MICHAEL TRUELL

LENNY'S PODCAST

BILINGUAL TRANSCRIPT

ORIGINAL BY

Lenny Rachitsky

@lennysan • x.com/lennysan

ANALYSIS BY

@Penny777 • x.com/penny777

Generated: 2026-01-16

Michael Truell - 双语对照

This is the complete bilingual transcript for Lenny's Podcast featuring Michael Truell, the co-founder and CEO of Cursor (Anysphere).

[00:00:00] Michael Truell

English:

... our goal with Cursor is to invent a new type of programming, a very different way to build software. So a world kind of after code, I think that more and more being an engineer will start to feel like being a logic designer, and really, it will be about specifying your intent for how exactly you want everything to work.

中文翻译:

……我们在 Cursor 的目标是发明一种新型的编程方式，一种截然不同的软件构建方式。也就是一个“后代码”时代的世界。我认为，工程师的角色将越来越像是一个“逻辑设计师”，其核心将在于明确你的意图，即你到底希望一切如何运作。

[00:00:16] Lenny Rachitsky

English:

What is the most counter-intuitive thing you've learned so far about building Cursor?

中文翻译:

到目前为止，在构建 Cursor 的过程中，你学到的最反直觉的事情是什么？

[00:00:20] Michael Truell

English:

We definitely didn't expect to be doing any of our own model development. And at this point, every magic moment in Cursor involves a custom model in some way.

中文翻译:

我们最初完全没预料到会进行任何自主模型的开发。而现在，Cursor 中每一个“奇迹时刻”在某种程度上都涉及到了自定义模型。

[00:00:26] Lenny Rachitsky

English:

What's something that you wish you knew before you got into this role?

中文翻译:

在担任这个角色之前，有什么是你希望自己早点知道的?

[00:00:29] Michael Truell

English:

Many people you hear hire too fast, I think we actually hired too slow to begin with.

中文翻译:

你经常听到很多人说招人太快了，但我认为我们起步阶段其实招得太慢了。

[00:00:35] Lenny Rachitsky

English:

You guys went from $0 to 100 million ARR in a year and a half, which is historic. Was there an inflection point where things just started to really take off?

中文翻译:

你们在一年半的时间里将 ARR（年度经常性收入）从 0 做到了 1 亿美元，这是史无前例的。这中间是否存在一个转折点，让业务真正开始腾飞?

[00:00:43] Michael Truell

English:

The growth has been fairly just consistent on an exponential. And exponential to begin with feels fairly slow when the numbers are really low, and it didn't really show off to the races to begin with.

中文翻译:

增长基本上一直保持着稳定的指数级态势。在基数非常低的时候，指数级增长起初感觉相当缓慢，所以刚开始并没有那种“一骑绝尘”的感觉。

[00:00:51] Lenny Rachitsky

English:

What do you think is the secret to your success?

中文翻译:

你认为你们成功的秘诀是什么?

[00:00:53] Michael Truell

English:

I think it's been...

中文翻译:

我认为是……

[00:00:55] Lenny Rachitsky

English:

Today, my guest is Michael Truell. Michael is co-founder and CEO of Anysphere, the company behind Cursor. If you've been living under a rock and haven't heard of Cursor, it is the leading AI code editor, and is at the very forefront of changing how engineers and product teams build software. It's also one of the fastest growing products of all time, hitting 100 million ARR just 20 months after launching, and then 300 million ARR just two years since launch.

Michael's been working on AI for 10 years. He studied computer science and math at MIT, did AI research at MIT and Google, and is a student of tech and business history. As you'll soon see, Michael thinks deeply about where things are heading, and what the future of building software looks like. We chat about the origin story of Cursor, his prediction of what happens after code, his biggest counter-intuitive lessons from building Cursor, where he sees things going for software engineers, and so much more.

Michael does not do many podcasts. The only other podcast he's ever done is Lex Fridman, so it was a true honor to have Michael on. If you enjoy this podcast, don't forget to subscribe and follow it in your favorite podcasting app or YouTube. Also, if you become an annual subscriber of my newsletter, you get a year free of Perplexity, Linear, Superhuman, Notion, and Granola. Check it out at lennysnewsletter.com, and click bundle. With that, I bring you Michael Truell.

中文翻译:

今天的嘉宾是 Michael Truell。Michael 是 Anysphere 的联合创始人兼 CEO，这家公司开发了 Cursor。如果你一直消息闭塞（living under a rock）还没听说过 Cursor，它是目前领先的 AI 代码编辑器，处于改变工程师和产品团队构建软件方式的最前沿。它也是有史以来增长最快的产品之一，上线仅 20 个月 ARR 就达到 1 亿美元，上线两年即达到 3 亿美元 ARR。

Michael 研究 AI 已经 10 年了。他在麻省理工学院（MIT）学习计算机科学和数学，曾在 MIT 和 Google 从事 AI 研究，并且是技术和商业史的研究者。正如你即将看到的，Michael 对事物的发展方向以及软件构建的未来有着深刻的思考。我们聊到了 Cursor 的起源故事、他对“后代码时代”的预测、构建 Cursor 过程中最反直觉的教训、他如何看待软件工程师的未来，以及更多内容。

Michael 很少参加播客。他唯一参加过的另一个播客是 Lex Fridman，所以能请到 Michael 真是莫大的荣幸。如果你喜欢这个播客，别忘了在常用的播客应用或 YouTube 上订阅和关注。此外，如果你成为我时事通讯的年度订阅者，你将免费获得一年的 Perplexity、Linear、Superhuman、Notion 和 Granola。请访问 lennysnewsletter.com 并点击 bundle 查看。下面，让我们欢迎 Michael Truell。

[00:02:14] Lenny Rachitsky (Sponsor Break - Eppo)

English:

This episode is brought to you by Eppo. Eppo is a next-generation A/B testing and feature management platform built by alums of Airbnb and Snowflake for modern growth teams... [Full sponsor text omitted for brevity in translation, focusing on the core content]

中文翻译:

本集节目由 Eppo 赞助。Eppo 是由 Airbnb 和 Snowflake 的校友为现代增长团队打造的下一代 A/B 测试和功能管理平台……（赞助商内容略）

[00:03:31] Lenny Rachitsky (Sponsor Break - Vanta)

English:

This episode is brought to you by Vanta. When it comes to ensuring your company has top-notch security practices, things get complicated fast... [Full sponsor text omitted for brevity]

中文翻译:

本集节目由 Vanta 赞助。当涉及到确保公司拥有顶级的安全实践时，事情会迅速变得复杂……（赞助商内容略）

[00:04:26] Lenny Rachitsky

English:

Michael, thank you so much for being here. Welcome to the podcast.

中文翻译:

Michael，非常感谢你能来。欢迎来到本播客。

[00:04:30] Michael Truell

English:

Thank you. It's great to be here. Thank you for having me.

中文翻译:

谢谢。很高兴来到这里。感谢你的邀请。

[00:04:33] Lenny Rachitsky

English:

When we were chatting earlier, you had this really interesting phrase, this idea of what comes after code. Talk about that, just the vision you have of where you think things are going in terms of moving from code to maybe something else.

中文翻译:

我们之前聊天时，你提到了一个非常有趣的短语，即“后代码时代”（what comes after code）的想法。谈谈这个吧，关于你认为事物将如何从代码转向其他形式的愿景。

[00:04:45] Michael Truell

English:

Our goal with Cursor is to invent sort of a new type of programming, a very different way to build software, that's kind of just distilled down into you describing the intent to the computer for what you want in the most concise way possible, and really distilled down to just defining how you think the software should work, and how you think it should look. With the technology that we have today, and as it matures, we think you can get to a place where you can invent a new method of building software that's [inaudible 00:05:16] higher level, and more productive, in some cases, more accessible too.

And that process will be a gradual moving away from what building software looks like today. I want to contrast it with maybe the vision of what software looks like in the future that I think... A couple of visions

that are in a popular consciousness that we at least have some disagreement with. One is, there's a group of people who think that software building in the future is going to look very much like it does today, which mostly means text editing, formal programming languages, like TypeScript, and Go, and C, and Rust. And then there's another group that kind of thinks you're just going to type into a bot, and you're going to ask it to build you something, and then you're going to ask it to change something about what you're building, and it's kind of like this chatbot, Slackbot style where you're talking to your engineering department.

And we think that there are problems with both of those visions. I think that on the chatbot style end of things... And we think it's going to look weirder than both. The problem with the chatbot style end of things is that lacks a lot of precision. If you want humans to have complete control over what the software looks like, and how it works, you need to let them gesture at what they want to be changed in a form factor that's more precise than just, "Change this about my app." In a text box, removed from the whole thing. And then the version of the world where nothing changes we think is wrong, because we think the technology is going to get much, much, much better.

And so a world after code, I think that it looks like a world where you have a representation of the logic of your software that does look more like English, you have written down... You can imagine in [inaudible 00:07:08] form, you can imagine in kind of an evolution of programming language towards pseudocode. You have written down the logic of the software, and you can edit that at a high level, and you can point at that. And it won't be the impenetrable millions of lines of code, it'll instead be something that's much Terser, and easier to understand, easier to navigate. But that world where the kind of crazy, hard to understand symbols start to evolve towards something that's a little bit more human-readable, and human-editable, is one that we're working towards.

