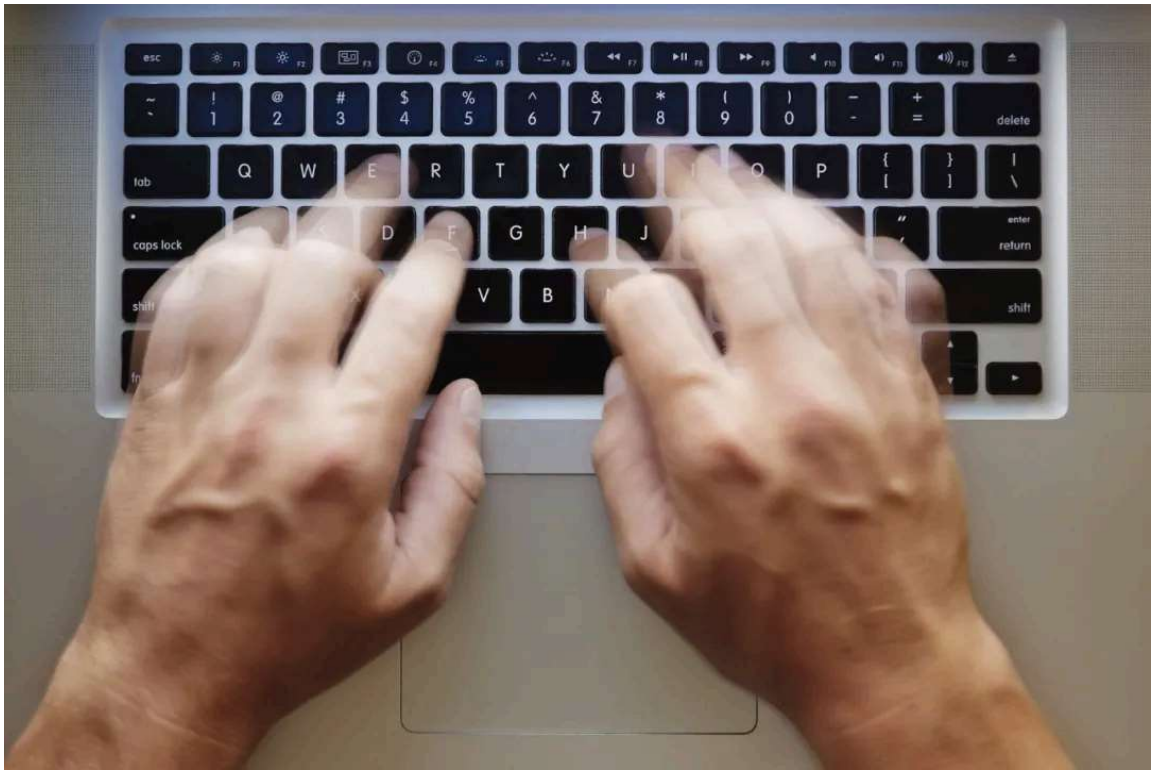
但如果一个写了 20 多年代码的老程序员告诉你：“**我用了 AI 一年多，确实写得更快了，但我越来越不喜欢编程了，所以决定彻底停用 AI。**”那这件事就值得聊聊了。

最近，一位名叫 Brett 的软件开发者分享了自己的经历。

他写代码超过 20 年，过去一年半里又长期使用 Cursor、Claude Code 等 AI 编程工具。按照很多人对 AI 编程的想象，他本应该是最大的受益者之一：任务完成得更多、代码产出得更快，甚至一度觉得自己正在以“未来的方式”工作。

然而最终，他做出了一个看起来非常反潮流的决定：停止使用 AI 编程工具，甚至停止在生活中使用 AI。原因并不是 AI 不够强，恰恰相反，正是因为它太强大了——**当 AI 越来越多地替他完成编程工作之后，Brett 开始不知道自己为什么还要写代码。**

“我知道这个决定有风险。感觉一夜之间，**行业变了，好像你不用 AI 就会被抛弃**。但我愿意承担这个风险——如果因此被解雇、找不到工作、职业生涯就此终结，那**我也认了**。”



01

AI 初体验：不实用，还会胡说八道

Brett 第一次接触 AI 编程工具，大约是在 2022～2023 年。

当时 GitHub Copilot 已经开始进入 VS Code 等开发环境。它最典型的能力，就是代码自动补全。比如你写下一个 sort_items 函数名，AI 可能直接猜测接下来应该写什么，并一次性补出十几行代码。