中文翻译:

我们在 Cursor 的目标是发明一种新型的编程方式，一种截然不同的软件构建方式。它被简化为你以最简洁的方式向计算机描述你的意图，即你想要什么，并真正浓缩为定义你认为软件应该如何运作以及应该长什么样。利用我们现有的技术，随着它的成熟，我们认为可以创造出一种更高层次、更高效、在某些情况下也更易上手的软件构建新方法。

这个过程将是逐渐脱离今天的软件构建模式。我想把它与目前大众认知中的几种未来软件愿景做个对比，我们对这些愿景持不同意见。一种观点认为，未来的软件构建看起来会和今天非常相似，主要还是文本编辑、正式的编程语言（如 TypeScript, Go, C, Rust）。另一种观点认为，你只需要对着机器人打字，让它帮你造个东西，或者改个东西，就像聊天机器人或 Slack 机器人一样，仿佛你在和你的工程部门对话。

我们认为这两个愿景都有问题。而且我们认为未来的样子会比这两者都更“怪异”。聊天机器人模式的问题在于缺乏精确度。如果你希望人类对软件的外观和运作方式拥有完全的控制权，你需要让他们能够以比在文本框里说一句“改一下我的 App”更精确的方式来指明想要修改的地方。而那种认为一切都不会改变的观点也是错误的，因为我们相信技术会变得好得多。

所以，“后代码”的世界，我认为它看起来像是一个你拥有软件逻辑表达的世界，而这种表达看起来更像英语。你可以想象它是一种向“伪代码”（pseudocode）演进的编程语言。你写下软件的逻辑，可以在高层次上进行编辑和指点。它不再是那几百万行难以理解的代码，而是一些更简洁、更易理解、更易导航的东西。那个让疯狂、难懂的符号演变成更具可读性、可编辑性的世界，正是我们努力的方向。

[00:07:36] Lenny Rachitsky

English:

This is a profound point. I want to make sure people don't miss what you're saying here, which is that what you're envisioning in the next year essentially is kind of when things start to shift, is, people move

away from even seeing code, having to think in code in JavaScript and Python, and there's this abstraction that will appear, essentially pseudocode, describing what the code should be doing more in English sentences.

中文翻译:

这是一个深刻的观点。我想确保大家没有错过你在这里说的话，也就是说，你预见在未来一年左右，事情会开始发生转变：人们不再需要直接看代码，不再需要用 JavaScript 或 Python 的逻辑去思考，而是会出现一种抽象层，本质上是伪代码，用英语句子来描述代码应该做什么。

[00:07:59] Michael Truell

English:

Yep. We think it ends up looking like that, and we're very opinionated that that path goes through existing professional engineers, and it looks like this evolution away from code. And it definitely looks like the human still being in the driver's seat, and the human having both a ton of control over all aspects of the software and not giving that up. And then also the human having the ability to make changes very quickly, having a fast duration loop and not just having something in the background that's super slow and takes weeks, go do all your work for you.

中文翻译:

是的。我们认为最终会变成那样，而且我们坚信这条路径会通过现有的专业工程师来实现，它看起来像是从代码中进化出来的。它绝对意味着人类仍然掌握主动权（in the driver's seat），人类对软件的所有方面都拥有巨大的控制权且不会放弃。同时，人类能够非常快速地做出更改，拥有快速的迭代循环，而不是让后台某个超级慢的东西花上几周时间替你完成所有工作。

[00:08:33] Lenny Rachitsky

English:

This begs the question for people that are currently engineers, or thinking about becoming engineers, or designers, or product manager, what skills do you think will be more and more valuable in this world of what comes after code?

中文翻译:

这就引出了一个问题：对于现在的工程师，或者想成为工程师、设计师、产品经理的人来说，在这个“后代码”世界里，哪些技能会变得越来越有价值？

[00:08:50] Michael Truell

English:

I think taste will be increasingly more valuable. And I think often when people think about tastes in the realm of software, they think about visuals, or taste over smooth animations, and coloring things, UI, UX, et cetera on the visual design of things. And the visual side of things is an important part of defining a piece of software, but then, as mentioned before, I think that the other half of defining a piece of software is the logic of it, and how the thing works.

And we have amazing tools for specing out the visuals of things, and then when you get into the logic of how a piece of software works, really, the best representation we have of that is code right now. You can kind of gesture at it with Figma, and you can gesture at it with writing down notes, but it's when you have

an actual working prototype. And so I think that more and more, being an engineer will start to feel like being a logic designer, and really, it will be about specifying your intent for how exactly you want everything to work. It'd be more about the whats, and a little bit less about how exactly you're going to do things under the hood.

I think taste will be increasingly important. I think one aspect of software engineering, and we're very far from this right now, and there are lots of funny memes going around the internet about some of the trials and tribulations people can run into if they trust AI for too many things that comes to engineering, around building apps that have glaring deficiencies, and problems, and functionality issues. But I think we will get to a place where you'll be able to be less careful as a software engineer, which, right now, is an incredibly, incredibly important skill. We'll move a little bit from carefulness, and a little bit more towards taste.

中文翻译:

我认为“品味”（taste）将变得越来越有价值。通常当人们想到软件领域的品味时，会想到视觉效果、流畅的动画、配色、UI/UX 等视觉设计方面。视觉确实是定义软件的重要部分，但正如之前提到的，定义软件的另一半是它的逻辑，即它是如何运作的。

我们已经有了很棒的工具来规划视觉效果，但当你深入到软件运作的逻辑时，目前我们拥有的最佳表达方式就是代码。你可以用 Figma 勾勒它，也可以写笔记描述它，但只有当你拥有一个实际可运行的原型时才算数。所以我认为，工程师会越来越像“逻辑设计师”，核心在于明确你的意图，即你到底希望一切如何运作。它将更多地关乎“做什么”（what），而较少关乎底层具体“怎么做”（how）。

我认为品味会日益重要。目前软件工程的一个方面——虽然我们现在离那还很远，网上也有很多搞笑的梗，说如果人们在工程上过度信任 AI 会遇到各种磨难，比如造出的 App 有明显的缺陷和功能问题。但我认为我们最终会达到一个境界，作为软件工程师，你不需要像现在这样“谨小慎微”，而“严谨”在目前是一项极其重要的技能。我们将从“严谨”稍微转向“品味”。

[00:10:40] Lenny Rachitsky

English:

This makes me think of vibe coding, is that kind of what you're describing when you talk about not having to think about the details as much, and just kind of going with the flow?

中文翻译:

这让我想到了“氛围编程”（vibe coding），当你谈到不需要考虑那么多细节、顺其自然时，是指这种感觉吗？

[00:10:49] Michael Truell

English:

I think it's related. I think that vibe coding right now describes exactly this state of creation that is pretty controversial, where you're generating a lot of coding, you aren't really understanding the details. That is a state of creation that then has lots of problems, you don't really... By not understanding the details under the hood right now, you then very quickly get to a place where you're kind of limited at a certain point, where you create something that's big enough that you can't change. And so I think some of the ideas that we're interested around, how do you give people continued control over all the details when they don't really understand the code? I think that solutions there are very relevant to the people who are vibe coding right now. I think that right now, we lack the ability to let the tastemakers actually have complete control over the software. One of the issues also with vibe coding, and letting taste really shine

through from people is, you can create stuff, but a lot of it the AI making decisions that are unwieldy and you don't have to control over.

中文翻译:

我觉得有关联。目前的“氛围编程”描述的正是一种颇具争议的创作状态：你生成了大量代码，但并不真正理解细节。这种创作状态会带来很多问题。因为不理解底层细节，你很快就会遇到瓶颈，比如当你创造的东西大到一定程度时，你就没法修改它了。所以我们感兴趣的一些想法是：当人们不真正理解代码时，如何让他们持续控制所有细节？我认为这些解决方案对于现在进行“氛围编程”的人来说非常重要。目前，我们还缺乏让“品味决策者”真正完全控制软件的能力。氛围编程的一个问题是，虽然你能创造出东西，但其中很多决策是 AI 做的，这些决策可能很笨重，而且你无法控制。

[00:11:56] Lenny Rachitsky

English:

One more question along these lines. You threw out this word taste. When you say taste, what are you thinking?

中文翻译:

关于这一点再问一个问题。你提到了“品味”这个词。当你提到品味时，你具体指什么？

[00:12:01] Michael Truell

English:

I'm thinking having the right idea for what should be built. It will become more and more about effortless translation of, here's exactly what you want built, here's how you want everything to work, here's how you want it to look. And then you'll be able to make that on a computer, and it will less be about this kind of translation layer of, you and your team have a picture of what you'd want to build, and then you have to really painstakingly, labor-intensive, lay out that into a format that a computer can then execute and interpret. I think less than the UI side of things, maybe taste is a little bit of a misnomer, but just about having the right idea for what should be built.

中文翻译:

我指的是对于“应该建造什么”有正确的想法。它将越来越关乎于如何毫不费力地转化：这就是你想要建造的东西，这就是你希望它运作的方式，这就是你希望它看起来的样子。然后你就能在计算机上实现它。它将不再那么依赖于那种繁琐的转化层——即你和团队脑子里有个蓝图，然后必须费尽心力、耗费大量劳动将其转化为计算机可以执行和解释的格式。除了 UI 方面，也许“品味”这个词不完全准确，但它核心就是关于对“该造什么”有正确的判断。

[00:12:39] Lenny Rachitsky

English:

Awesome. Okay. I want to come back to these topics, but I want to actually zoom us back out to the beginnings of Cursor. I have never heard the origin story, I don't think many people know how this whole thing started. Basically you guys are building one of the fastest growing products in the history of the world, it's changing the way people build products, it's changing careers, professions, it's changing so much. How did it all begin? Any memorable moments along the journey of the early days?

中文翻译:

太棒了。好的，我稍后会回到这些话题，但现在我想把镜头拉回到 Cursor 的起点。我从未听过它的起源故事，我想很多人也不知道这一切是怎么开始的。基本上，你们正在构建世界上增长最快的产品之一，它正在改变人们构建产品的方式，改变职业和专业，改变了太多东西。这一切是怎么开始的？早期阶段有什么难忘的时刻吗？

[00:13:05] Michael Truell

English:

Cursor kind of started as a solution search of a problem, and a little bit where it very much came from reflecting on how AI was going to get better over the course of the next 10 years. There were kind of two defining moments, one was being really excited by using the first beta version of Code Pilot, actually. This was the first time we had used an AI product that was really, really, really useful, and was actually just useful at all, and wasn't just a vaporware kind of demo thing.

And in addition to being the first AI product that we'd use that was useful, Code Pilot was also one of the most useful, if not the most useful dev tool we'd ever adopted, and that got us really excited. Another moment that got us really excited was the series of scaling on papers coming out of OpenAI and other places that showed that even if we had no new ideas, AI was going to get better and better just by pulling on simple levers, like scaling up the models, and also scaling up the data that was going into the models.

And so at the end of 2021, beginning of 2022, this got us excited about how AI products were now possible, this technology was going to mature into the future. And it felt like when we looked around, there were lots of people talking about making models, and it felt like people weren't really picking an area of knowledge work and thinking about what it was going to look like as AI got better and better. And that set us on the path to an idea generation exercise, it was like, "How are these areas of knowledge work going to change in the future as this tech gets more mature? What is the end state of the work going to look like? How are the tools that we use to do that work going to change? How are the models going to need to get better to support changes in the work? And once scaling and pre-training ran out, how are you going to keep pushing for technological capabilities?"

And the misstep at the beginning of Cursor is we actually worked on... We sort of did this whole grand exercise, and we decided to work on an area of knowledge work that we thought would be relatively uncompetitive, and sleepy, and boring, and no one would be looking at it, because we thought, "Oh, coding's great, coding's totally going to change with this AI, but people are already doing that." So there was a period of four months to begin with, where we were actually working on a very different idea, which was helping to automate and augment mechanical engineering, and building tools for mechanical engineers.

There were problems from the get-go in that. Me and my co-founders, we weren't mechanical engineers. We had friends who were mechanical engineers, but we were very much unfamiliar with the field. So there was a little bit of a blind man and the elephant problem from the get-go. There were problems around, how would you actually take the models that exist to today and make them useful for mechanical engineering? The way we netted out is, you need to actually develop your own models from the get-go. And the way we did that was tricky, and there's not a lot of data on the internet of 3D models of different tools and parts, and the steps that I expect to build up to those 3D models, and then getting them from the sources that have them is also a tricky process too.

But eventually what happened was, we came to our senses, we realized we're not super excited about mechanical engineering, it's not the thing we want to dedicate our lives to. And we looked around, and in the area of programming, it felt like despite a decent amount of time ensuing, not much has changed, and it felt like the people that were working on the space maybe had a disconnect with us, and it felt like

they weren't being sufficiently ambitious about where everything was going to go in the future, and how all of software creation was going to blow through these models. And that's what set us off on the path to building Cursor.

中文翻译:

Cursor 的诞生有点像是“为问题寻找解决方案”，很大程度上源于对未来 10 年 AI 将如何发展的反思。有两个关键时刻：一是我们第一次使用 GitHub Copilot 的测试版时感到非常兴奋。那是我们第一次使用到真正、真正非常有用的 AI 产品，而不仅仅是一个虚假宣传（vaporware）的演示。

除了它是第一个好用的 AI 产品外，Copilot 也是我们采用过的最有用（如果不是最有用的话）的开发工具之一，这让我们非常兴奋。另一个让我们兴奋的时刻是 OpenAI 等机构发布的一系列关于“规模法则”（scaling laws）的论文，这些论文表明，即使我们没有新想法，只要通过简单的杠杆——比如扩大模型规模和增加数据量，AI 就会变得越来越好。

所以在 2021 年底、2022 年初，这让我们意识到 AI 产品现在是可能的，这项技术未来会走向成熟。当时我们环顾四周，发现很多人在谈论做模型，但似乎没有人真正挑选一个知识工作领域，去思考随着 AI 变强，那个领域会变成什么样。这开启了我们的头脑风暴：随着技术成熟，这些知识工作领域未来会如何变化？工作的终极状态是什么？我们使用的工具会如何改变？模型需要如何改进来支持这些变化？当规模化和预训练达到极限后，如何继续推动技术能力？

Cursor 最初的一个失误是，我们做完这一系列宏大的思考后，决定去攻克一个我们认为相对没有竞争、沉闷且无聊的知识工作领域，因为我们觉得：“编程很棒，AI 肯定会改变编程，但已经有人在做了。”所以最初有四个月的时间，我们其实在做一个完全不同的想法：帮助机械工程实现自动化和增强，为机械工程师构建工具。

从一开始就有很多问题。我和我的联合创始人都不是机械工程师。虽然我们有机械工程师朋友，但我们对这个领域非常陌生。所以一开始就有种“盲人摸象”的感觉。还有一个问题是，你如何让现有的模型对机械工程有用？我们的结论是，你必须从头开发自己的模型。但这很棘手，互联网上没有多少关于不同工具和零件的 3D 模型数据，也没有构建这些模型所需的步骤数据，从拥有这些数据的来源获取它们也非常困难。

但最终，我们清醒了过来，意识到我们对机械工程并没有那么大的热情，这不是我们想奉献一生的事业。我们环顾四周，发现在编程领域，尽管时间流逝，但并没有发生太大变化。感觉在这个领域工作的人可能与我们脱节了，他们对未来的发展方向不够雄心勃勃，没意识到所有的软件创作都将通过这些模型实现。这就是我们走上构建 Cursor 之路的契机。

[00:17:04] Lenny Rachitsky

English:

Okay. So interesting. Okay, so first of all, I love that... This is advice that you often hear of go after a boring industry because no one's going to be there, and there's opportunity. And sometimes it works, but I love that in this journey, it's like, "No, actually, go after the hottest, most popular space, AI coding, app building." And it worked out. And the way you phrased it just now is, you didn't see enough ambition potentially, that you thought there was more to be done. So it feels like that's an interesting lesson. Even if something looks like, "Okay, it's too late, there's GitHub, Code Pilot's out there." Some other products. If you notice that they're just not as ambitious as they could be, or as you are, or you see almost a flaw in their approach, that there's still a big opportunity. Does that resonate?

中文翻译:

太有意思了。首先，我很喜欢这一点……人们常听到的建议是去寻找一个无聊的行业，因为那里没人，有机会。有时这确实奏效，但我喜欢你们这段旅程中的反转：“不，实际上，去追求最热门、最受欢迎的领域——AI 编程和应用构建。”而且成功了。你刚才的说法是，你觉得现有的参与者可能不够雄心勃勃，你认为还有更多

事情可以做。这似乎是一个有趣的教训：即使某个领域看起来“已经太晚了，已经有 GitHub Copilot 了”，如果你发现他们不够有野心，或者你看到了他们方法的缺陷，那里仍然有巨大的机会。你认同吗？

[00:17:46] Michael Truell

English:

That totally resonates. A part of it is, you need there to be leapfrogs that can happen, you need there to be things that you can do. And I think the exciting thing about AI is, in a bunch of places, and I think this is very much still true of our space, and can talk about how we think about that and how we deal with that, but I think that just the ceiling is really high. And yes, if you look around, probably even if you take the best tool, any of these fields, there should be a lot more that needs to be done over the next few years. Having that space, having that high ceiling, I think is unique amongst areas of software, at least the degree to which it is high with AI.

中文翻译:

完全认同。部分原因在于，你需要存在实现“跨越式发展”的可能性，需要有你能做的事情。我认为 AI 令人兴奋的地方在于，在很多领域（我认为我们的领域依然如此），天花板真的非常高。是的，如果你环顾四周，即使是最好的工具，在未来几年也还有大量工作要做。拥有这种空间和极高的天花板，我认为在软件领域是独一无二的，至少 AI 带来的高度是前所未有的。

[00:18:30] Lenny Rachitsky

English:

Let's come back to the IDE questions. So there's a few routes you could have taken, and other companies are doing different routes. So there's building an IDE for engineers to work within and adding AI magic to it, there's another route of just a full AI agentic dev product, and then there's just a model that is very good at coding, and focusing on building the best possible coding model. What made you decide and see that the IDE path was the best route?

中文翻译:

让我们回到 IDE（集成开发环境）的问题。你们本可以走几条不同的路线，其他公司也在尝试不同的路径。比如为工程师构建一个 IDE 并加入 AI 魔法；或者走全 AI 智能体（agentic）开发产品的路线；再或者只做一个非常擅长编程的模型，专注于构建最好的编程模型。是什么让你决定并认定 IDE 路径是最佳选择？

[00:18:54] Michael Truell

English:

The folks who were from the get-go working on just a model were working on end-to-end automation programming. I think they were trying to build something very different from us, which is, we care about giving humans control over all of the decisions in the end tool that they're building. And I think those folks were very much thinking of a future where end-to-end, the whole thing is done by AI, and maybe the AI is making all the decisions too. And so, one, there was a personal interest component. Two, I think that always, we've tried to be intense realists about where the technology is today, very, very, very excited about how AI is going to mature over the course of many decades. But I think that sometimes people... There's an instinct to see AI do magical things in one area, and then kind of anthropomorphize these models, and think it's better than a smart person here, and so it must be better than a smart person there.

But these things have massive issues, and we... From the very start, our product development process was really about dogfooding, and using the tool intensely every day. And we never wanted to ship anything that wasn't useful to us, and we had the benefit of doing that because we were the end users part of our product. And I think that that instills a realism in you around where the tech is right now, and so that definitely made us think that we need the humans to be in the driver's seat, the AI cannot do everything. We're also interested in giving humans that control too for personal reasons, and so that gets you away from just your model company that also gets you away from just this end-end stuff without the human having control.

And then the way you get to an IDE versus maybe a plug-in to an existing coding environment is the belief that programming is going to flow through these models, and the active programming is going to change a lot over the course of the next few years. And that the extensibility that existing coding environments have is so, so, so limited, so if you think that the UIs may change a lot, if you think that the form factor programming is going to change a lot, necessarily need to have control over the entire application.

中文翻译:

那些从一开始就只做模型的人，是在做端到端的自动化编程。我认为他们想做的东西和我们非常不同。我们关心的是让渡给人类对最终工具中所有决策的控制权。而那些人可能在构想一个未来，即端到端的一切都由 AI 完成，甚至决策也由 AI 制定。

所以，第一，有个人兴趣的成分。第二，我认为我们一直试图对当前的技术水平保持极度的现实主义，虽然我们对 AI 在未来几十年的成熟感到非常非常兴奋。但我觉得有时人们会有一种本能，看到 AI 在某个领域做了神奇的事，就开始把模型人格化，觉得它在这里比聪明人强，那么在别处也一定比聪明人强。

但这些模型存在巨大的问题。从一开始，我们的产品开发过程就是关于“吃自己的狗粮”（dogfooding），每天高强度使用这个工具。我们从不想发布任何对自己没用的东西，我们有这个优势，因为我们本身就是产品的终端用户。这让你对技术的现状有一种现实感，因此这绝对让我们认为需要人类掌握主动权，AI 不能包办一切。出于个人原因，我们也希望赋予人类这种控制权，这让你远离了单纯的模型公司，也远离了那种没有人类控制的端到端自动化。

至于为什么选择 IDE 而不是现有环境的插件，是因为我们相信编程将通过这些模型流动，编程行为在未来几年会发生巨大变化。现有编码环境的可扩展性非常有限，如果你认为 UI 会发生巨变，如果你认为编程的形式会发生巨变，你就必须掌控整个应用程序。

[00:21:04] Lenny Rachitsky

English:

I know that you guys today have an IDE, and that's probably the bias you have of this is maybe where the future is heading, but I'm just curious, do you think a big part of the future is also going to be AI engineers that are just sitting in Slack and just doing things for you? Is that something that fits into Cursor one day?