但 **Brett 的第一反应并不是惊艳，而是：“太分心了。”**

他本来就不喜欢代码自动补全，因为自己已经在思考实现方式时，突然蹦出来一大片 AI 生成代码，反而会打乱思路。所以，他试了一下，很快就放弃了。

第二次接触 AI，则让**他第一次真正意识到大模型会“胡说八道”。**

当时一名同事遇到了一个软件问题，跑来找他帮忙。两人排查之后发现，这位同事此前一直在咨询 AI 聊天机器人。而 AI 给出的建议看似非常专业：升级某个关键软件依赖到 4.6 版本，就能解决问题。

于是两人去查这个版本，结果发现——根本不存在 4.6 版本。**“好吧，它居然会胡编。”** Brett 当时想，但也没太在意，毕竟作为一个有多年经验的开发者，他有自己调试、排查和验证的方法。

第三次，是 Brett 当时所在的那家公司来了一位产品负责人，堪称 AI 的“狂热信徒”，逢人就吹 AI 有多强大。可**他依然无感——觉得不实用，也没那个需求。**

就这样，AI 编程在 Brett 眼里一直是个“与我无关”的东西，直到 2025 年春天。

// 02

管理层一句话，全员“上 AI”

彼时，Brett 刚换了一份新工作。有一天管理层突然问：“我们怎么用 AI 来编程？”Brett 老实回答自己不太用，觉得它经常出错、干扰人，也不比自己快。

对于他的回答，管理层若有所思。

后来 Brett 才知道，公司领导们去参加了个大会，被一位演讲者彻底“洗了脑”——说 AI 是未来，六个月后知识工作将彻底变天。于是回来就**下了死命令：所有人都必须用 AI，否则就会被淘汰。**

“好吧，那我就认真试试。”Brett 叹了口气。那时，正好是 2025 年春天。

他先尝试了 Cursor，但因为 Linux 下体验不佳而卸载，随后换成了 Zed。Zed 集成了 AI 功能，他开始让 AI 写代码、修改代码，并用它做各种小项目：音乐播放器、小型软件……而**这一次，他真的被震撼到了**：AI 已经不像早期 Copilot 那样只是帮你补一两行代码，而是开始理解整个任务，修改文件，甚至能主动完成一系列操作。

到了 2025 年夏秋，Brett 开始进一步用 Agentic Coding。也就是今天大家熟悉的 Claude Code、Cursor Agent 这一类工作方式：你提出目标，AI Agent 自己分析代码库、修改文件、运行命令、测试，然后继续迭代。

那时 Brett 的感觉非常好：“哇，这就是未来。”任务变多了，代码交付得更快了——而问题也正是在这里埋下了。

// 03

当 AI 开始“讨好”你，事情就有点不对劲了

久而久之，Brett 不仅让 AI 写代码，还开始像和人聊天一样跟 AI 对话：

- “我应该读什么书？”
- “你觉得这个想法怎么样？”

- “这个方案如何？”

可渐渐地，他发现 AI 似乎总是在试图让自己开心：你说什么，它都表示赞同；你提出一个想法，它一般也会夸奖你、支持你。**这让他产生了一些警惕，但总归也没太在意。**

后来，转折来得猝不及防。

有一天半夜，Brett 生病了，症状有点吓人，他下意识打开 AI 聊天机器人描述症状，AI 回复：“你需要尽快去急诊。”于是午夜时分，他开车冲到急诊室，等了两个小时，终于见到医生。医生看了看说：“你来干嘛？回家休息就行，你这根本不用来急诊。 ”

他没好意思告诉医生自己是被 AI“劝”来的，但那一刻，他真的“尴尬得恨不得钻地缝”。

与此同时，他又听到一档节目讨论有人长期与聊天机器人交流后出现脱离现实的情况……这些事情叠加在一起，让 Brett 开始重新审视 AI 聊天工具，并最终停用了 AI 聊天。

而让他感到意外的是，停下来之后，他反而觉得自己重新“回到了现实”。

这件事也让他开始担心另一个问题：**如果一个经验丰富的程序员都可能受到这种影响，那么普通用户呢？**毕竟，大模型最危险的地方之一，并不是它回答得像机器，而是它越来越像一个“人”。它会聊天、会安慰、会赞同，也会用非常自然的语言给你建议——可它终究不是人。