中文翻译:

我知道你们现在做的是 IDE，这可能是你们对未来走向的偏好，但我很好奇，你是否认为未来的很大一部分也会是那些“坐在 Slack 里”为你干活的 AI 工程师？这在未来的某一天会融入 Cursor 吗？

[00:21:20] Michael Truell

English:

I think you'll want the ability to move between all of these things fairly effortlessly, and sometimes I think you will want to have the thing kind of go spin off on its own for a while, and then I think you'll want the

ability to pull in the AI's work, and then work with it very, very, very quickly, and then maybe have it go spin off again. And so these kind of background versus foreground form factors, I think you want that all to work well in one place. And I think the background stuff, there's a segment of programming that it's especially useful for, which is type of programming tasks where it's very easy to specify exactly what you want without much description, and exactly what correctness looks like without much description.

Bug fixes are a great example of that, but it's definitely not all of programming. So I think that what the IDE is will totally change over time, and our approach to having our own editor was premised on, it's going to have to evolve over time. And I think that that will both include, you can spin off things from different surface areas like Slack, or your issue tracker, or whatever it is, and I think that will also include the pane of glass that you're staring at is going to change a lot. We just mostly think of an IDE as the place where you are building software.

中文翻译:

我认为你会希望能够毫不费力地在这些形式之间切换。有时你希望让 AI 自己去跑一会儿，然后你再把 AI 的成果拿回来，非常快速地处理它，然后再让它继续跑。所以这种“后台”与“前台”的形式，我认为你希望它们在一个地方协同工作。后台模式对某些编程环节特别有用，比如那些很容易描述需求、也很容易定义正确性的任务。

修复 Bug 就是一个很好的例子，但这绝对不是编程的全部。所以我认为 IDE 的定义会随时间彻底改变，我们坚持做自己的编辑器就是基于“它必须不断进化”的前提。这既包括你可以从 Slack 或任务跟踪器等不同界面启动任务，也包括你盯着的那块“玻璃面板”（编辑器界面）会发生很大变化。我们只是把 IDE 看作你构建软件的地方。

[00:22:38] Lenny Rachitsky

English:

I think something people don't talk enough about with talking about agents and all these AI engineers that are going to be doing all this stuff for you, is basically we're all becoming engineering managers, with a lot of reports that are just not that smart, and you have to do a lot of reviewing, and approving, and specifying. I guess thoughts on that, and is there anything you could do to make that easier? Because that sounds really hard. Anyone that has had a large team, being like, "Oh my god, all these junior people just checking in with me doing not high quality work over and over." It's just like, "What a life. It's going to suck."

中文翻译:

我觉得人们在谈论智能体和 AI 工程师时，有一点谈得不够：本质上我们都变成了工程经理，带着一群不太聪明的下属，你必须做大量的审查、批准和明确指令的工作。你对此怎么看？有什么办法能让这变得更容易吗？因为这听起来很辛苦。任何带过大团队的人都会觉得：“天哪，所有这些初级人员不断向我汇报，产出的工作质量还不高。”这种生活简直太糟糕了。

[00:23:11] Michael Truell

English:

Yeah. Maybe you [inaudible 00:23:12] one-on-ones with [inaudible 00:23:15].

中文翻译:

是啊。也许你还得跟它们开一对一会议（1-on-1s）。

[00:23:15] Lenny Rachitsky

English:

So many one-on-ones.

中文翻译:

开不完的一对一会议。

[00:23:17] Michael Truell

English:

Yeah. So the customers we've seen have most success with AI I think are still fairly conservative about some of the ways in which they use this stuff. And so I do think today, the most successful customers really lean on things like our next edit prediction, where your coding is normal, and making the next into actions you're going to do. And then they also really lean on scoping down the stuff that you're going to hand off to the bot, and for a fixed percent of your time spent reviewing code, from an agent, or from an AI overall, you could... There's two patterns. One is, you could spend a bunch of time specifying things up front, the AI goes and works, and then you then go and review the AI's work, and then you're done. That's the whole task.

Or you can really chop things up. So you can specify a little bit, AI writes something, review, specify a little bit, AI writes something, review. Autocompletes all in the way of that spectrum. And still we see often the most successful people using these tools are chopping things up right now, and keeping things fairly [inaudible 00:24:28].

中文翻译:

是的。我们看到那些使用 AI 最成功的客户，在某些使用方式上其实还是相当保守的。我认为目前最成功的客户非常依赖我们的“下一处编辑预测”（next edit prediction），即你正常编码，它预测你接下来的动作。同时，他们也非常擅长缩小交给机器人的任务范围。

关于审查 AI 或智能体的工作时间，有两种模式：一种是你花大量时间预先说明，AI 去干活，然后你回来审查它的全部成果，任务结束。另一种是把任务切得很碎：说明一点点，AI 写一点点，你审查一下；再说明一点点，AI 再写一点点。自动补全就处于这个光谱的一端。我们发现，目前最成功的人通常是把任务切碎，保持紧密的控制。

[00:24:27] Lenny Rachitsky

English:

That sounds less terrible. I'm glad there's a solution here. I want to go back to you guys building Cursor for the first time. What was the point where you realized this is ready? What was a moment of, "Okay, I think this is time to put it out there, and see what happens"?

中文翻译:

这听起来没那么糟糕了。很高兴有解决方案。我想回到你们第一次构建 Cursor 的时候。什么时候你意识到它准备好了？那个“好吧，我觉得是时候把它推向世界看看会发生什么”的时刻是什么时候？

[00:24:41] Michael Truell

English:

So when we started building Cursor, we were fairly paranoid about spinning for a while, without releasing to the world. And so to begin with too, we actually... The first version of Cursor was hand-rolled. Now we use VS Code as a base, like many browsers use Chromium as a base, and hit foot off of that. To begin with, we didn't, and built the prototype of Cursor from scratch, and that involved a lot of work. We had to build our own... There were a lot of things that go into a modern code editor, including support for many different languages, and navigation support for moving amongst the language, error tracking support for things. There's things like an integrated command line, the ability to use remote servers, the ability to connect to remote servers to view and run code. And so we kind of just went on this blitz of building things incredibly quickly, building our own editor from scratch, and then also the AI components.

It was after maybe five weeks that we were living on the editor full-time, and had thrown away our previous editor, and we're using a new one. And then once it got to a point where we found it a bit useful, then we put it in other people's hands, and had this very short beta period. And then we launched it out to the world within a couple of months from the first line of code, I think it was probably three months. And it was definitely a, "Let's just get this out to people and build in public quickly." The thing that took us by surprise is we thought we would be building for a couple hundred people for a long time. And from the get-go, there was an immediate rush of interest, and a lot of feedback too. That was super helpful, we learned from that. That's actually why we switched to being based off of VS Code instead of just this hand-rolled thing. A lot of that was motivated by the initial user feedback, and then had been iterating in public from there.

中文翻译:

当我们开始构建 Cursor 时，我们非常担心闭门造车太久而不发布。所以一开始，Cursor 的第一个版本是纯手写的。现在我们以 VS Code 为基础，就像很多浏览器以 Chromium 为基础一样。但最开始我们是从零开始构建 Cursor 原型的，这涉及大量工作。我们必须构建自己的……现代代码编辑器包含很多东西：多语言支持、语言间的导航支持、错误跟踪支持。还有集成命令行、使用远程服务器的能力、连接远程服务器查看和运行代码的能力。我们当时就像闪电战一样，极其快速地构建一切，从零开始做编辑器，还有 AI 组件。

大约五周后，我们就完全搬到这个编辑器上工作了，扔掉了之前的编辑器，开始使用这个新的。一旦到了我们觉得它有点用的时候，我们就把它交到别人手里，经历了一个非常短的测试期。从写下第一行代码到向全球发布，大概只用了三个月。那绝对是一种“先推出去给人们用，快速在公开场合构建”的心态。让我们吃惊的是，我们本以为会在很长一段时间内只为几百个人服务。但从一开始，兴趣就呈爆发式增长，反馈也很多。这非常有帮助，我们学到了很多。这实际上也是为什么我们后来转向基于 VS Code 而不是继续用那个手写的东西，很大程度上是受初始用户反馈的驱动，然后我们就一直在公开迭代。

[00:26:44] Lenny Rachitsky

English:

I like how you understated the traction that you got. I think you guys went from $0 to 100 million ARR in a year, year and a half or something like that, which is historic. What do you think was the key to success of something like this? You just talked about dogfooding being a big part of it. You built it in three months, that's insane. What do you think is the secret to your success?

中文翻译:

我很喜欢你这种轻描淡写描述增长的方式。你们在一年或一年半左右的时间里从 0 增长到 1 亿美元 ARR，这是史诗级的。你认为成功的关键是什么？你刚才提到“吃自己的狗粮”是很重要的一部分。你们在三个月内就把它造出来了，这太疯狂了。你认为成功的秘诀是什么？

[00:27:12] Michael Truell

English:

The three-month version wasn't very good, and so I think it's been a sustained paranoia about, there are all of these ways in which this thing could get better. The end goal is really to invent a very new form of programming that involves automating a lot of coding, as we know today. And no matter where we are with Cursor, it feels like we're very, very far away from that end goal, there's always a lot to do. A lot of it hasn't been over rotated on that initial push, but instead is the continued evolution of the tool, and just making the tool consistently better.

中文翻译:

三个月时的那个版本其实并不好。所以我认为成功的关键在于一种持续的危机感（paranoia），总觉得这个东西还有很多可以改进的地方。我们的终极目标是发明一种全新的编程形式，实现我们今天所知的许多编码工作的自动化。无论 Cursor 现在发展到什么程度，感觉离那个终极目标还非常遥远，总是有很多事情要做。成功的很大一部分不在于最初的那次冲刺，而在于工具的持续演进，以及让它变得越来越好。

[00:27:47] Lenny Rachitsky

English:

Was there an inflection point after those three months where things just started to really take off?

中文翻译:

在那三个月之后，是否有一个转折点，让一切真正开始起飞?

[00:27:51] Michael Truell

English:

To be honest, it felt fairly slow to begin with, and maybe it comes from some impatience on our part. I think there's the overall speed of the growth which continues to take us by surprise. I think one of the things that has been most surprising too is that the growth has been fairly just consistent on an exponential, of just consistent month-over-month growth, accelerated at times by launches on our part and other things. But an exponential to begin with feels fairly slow and the numbers are really low, and so it didn't really feel off to the races to begin with.

中文翻译:

老实说，起初感觉相当缓慢，也许是因为我们有点缺乏耐心。虽然整体增长速度一直让我们感到惊讶，但最令人惊讶的是，这种增长是非常稳定的指数级增长，月复一月地稳定增长，偶尔会因为我们的产品发布或其他因素而加速。但在开始阶段，当基数很小时，指数级增长感觉很慢，所以刚开始并没有那种“起飞”的感觉。

[00:28:32] Lenny Rachitsky

English:

To me this sounds like build it and they will come actually working. You guys just built an awesome product that you loved yourselves as engineers, you put it out, people just loved it, told everyone about it.

中文翻译:

在我听来，这就像是“只要你造出来，人们就会来”的真实写照。你们作为工程师，造出了一个自己热爱的超棒产品，推向市场后，人们也非常喜欢，并奔走相告。

[00:28:42] Michael Truell

English:

It being essentially all just us, the team working on the product, and making the product good in lieu of other things one could spend one's time on. We definitely spent time on tons of other things, for instance, building the team was incredibly important, and doing things like support rotations are very important. But some of the normal things that people would maybe reach for in building the company early on, we really let those fires burn for a long time, especially when it came to things like sales and marketing.

And so just working on the product, and building a product that you like first, your team likes, and then also then adjusting it for some set of users, that can kind of sound simple, but then, as you know, it's hard to do that well. And there are a bunch of different directions one could have run in, a bunch of different product directions.

I think focus, and strategically picking the right things to build, and prioritizing effectively is tricky. I think another thing that's tricky about this domain is, it's kind of a new form of product building, where it's very interdisciplinary in that we are something in between a normal software company and then a foundation model company, in that we're developing a product for millions of people, and that side of things has to be excellent, but then also one important dimension of product quality is doing more and more on the science, and doing more and more on the model side of things in places where it makes sense. And so that element of things doing that well too has been tricky. The overall thing would note is maybe some of these things sound simple to specify, but doing them well is hard, and they're a lot of different way you can run in.

中文翻译:

基本上就是我们整个团队全身心投入到产品中，把产品做好，而不是把时间花在其他事情上。当然我们也花了很多时间在其他事情上，比如组建团队极其重要，做客服轮岗也很重要。但一些人们在创业初期通常会去做的事情，我们其实让它们“自生自灭”了很久，尤其是销售和市场营销。

所以，仅仅是专注于产品，做一个你自己喜欢、团队喜欢的产品，然后根据用户群体进行调整。这听起来很简单，但如你所知，要做好很难。有很多不同的方向可以走，很多不同的产品路径。

我认为专注、战略性地选择正确的东西去构建，并有效地确定优先级是非常棘手的。这个领域的另一个难点在于，它是一种新型的产品构建形式，非常跨学科。我们介于普通软件公司和基础模型公司之间：我们为数百万人开发产品，这方面必须做到卓越；但同时，产品质量的一个重要维度是在科学研究和模型端投入越来越多。要把这两方面都做好非常困难。总的来说，有些事情说起来简单，但做起来很难，而且有很多干扰项。

[00:30:30] Lenny Rachitsky (Sponsor Break - OneSchema)

English:

I'm excited to have Andrew Luo joining us today. Andrew is CEO of OneSchema... [Sponsor text omitted]

中文翻译:

很高兴今天 Andrew Luo 加入我们。Andrew 是 OneSchema 的 CEO……（赞助商内容略）

[00:32:05] Lenny Rachitsky

English:

What is the most counterintuitive thing you've learned so far about building Cursor, building AI products?

中文翻译:

在构建 Cursor 和 AI 产品的过程中，你学到的最反直觉的事情是什么？

[00:32:11] Michael Truell

English:

I think one thing that's been counterintuitive for us, [inaudible 00:32:14] added a little bit before, but is, we definitely didn't expect to be doing any of our own model development when we started. As mentioned, when we got into this, there were companies that were immediately from the get-go going and just focusing on training model from scratch. And we had done the calculation for what it to train before, and just knew that that was not [inaudible 00:32:36] going to be able to do. And also felt a bit like focusing one's attention in the wrong area, because there were lots of amazing models out there, and why develop all this work to replicate what other players had done. Especially on the pre-training side of things, taking a neural network that knows nothing, and then teaching it the whole internet.

And so we thought we weren't going to be doing that at all, and it seems clear to us from the start that the existing models, there were lots of things that they could be doing for us that they weren't doing, because there wasn't the right tool built for them. In fact though, we do a ton of model development, and internally, it's a big focus for us on the hiring front, and have assembled a fantastic team there.

And it's also been a big win on the product quality side of things for us. And at this point, every magic moment in Cursor involves a custom model in some way. So that was definitely counterintuitive, and surprising, and it's been a gradual thing, where there was an initial use case for training our own model, where it really didn't make sense to use any of the biggest foundation models. That was incredibly successful, moved to another use case that worked really well, and had been going from there. And one of the helpful things in doing this sort of model development is picking your spots carefully, not trying to reinvent the wheel, not trying to focus on places, and maybe where the best foundation models are excellent, but instead kind of focusing on their weaknesses, and how you can complement them.

中文翻译:

我认为对我们来说最反直觉的一点是——我之前稍微提过——我们刚开始时完全没打算做任何自主模型开发。正如所说，当我们进入这个领域时，有些公司从第一天起就专注于从零开始训练模型。我们算过训练成本，知道那不是我们能做的。而且感觉那有点放错了重点，因为外面已经有很多很棒的模型了，为什么要重复造轮子去复制别人的工作呢？尤其是预训练方面，让一个一无所知的神经网络去学习整个互联网。

所以我们本以为完全不会碰这一块。我们一开始觉得，现有模型有很多能为我们做的事还没被发掘，只是因为还没人为它们构建合适的工具。但事实上，我们现在做了大量的模型开发。在内部，这是我们招聘的一大重点，我们也组建了一支梦幻团队。

这对我们的产品质量来说也是巨大的胜利。目前，Cursor 中的每一个“奇迹时刻”在某种程度上都涉及到了自定义模型。这绝对是反直觉且令人惊讶的。这是一个循序渐进的过程：最初有一个用例，使用大型基础模型完全不划算，于是我们训练了自己的模型。结果非常成功，接着我们又应用到了另一个用例，效果也很好，就这样发展了下来。做这种模型开发的一个窍门是“精准选点”，不要试图重新发明轮子，不要在基础模型已经做得很好的地方使劲，而是专注于它们的弱点，以及你如何能补充它们。

[00:34:05] Lenny Rachitsky

English:

I think this is going to be surprising to a lot of people hearing that you have your own models. When people talk about Cursor and all the folks in the space, they would kind of call them GPT wrappers, they're just sitting on top of ChatGPT or Sonnet. What you're saying is that you have your own models, talk about just the stack behind the scenes.

中文翻译:

我想很多人听到你们有自己的模型会感到惊讶。当人们谈论 Cursor 和这个领域的其他产品时，通常会称它们为“GPT 套壳”（GPT wrappers），只是套在 ChatGPT 或 Claude Sonnet 之上。而你却说你们有自己的模型，聊聊幕后的技术栈吧。

[00:34:21] Michael Truell

English:

Yeah, of course. So we definitely use the biggest foundation models a bunch of different ways, they're really important components of bringing the Cursor experience to people. The places where we use our own models, so sometimes it's to survey a use case that a foundation model wouldn't be able to serve at all for cost or speed reasons. And so one example of that is the autocomplete side of things. And so this can be a little bit tricky for people who don't code to understand, but code is this weird form of work, where sometimes, really, the next 5, 10, 20, 30 minutes of your work is entirely predictable from looking over your shoulder.

And I would contrast this with writing. So writing, lots of people are familiar with Gmail's autocomplete, and the different forms of autocomplete that show up when you're trying to post text messages, or emails, or things like that. They can only be so helpful, because often, it's just really not clear what you're going to be writing just by looking at what you've written before. But in code sometimes, when you edit a part of a code base, you're going to need to change things, and in other parts of code base, and it's entirely clear how you're going to need to change things.

So one core part of Cursor is this really suit to autocomplete experience, where you predict the next set of that you're going to be doing across multiple files, across multiple places within a file. And making models good at that use case, one, there's a speed component of, those models need to be really fast, they need to give you a completion within 300 milliseconds. There's also this cost component of, we're running tons, and tons, and tons of molecules, every keystroke, we need to be changing our prediction for what you're going to do next. And then it's also this really specialty use case of, you need models that are really good, not at completing the next token, just a generic tech sequence, but are really good at autocompleting a series of diffs, looking at what's changed within a code base, and then creating the next set of things that are going to change, both deleted and added and all of that, and we found a ton of success in training models specifically for that task.

So that's a place where no foundation models are involved, it's kind of our own thing. We don't have a lot of labeling or branding about this in the app, power is a very core part of Cursor. And then another set of places where a user own models are to help things like Sonnet, or Gemini, or GPT, and those sit both on the inputs of those big models, and on the output. On the input side of things, those models are searching throughout a code base, try to figure out the parts of a code base to show to one of these big models. You can kind of think about this as a mini Google search that's specifically built for finding the relevant parts of the code base to show one of these big models.