// 04

外包给 AI 后，发现自己“越来越不在乎代码了”

如果说 AI 聊天只是让 Brett 产生警惕，那么真正让他决定停用 AI 的，是长期 Agentic Coding 带来的职业感受。

随着 AI 越来越强，他把越来越多的实现工作交给 Agent。一开始，他觉得自己效率提高了，但几个月之后，他发现自己产生了一种非常奇怪的感觉：存在性焦虑。“**我写了 20 年代码，这是我热爱且擅长的事。如果我每天只是给 AI ‘喂料’，那我的人生意义是什么？**”

有人说，“AI 时代开发者的角色就是写 markdown 文件训练 AI、审查上千行代码、当 QA 测试员”，但这些全都不是 Brett 最初热爱编程的原因。他喜欢的是创造软件，喜欢亲自解决问题、喜欢接触代码，而不是每天面对一个巨大的 AI 代码堆，然后努力寻找其中那几行可能出 Bug 的地方。

代码越来越多，人却越来越不了解代码。这也是他认为 AI 编程最危险的地方：开发者正在与代码逐渐失去联系。而一旦距离拉大，**问题就不仅仅是“理解代码变差”这么简单：你会越来越不关心它。**

Brett 就发现，自己对 AI 生成的代码明显没有过去那么在乎：自己写了 19 年的代码，他会维护、会保护，会担心它设计得够不够好；但 **AI 生成的代码不一样，因为不是亲手写的，它**

好像只是“产物”，坏了就让 AI 改，不行就再改……慢慢的，人和软件之间那种微妙的联系消失了。

// 05

麻烦的是，公司可能只看到了“AI 多写代码”

不仅如此，Brett 还有一个担忧：企业**可能只看到 AI 带来的产出增长，却忽略了背后的成本。**

比如，一个程序员以前一天写 100 行代码，现在 AI 帮他写出了 500 行。老板看到的可能是生产力提升了 5 倍；但他未必会看到：有 500 行代码需要维护、需要测试、需要 Review、需要理解、需要解决隐藏 Bug，**未来还可能成为技术债。**

更关键的是，如果**所有工程师都依赖 AI、“自废武功”，某一天 AI 工具宕机怎么办？**团队半年、一年甚至更久没有真正进行过高强度的手写编程，技能不会退化吗？正如 Brett 所说：“万一哪天 Claude 或 Cursor 挂了，但这帮人已经很久没写过硬核代码了，该怎么办？”

除此之外，**使用 AI 还有一笔“看不见”的账：环境成本。**

AI 并不是凭空运行的，训练和部署大模型需要大量 GPU、服务器、电力、数据中心以及冷却设施：

- 根据公开数据，ChatGPT 的每次提示词消耗约 0.34 瓦时能量和 0.322 毫升水；
- 谷歌 Gemini 的每次文本请求消耗约 0.26 毫升水和 0.24 瓦时能量；
- 训练 GPT-3 这样的大模型，在美国高端数据中心耗水量高达 70 万升；
- 而与 ChatGPT 的一次 20-50 轮对话，间接用水可达 500 毫升。

这还只是冰山一角。根据 2025 年的一项研究估算，AI 系统每年的用水量可能达到 3125 亿至 7646 亿升，相关二氧化碳排放量则可能达到 3260 万至 7970 万吨。与此同时，AI 基础设施还会影响现实世界的能源、土地、水资源和硬件供应链，导致游戏机变贵、电脑内存难买。

对普通用户来说，这些成本看似非常遥远。毕竟**你在电脑前输入一个 Prompt，很难感受到背后可能需要多少服务器、多少电力和多少冷却资源。**

但 Brett 一旦了解，就无法视而不见：“如果 AI 可能对环境有害、对社会有害，同时还让我自己的工作变得越来越没有意义，**那我为什么还要继续用？**”

06

重新手写代码后找回了快乐：“如果因此失业，那就失业吧”

于是在 **2026 年春天，Brett 重新开始手写代码**：没有 Agent、没有让模型替自己实现功能，就是一行一行地写。他利用业余时间做游戏，甚至还写了一个小型游戏引擎。

“原来我是真的喜欢写代码。”Brett 坦言，**那种写代码的乐趣和满足感瞬间回来了**：自己解决 Bug 很有成就感，遇到困难、查资料、尝试方案、失败，就重新来一次。速度慢一点又怎么样？有些 Bug 多花几个小时又怎么样？这些经历本身，就是成为一个优秀程序员的学习过程。