And then on the output side of things, we take the sketches of the changes that these models are suggesting, you make with that code base. And then we have models that then fill in the details of, the high level thinking is done by the smartest models, they spend a few tokens on doing that, and then these smaller specialty incredibly fast models, coupled with some inference tricks, then take those high

level changes and turn them actually into full code diffs. And so it's been super helpful for pushing on quality in places where you need a specialty task, and it's been super helpful for pushing on speed, which is such an important dimension of product quality for us too.

中文翻译:

当然。我们确实在很多方面使用了最强大的基础模型，它们是为用户提供 Cursor 体验的重要组成部分。但在某些地方我们使用自己的模型，通常是因为出于成本或速度原因，基础模型根本无法胜任。

一个例子是“自动补全”（autocomplete）。对于不写代码的人来说可能有点难理解，但代码是一种奇特的工作形式：有时，你接下来 5 到 30 分钟的工作，只要看一眼你之前的操作，就完全是可以预测的。

这和写作不同。很多人熟悉 Gmail 的自动补全，或者发短信时的联想输入。它们的作用有限，因为仅看前文往往不清楚你接下来要写什么。但在代码中，当你修改了代码库的一部分，通常需要相应地修改其他部分，而且这种修改方式是非常明确的。

所以 Cursor 的核心部分之一就是这种极其强大的自动补全体验，它可以预测你在多个文件、多个位置将要进行的一系列操作。要让模型擅长这个用例，首先是速度：模型必须极快，在 300 毫秒内给出补全。其次是成本：你每敲一下键盘，我们都要运行模型并更新预测。最后是专业性：你需要模型不仅擅长预测下一个词（token），还要擅长预测一系列“代码差异”（diffs）——观察代码库发生了什么变化，然后生成下一组增删改的操作。我们在为此专门训练模型方面取得了巨大成功。

所以在这些地方没有涉及基础模型，完全是我们自己的东西。我们在 App 里没怎么宣传这一点，但它是 Cursor 的核心动力。此外，我们还有一些模型是用来辅助 Sonnet、Gemini 或 GPT 的，它们位于大模型的输入端和输出端。在输入端，这些模型负责在整个代码库中搜索，找出最相关的部分展示给大模型。你可以把它看作一个专门为大模型寻找上下文的“微型谷歌搜索”。

在输出端，大模型给出修改的草图或高层思路，然后由我们的小型专业模型（结合一些推理技巧）来填充细节，将高层指令转化为完整的代码差异。这在需要专业任务的地方极大地提升了质量，也极大地提升了速度，而速度对我们来说是产品质量极其重要的维度。

[00:37:39] Lenny Rachitsky

English:

This is so interesting. I just had Kevin Weil on the podcast, CPO of OpenAI, and he calls this the ensemble of models, that's the same way-

中文翻译:

太有意思了。我刚采访过 OpenAI 的 CPO Kevin Weil，他称之为“模型集成”（ensemble of models），这和你们的方式一样——

[00:37:46] Michael Truell

English:

Yes.

中文翻译:

是的。

[00:37:46] Lenny Rachitsky

English:

... they work, to use the best feature of each one, and to your point, the cost advantages of using cheaper models. These other models, are they based on Llama and things like that, just open source models that you guys plug into and build on?

中文翻译:

……他们也是这样运作的，利用每个模型的最佳特性，并且正如你所说，利用更便宜模型的成本优势。这些其他模型是基于 Llama 之类的吗？就是你们接入并在此基础上构建的开源模型？

[00:38:00] Michael Truell

English:

Yeah. So again, we try to be very pragmatic about the place that we're going to do this work, and we don't want to reinvent the wheel. And so starting from the very best pre-trained models that exist out there, often open source ones, sometimes in collaboration with these big model providers that don't share their weights out into the world, because the thing we care about last is the ability to read line by line, the matrix of weights that then go to give you a certain output. We just care about the ability to train these things, to post-train them. And so by and large, yes, open source models, sometimes working with the closed source providers too to tune things.

中文翻译:

是的。我们在这方面非常务实，不想重新发明轮子。所以我们会从现有的最好的预训练模型开始，通常是开源模型，有时也会与那些不公开权重的模型提供商合作。因为我们最不关心的就是逐行阅读权重矩阵，我们只关心训练和后训练（post-train）这些模型的能力。所以大体上是的，使用开源模型，有时也与闭源提供商合作进行微调。

[00:38:42] Lenny Rachitsky

English:

This leads to a discussion that a lot of AI founders always think about and investors, which is moats, and defensibility in AI. So it feels like one is custom models, is a moat in the space. How do you just think about long-term defensibility in the space, knowing there's other folks, as you said, launching constantly trying to eat your lunch?

中文翻译:

这引出了一个很多 AI 创始人和投资者都在思考的话题：AI 领域的“护城河”（moats）和防御性。感觉自定义模型就是这个领域的一种护城河。既然如你所说，总有其他人不断推出产品试图抢占你的市场，你如何看待这个领域的长期防御性？

[00:39:03] Michael Truell

English:

I think that there are ways to build in inertia and traditional moats, but I think by and large, we're in a space where it is incumbent on us to continue to try to build the best thing, and everyone in this industry. And I truly just think that the ceiling is so high that no matter what entrenchment you build, you can be leapfrogged. And I think that this resembles markets that are maybe a little bit different from normal software markets, normal enterprise markets of the past. I think one that comes to mind is the market for

search engines at the end of 1999, or at the end of the '90s and beginning of the 2000s. I think another market that comes to mind that resembles this market in many ways, it's actually just the development of the peripheral computer and many computers in the '70s, '80s, '90s.

And I think that, yes, in each of those markets, the ceiling was incredibly high, it was possible to swish. You could keep getting value for the incremental hour of a smart person's time, the incremental R&D dollar for a really long time, you wouldn't run out of useful things to build. And then in search in particular, not on the computer case, adding distribution was helpful for making the product better too, in that you could tune the algorithms, you could tune the learning based off of the data and the feedback you're getting from users. And I think that all of those dynamics exist in our market too. And so I think maybe the sad truth for people like us, but then the amazing truth for the world is, I think that there are many leapfrogs that exist, there's more useful things to build. We're a long way away from where we can compete in 5, 10 years, and it's incumbent in our state to keep that going.

中文翻译:

我认为确实有办法建立惯性和传统的护城河，但总的来说，我们所处的这个领域要求我们（以及这个行业的每个人）必须不断努力打造最好的产品。我真心认为这个领域的天花板太高了，无论你建立了什么样的防御工事，你都有可能被后来者跨越。

我觉得这类似于一些与过去普通的软件或企业市场不同的市场。我想到的一个例子是 1999 年底或 2000 年初的搜索引擎市场。另一个例子是 70、80、90 年代个人电脑和小型机的发展。

在这些市场中，天花板都极高。在很长一段时间内，你投入的每一小时聪明人的时间、每一美元研发经费，都能持续产生价值，你永远不会无事可做。特别是在搜索领域，增加分发渠道也有助于让产品变得更好，因为你可以根据用户的数据和反馈来调整算法。我认为所有这些动态也存在于我们的市场中。所以，对于像我们这样的人来说，这可能是一个残酷的事实，但对世界来说却是件好事：存在很多实现跨越的机会，还有更多有用的东西等着去建造。我们离 5 到 10 年后的竞争格局还很远，我们的职责就是保持这种进步。

[00:40:55] Lenny Rachitsky

English:

So what I'm hearing, this sounds like a lot more like a consumer sort of moat, where it's just, be the best thing consistently so that people stick with you versus creating lock-in and things like that, where they're just... Like Salesforce, where it's just contracts with the entire company, and you have to use this product.

中文翻译:

所以我听下来，这听起来更像是一种“消费者端”的护城河，即通过持续保持最佳体验让用户留下来，而不是像 Salesforce 那样通过与整个公司签订合同来创造“锁定”（lock-in），让你不得不使用这个产品。

[00:41:10] Michael Truell

English:

Yeah. I think the important thing to note is, if you're in a space where you run out of useful things to do very quickly, then that's not a great situation to be in. But if you're in a place where big investments, and having more and more great people working on the right path can keep giving you value, then you can get these economies of scale of R&D, and you can deeply work on the technology in the right direction, and get to a place where that is defensible. But yes, it is... I think there's a consumer-like tendency to it, and I really think it's just about building the best thing possible.

中文翻译:

是的。我认为重要的一点是，如果你处在一个很快就无事可做的领域，那处境会很糟糕。但如果你处在一个通过大规模投入、吸引越来越多优秀人才在正确道路上努力就能持续产生价值的领域，你就能获得研发的规模经济效应。你可以朝着正确的方向深耕技术，最终达到一个具有防御性的地位。但确实，它有一种类似消费品的倾向，核心就是打造尽可能最好的产品。

[00:41:48] Lenny Rachitsky

English:

Do you think in the future there's one winner in this space, or do you think it's going to be a world of a number of products like this?

中文翻译:

你认为未来这个领域会是“赢家通吃”，还是会存在多个类似的产品？

[00:41:53] Michael Truell

English:

I think the market is just so very big. You asked about the IDE thing early on, and one thing that I think a trip of some people that were thinking about the space is, they looked at the IDE market of the past 10 years, and they said, "Who's making money off of the editors?" It's this super fragmented space where everyone kind of has their own thing, with their own figuration, and there's one company that actually makes money off making great editors, but that company is only so big. And then the conclusion was, it was going to look like that in the future. And I think that the thing that people missed was that there was only so much you could do building an editor in the 2010s for coders, and the company that made money off of editors was doing things like making it easy to navigate around a code base, and doing some error checking and type checking for things, and having good debugging tools.

Which were all very useful, but I think that the set of things you can build for programmers, I think the set of things you can build for knowledge workers in many different areas just goes very far and very deep. The problem in front of all of us is the automation of a lot of busy work and knowledge work, and really changing all the areas of knowledge work in front of us to be much higher level and more productive.

So that was a long-winded way to say, I think the market's really, really big that we're in. I think it's much bigger than people have realized than the other building tools for developers in the past. And I think that there will be a bunch of different solutions. I think that there will be one company, to be determined if it's going to be us, but I do think that there will be one company that builds the general tool that builds almost all the world's software, and that will be a very, very generationally big business. But I think that there will be kind of niches you can occupy in doing something for a particular segment of the market, or for a very particular part of the software development life cycle. But the general programming shifts from just writing formal programming languages to something way higher level. This is the application you purchase and use to do that. I think that there will be generally one winner there, and it will be a very big business.

中文翻译:

我认为这个市场非常巨大。你之前问到 IDE 的问题，我觉得有些思考这个领域的人犯了一个错误：他们观察过去 10 年的 IDE 市场，然后问：“谁靠编辑器赚钱了？”那是一个极其碎片化的空间，每个人都有自己的配置，只有一家公司真正靠做优秀的编辑器赚到了钱，但那家公司的规模也就那样。于是人们得出结论：未来也会是这样。

但我认为人们忽略了一点：在 2010 年代，你能为程序员做的编辑器功能也就那么多。当时赚钱的公司做的是方便代码库导航、错误检查、类型检查和调试工具。这些都很有用，但我认为现在你能为程序员（以及许多领域的知识工作者）做的事情要深远得多。我们面前的问题是大量繁琐工作和知识工作的自动化，是将所有知识工作领域提升到更高层次、更高效的水平。

所以，长话短说，我认为我们所处的市场非常非常大，比人们过去认为的开发者工具市场要大得多。虽然会有很多不同的解决方案，也会有细分市场的利基产品，但我认为最终会有一家公司（是否是我们还有待观察）构建出一种通用工具，世界上几乎所有的软件都将通过它来构建。那将是一个具有划时代意义的巨型业务。当通用编程从编写正式语言转向更高层次的形式时，你购买并使用的那个应用程序，很可能只会产生一个大赢家。

[00:44:10] Lenny Rachitsky

English:

Juicy. Along those lines, it's interesting that Microsoft was actually at the center of this first, with an amazing product, amazing distribution, Copilot you said was the thing that got you over the hump of, "Wow, there could be something really big here." And it doesn't feel like they're winning, it feels like they're falling behind. What do you think happened there?

中文翻译:

真精彩。顺着这个思路，有趣的是微软最初其实处于这个领域的中心，拥有出色的产品和强大的分发能力。你刚才说 Copilot 是让你意识到“哇，这里大有可为”的契机。但现在感觉他们并没有赢，反而像是落后了。你认为发生了什么？

[00:44:34] Michael Truell

English:

I think that there are specific historical reasons why Copilot might not have lived up... So far have lived up to the expectations that some people have for it, and then I think that there are structural reasons. I think the structural reason is... And to be clear, Microsoft, in the Copilot case, obviously a big inspiration for our work, and in general, I think they do lots of awesome things, and we're users of many Microsoft products, but I think that this is a market that's not super friendly to incumbents, in that a market that's friendly to incumbents might be one where there's only so much to do, it kind of gets commoditized fairly quickly, and you can bundle that in with other products, and where the ROI between different products is quite small. And in that case, perhaps it doesn't make sense to buy the innovative solution, it makes sense to just kind of buy the thing that's bundled in with other stuff.

Another market that might be particularly helpful for incumbents is one where there's... From the get-go, you have your stuff in one place, and it's really, really excruciatingly hard to switch, and for better or for worse. I think in our case, you can try out different tools, and you can decide which product you think is better. And so that's not super friendly to incumbents, and that's more friendly to whoever you think is going to have the most innovative product. And then the specific historical reasons, as I understand them are the group of people that worked on the first version of Copilot have, by and large, gone on to do other things at other places. I think it's been a little hard to coordinate among all the different departments and parties that might be involved in making something like this.

中文翻译:

我认为 Copilot 到目前为止没有达到某些人的预期，既有特定的历史原因，也有结构性原因。结构性原因是——首先要声明，微软的 Copilot 显然是我们工作的巨大灵感来源，而且我认为他们做了很多了不起的事情，我们也是很多微软产品的用户。但这个市场对现有的巨头（incumbents）并不十分友好。

对巨头友好的市场通常是那些“能做的事情有限”的市场，产品很快就会商品化（commoditized），你可以把它和其他产品捆绑销售，不同产品之间的投资回报率（ROI）差异很小。在这种情况下，买创新的解决方案可能没意义，直接买捆绑好的就行。

另一种对巨头有利的市场是那种切换成本极高、极其难以迁移的市场。但在我们的案例中，你可以尝试不同的工具，并决定哪个产品更好。这就不利于巨头，而有利于拥有最创新产品的人。至于特定的历史原因，据我了解，开发第一版 Copilot 的那群人大多已经离职去别处了。而且在微软这样的大公司里，协调所有涉及此类产品的部门和利益方可能有点困难。

[00:46:15] Lenny Rachitsky

English:

I want to come back to Cursor. A question I like to ask everyone that's building a tool like this, if you could sit next to every new user that uses Cursor for the first time, just whisper a couple tips in their ear to be more successful, most successful with Cursor, what would be 1 or 2 tips?

中文翻译:

我想回到 Cursor。我喜欢问每一个构建此类工具的人一个问题：如果你能坐在每一个第一次使用 Cursor 的新用户旁边，在他们耳边低声说几个建议，让他们能最成功地使用 Cursor，那会是哪一两个建议？

[00:46:32] Michael Truell

English:

I think right now, and we'd want to fix this at a product level, a lot of being successful with Cursor is kind of having a taste for what the models can do, both what complexity of a task they can handle, and how much you need to specify things to that model, but having a taste for the quality of the model, and where its gaps exist, and what it can do and what it can't. And right now, we don't do a good job in the product of educating people around that, and maybe giving people some swim lanes, giving people some guidelines.

But to develop that taste, would give two tips. So one is, as mentioned before, would bias less toward, trying in one go to tell the model, "Hey, here's exactly what I want you to do." Then seeing the output, and then either being disappointed or accepting the entire thing for an entire big task. Instead what I would do is I would chop things up into bits, and you can spend basically the same amount of time specifying things overall, but chopped up more. So you're specifying a little bit, you're getting a little bit of work, you're specifying a little bit, getting a little bit of work, and not doing as much the, "Let's write a giant thing telling the model exactly what to do." I think that will be a little bit of a recipe for disaster right now.

And so biasing toward chopping things up. At the same time, and it might make sense to do this on a side project and not on your professional work, I would encourage people to, especially developers who are used to existing workflows for building software, I would encourage people to explicitly try to fall on their face, and try to discover the limits of what these models can do by being ambitious in a safe environment, like perhaps a side project, and trying to kind of go around town, use AI to the fullest. Because a lot of the time, we run into people who haven't given the AI yet a fair shake, and are underestimating its abilities. So generally biasing towards chopping things up and making things smaller, but to discover the limits of what you can do there, explicitly just try to go for broke in a safe environment, and get a taste for... You might be surprised in some of the places where the model doesn't break.

中文翻译:

我认为目前——我们也想在产品层面解决这个问题——成功使用 Cursor 的很大一部分在于培养一种对模型能力的“感觉”：它能处理多复杂的任务，你需要向它说明到什么程度，以及它的质量、短板、能做什么和不能做什么。目前我们的产品在引导用户方面做得还不够好。

为了培养这种感觉，我有两个建议：第一，正如之前提到的，不要试图一次性告诉模型：“嘿，这就是我想要你做的所有事”，然后看输出，要么失望，要么全盘接受。相反，你应该把任务切碎。你花在说明上的总时间是一样的，但要分步进行：说明一点，得到一点结果；再说明一点，再得到一点结果。不要写一大段话让模型做个大工程，那在目前往往是灾难。

所以要倾向于切碎任务。同时，我建议大家（尤其是在侧边项目而不是正式工作中）去大胆尝试，甚至故意去“碰壁”。特别是那些习惯了传统工作流的开发者，我鼓励你们在安全的环境（如个人项目）中表现得更有野心，尝试 AI 的极限。因为很多时候，我们遇到的人还没给 AI 一个公平的机会，低估了它的能力。所以，总的来说，平时要把任务切小，但为了发现极限，要在安全环境下尝试“孤注一掷”，去感受一下——你可能会惊讶于模型在某些地方竟然没有崩溃。

[00:48:45] Lenny Rachitsky

English:

What I'm essentially hearing is build a gut feeling of what the model can do, and how far it can take an idea versus just kind of guiding it along. And I bet that you need to rebuild this gut every time there's a new model launch, when it's on... I don't know, 4.0 comes out, you have to do this again. Is that generally right?

中文翻译:

我听到的核心是：建立一种关于模型能做什么、能把一个想法推进到什么程度的直觉，而不仅仅是引导它。我敢打赌，每当有新模型发布时（比如 GPT-4.0 出来），你都得重新建立这种直觉。大体上是这样吗？

[00:49:04] Michael Truell

English:

Yes. For the past few years, it hasn't been as big as I think the first experience people have had with some of these big models. This is also a problem we would hope to solve much better just for users, and take the burden off of them. But each of these things have slightly different quirks and different personalities.

中文翻译:

是的。虽然过去几年这种变化没有人们第一次接触大模型时那么剧烈，但确实如此。这也是我们希望为用户解决的问题，减轻他们的负担。但每个模型确实都有略微不同的脾气和个性。

[00:49:26] Lenny Rachitsky

English:

Along these lines, something that people are always debating tools like Cursor, are they more helpful to junior engineers, or are they more helpful to senior engineers? Do they make senior engineers 10X better? Do they make junior engineers more like senior engineers? Who do you think benefits most today from Cursor?

中文翻译:

顺着这个话题，人们一直在争论像 Cursor 这样的工具：它们是对初级工程师更有帮助，还是对高级工程师更有帮助？它们是让高级工程师变强 10 倍，还是让初级工程师变得更像高级工程师？你认为今天谁从 Cursor 中获益最多？

[00:49:43] Michael Truell

English:

I think across the board. Both of these cohorts benefit in big ways. It's a little hard to say on the relative ranking. I will say, they fall into different anti-patterns. The junior engineers we see going a little too wholesale, relying on AI for everything, and we're not yet in a place where you can kind of do that end-to-end on a professional tool, working with tens, hundreds of other people within a long-lived code base. And then the senior engineers... For many folks, it's not true for all, and we actually often... One of the ways these tools are adopted is, there's developer experience teams within companies, often those are staffed by incredibly senior people, because often, those are people who are building tools to make the rest of the engineers within an organization more productive.