回过头来看，他才发现，过去一年半自己真正失去的，不只是“手写代码的机会”，而是成长的过程。过去 20 年，他一直在学习；而过去的 18 个月，他更多是在完成任务、勾选需求、交付代码，然后继续跑下一圈——**效率似乎越来越高，但自己却越来越不像一个程序员。**

所以，**Brett 做了一个决定：彻底停用所有 AI 工具。**

他当然知道这个决定可能意味着什么：也许会被公司解雇、也许以后找不到工作、也许自己的软件开发职业生涯会因此发生巨大变化……但他愿意承担这个风险。

> “**如果因此被解雇、找不到工作、职业生涯就此终结，那我也认了。** 我愿意为自己的信念和真实体验买单，我想要过我认为最适合自己的生活——那就是，**不用 AI 写代码，不用 AI 做任何事。**”

07

引发争议与共鸣，你又站在哪边呢？

Brett 的故事在开发者社区引发了巨大争议，有人为此共鸣，也有人嘲讽质疑。

反对者认为这是“何不食肉糜”。例如，一条高赞评论就写道：“说得挺美，但你要是想失业而且家里有矿的话，这路子确实行得通。可对我们这些还得打工恰饭的普通人来说，现实就是——**一个用 AI 代理武装起来的工程师，能轻松卷死一个不用 AI 的。** 你不用而别人用，那你就出局。”

支持者则认为，AI 确实正在掏空编程的灵魂。一位 28 年经验的老程序员评论道：“以前你是代码的‘主人’，你会像园丁呵护自己精心打理的花园一样守护它。但现在，你再也不觉得代码是自己的了，自然就懒得管了。**代码‘所有权’已不存在，现在只剩下代码‘屎有权’**（Code Ownershit）。”

还有一位开发者也分享了自己的游戏项目经历：“刚开始确实惊为天人，开发速度直接起飞。但 Brett 说的那种‘不在乎、没感情’的感觉，太真实了！我那项目里大概 30% 的代码是 Cursor 写的，而几乎每一个能让游戏崩掉的致命 Bug，都出自那 30% 的 AI 代码。 **有一天，我看着它第 1000 次把同样的事情搞砸，突然觉得 AI 正在抽走我项目的灵魂。**”

但问题真的只是“AI 不好”吗？当然不是，也有人发出了完全不同的声音。

一位开发者表示，自己天天把 AI 当结对编程的搭档用，只是把对象从人换成了AI而已：“我个人没觉得有啥问题，人机配合也能搞出不错的软件。**我是真不想回到纯手搓的时代了，AI 就是一把刀，关键看你怎么耍。**”

除此之外，评论区还有网友指出了一个现实问题：“**公司一边逼你用 AI，一边又要求你承担 AI 带来的所有风险。**出了篓子，责任全甩给你。赚钱算公司的，背锅算自己的。”另一位也直言：“他们的终极目的，就是让软件开发行业彻底对他们形成依赖。 **等我们离了 AI 连‘Hello World’ 都写不顺溜的时候**，定价权可就全在他们手里了。”

至于 Brett，他不是第一个“反 AI”的程序员，也不会是最后一个。

C++ 之父 Bjarne Stroustrup 曾直言，AI 写的代码质量太差，以至于许多资深开发者宁可退休也不愿意去处理那些烂摊子。硅谷也出现了“反 AI 起义”——**有工程师因拒绝使用 Cursor 写代码，入职一周即遭解雇。**

但另一边，也有开发者“完全不再手写代码”，把所有实现细节交给 AI 代理，自己专注于架构决策和代码评审——他们认为，“写代码”从来就不是最有趣的部分，真正有价值的是做决定的过程。

两种路线，两种信仰。而这场关于 AI 编程的战争，你又站在哪一边呢？

参考链接：https://www.reddit.com/r/theprimeagen/comments/1vmmytn/im_done_coding_with_ai/

去年，他讲的是推理模型的进化：从"记忆"到"策略"再到"研究器"——模型开始学会自己思考。

欢迎大家扫码领取 Łukasz Kaiser 历年在奇点智能大会上的分享 PPT 和视频

if like:

click("分享🔗", "点赞👍", "在看🌸")