And we've seen some very, very boundary pushing kind of... We've seen people who are on the front lines of really trying to adopt the technology as much as possible there. But by and large, I would say on average, as a group, the senior engineers underrate what AI can do for them, and stick to their existing workflows. And so the relative ranking is a little hard, I think they fall into different anti-patterns, but they both, by and large, yet get big benefits with these tools.

中文翻译:

我认为是全方位的。这两个群体都获益匪浅。很难说谁获益更多，但他们会陷入不同的“反模式”（anti-patterns）。初级工程师往往过于依赖 AI，想把一切都交给它，但我们还没达到能在专业工具上、在涉及成百上千人的长期代码库中实现全自动化的程度。

而高级工程师——虽然并非所有人，但通常情况下——他们往往低估了 AI 能为他们做的事，坚持原有的工作流。不过，这些工具的一种推广方式是通过公司的“开发者体验团队”，这些团队通常由极其资深的人组成，因为他们负责构建让全公司工程师更高效的工具。我们看到那里有很多挑战边界的人。总的来说，初级工程师期望太高，高级工程师期望太低。他们都从工具中获得了巨大收益，只是面临的问题不同。

[00:51:04] Lenny Rachitsky

English:

That makes absolute sense. I love that it's two ends of the spectrum, expect too much, don't expect enough. It's like the three bears allegory.

中文翻译:

完全理解。我喜欢这种光谱两端的对比：期望太高和期望不足。就像“三只小熊”的寓言一样。

[00:51:15] Michael Truell

English:

Yeah.

中文翻译:

是的。

[00:51:16] Lenny Rachitsky

English:

Yeah. Okay.

中文翻译:

好的。

[00:51:18] Michael Truell

English:

Yeah. Maybe the sort of senior, but not staff, right in the middle.

中文翻译:

是的，也许那些资深但还没到 Staff 级别的工程师正好处于中间。

[00:51:24] Lenny Rachitsky

English:

Interesting. Okay. Just a couple more questions. What's something that you wish you knew before you got into this role? If you could go back to Michael at the beginning of Cursor, which was not that long ago, and you could give him some advice, what's something that you would tell him?

中文翻译:

有意思。好的，最后还有几个问题。在担任这个角色之前，有什么是你希望自己早点知道的？如果你能回到 Cursor 刚开始时的 Michael 身边（其实也没多久以前），并给他一些建议，你会告诉他什么？

[00:51:38] Michael Truell

English:

The tough thing with this is, it feels like so much of the hard-won knowledge is tacit, and a bit hard to communicate verbally. And the sad fact of life feels like for some areas of human endeavor, you kind of do need to fall on your face to... Either need to fall on your face to learn the correct thing, or you need to be around someone who's a great example of excellence in the thing. And one area where we have felt this is hiring. I think that we actually were... So we tried to be incredibly patient on the hiring front.

It was really important to us that, both for personal reasons and also for, I think actually for the company's strategy, having a world-class group of engineers and researchers to work on Cursor with us was going to be incredibly important. Also, getting people who fit... A certain mix of intellectual curiosity and experimentation, because there can be so many new things we need to build. And then also an intellectual honesty, and maybe micro-pessimism, bluntness, because if all the noise, and... Especially as the company's grown, and the business has grown, keeping a level head I think is incredibly important too.

But getting the right group of people into the company was the thing that maybe more than anything else, apart from building the product, we really, really fussed over. We actually waited a long time to grow

the team because of that. And I think that many people you hear hired too fast, think we actually hired too slow to begin with. I think it could have been remedied, I think we could have been better at it.

And the method of recruiting that we ended up eventually falling into and working really well for us, which isn't that novel, of going after people that we think are really world-class, and recruiting them over the course of, in some cases, many years, ended up working for us in the end, but I don't think we were very good at it to begin with. And so I think that there were hard-won lessons around both who was the right profile, who actually made sense in that team, what did greatness look like, and then how to talk with someone about the opportunity, and get them excited if they really weren't looking for anything. There were lots of learnings there about how to do that well, and that took us a bit of time.

中文翻译:

这件事难就难在，很多辛苦换来的知识都是“隐性”的，很难用言语表达。生活中一个令人遗憾的事实是，在某些人类奋斗的领域，你确实需要亲自“碰壁”才能学到正确的东西，或者你需要待在一个卓越典范的身边。

我们感触最深的一个领域是招聘。我们在招聘上一直试图保持极度的耐心。对我们来说，无论是出于个人原因还是公司战略，拥有一群世界级的工程师和研究人员共同开发 Cursor 至关重要。我们需要的人要具备好奇心和实验精神，因为我们要造很多新东西；同时也要具备智识上的诚实，甚至是一点点“微观悲观主义”和直率，因为随着公司和业务的增长，保持冷静头脑极其重要。

除了打造产品，招到对的人是我们最操心的事情。为此我们等了很久才扩大团队。很多人说招人太快，但我认为我们起步时招得太慢了。我觉得这本可以改进，我们可以做得更好。

我们最终采用并证明非常有效的招聘方法其实并不新鲜：就是盯住那些我们认为世界级的人才，有时甚至会花几年的时间去跟进。但刚开始我们并不擅长这个。关于什么样的人才画像是正确的、什么样的人适合这个团队、卓越到底长什么样，以及如何与那些根本没在看机会的人交流并让他们感到兴奋，我们交了很多学费。

[00:54:12] Lenny Rachitsky

English:

What are some of those learnings for folks that are hiring right now? What's something you missed or learned?

中文翻译:

对于现在正在招聘的人来说，有哪些经验可以分享？你之前忽略了什么，或者学到了什么？

[00:54:18] Michael Truell

English:

I think to start with, maybe we actually biased a little bit too much towards looking for people who fit the archetype of well-known school, very young, had done the things that were high credential in those well-known school environments. And actually, I think found... Were lucky early on to find fantastic people who are willing to do this with us who were later careered. I think we should kind of spent a bunch of time on maybe a little bit the wrong profile to begin with, and part of that was a seniority thing. Part of that was kind of an interest and experience thing too, we have hired people who are excellent, excellent, excellent and very young, but they maybe look in some cases slightly different from being straight out of central casting.

Another lesson is just, we very much evolved our interview loop, and so now, we have a hand-rolled set of interview questions, and then core our... Core to how we interview too, is actually, we have people onsite

for two days, and do a project with us, a work test project. And that has worked really well, that increasingly you're finding that. I think how to learn about what people are interested in, and put our best foot forward, and letting them know about the opportunity when they're really not looking for anything, and have those conversations. There's definitely been... Gotten better at that over time.

中文翻译:

我想首先，我们最初可能有点过于偏向寻找那种“名校毕业、非常年轻、在名校环境里拿过高含金量证书”的典型人才。但实际上，我们很幸运地在早期发现了一些愿意加入我们的、处于职业生涯后期的优秀人才。我觉得我们起步时在错误的人才画像上浪费了不少时间。这部分是资历问题，部分是兴趣和经验的问题。虽然我们也招到了极其优秀的年轻人，但他们往往不是那种“流水线式”的名校生。

另一个教训是，我们彻底演进了面试流程。现在我们有一套手写的面试题，而核心环节是让候选人来公司实地待两天，和我们一起做一个项目，即“工作测试项目”。这效果非常好。此外，学会如何了解人们的兴趣点，在他们没看机会时以最好的姿态展示机会并进行对话，随着时间的推移，我们也做得越来越好了。

[00:55:53] Lenny Rachitsky

English:

Do you have a favorite interview question that you like to ask?

中文翻译:

你有没有最喜欢问的面试问题?

[00:55:56] Michael Truell

English:

I think this two-day work test which we thought would not scale past a few people has had surprising staying power. And the great thing about it is, it lets someone go end-to-end on it like a real project. It's not work that we use, it's canned list of projects. But it gives you two days of seeing a real work product, and it doesn't have to be incredibly time-enhancing other teams from time. You can take the time you would spend in a half day or one day onsite, and you kind of spread it out over those two days, and give someone a lot of time to do work on their projects, and so that can actually help it scale.

It helps to enforce, do you want to be around this person type test, because you are around this person for two days, a bunch of meals with them. We didn't expect that one to stick around, but that has been really, really important to our value to process, and then also important to getting people excited at, especially the very early stages of the company. Because before, people are using the product, and know about it. And when the product is comparatively not very good, really, the only thing you have going for you is a team of people that some people find special and want to be around. And the two days would give us a chance to just have this person meet us, and in some cases, hopefully get convinced that they want to throw in with us. That one was unexpected. Not exactly an interview question, but kind of like a forward interview.

中文翻译:

我认为是这个为期两天的“实战测试”。我们本以为这在公司规模扩大后就无法推行了，但它却出奇地持久。它的妙处在于让候选人像做真实项目一样端到端地完成任务。这不是我们实际要用的工作内容，而是一系列预设的项目。它让你有两天时间观察真实的产出，而且并不会占用其他团队太多时间。你可以把原本半天或一天的面试时间分散到这两天里，给候选人充足的时间去钻研。

这也有助于进行“你是否想和这个人共事”的测试，因为你们要相处两天，一起吃好几顿饭。我们没预料到这个环节能保留下来，但它对我们的评估流程至关重要，对吸引人才也至关重要，尤其是在公司早期。因为在产品还没那么出名、还没那么好用的时候，你唯一能吸引人的就是一支让人想加入的特别团队。这两天给了候选人了解我们的机会，希望能说服他们加入我们。这不完全是一个问题，更像是一种深度面试。

[00:57:29] Lenny Rachitsky

English:

The ultimate interview question. So just to be very clear about what you're describing, you give them an assignment, like, "Build this feature in our actual code base, work with the team to code it and ship it." Is that roughly right?

中文翻译:

终极面试题。为了明确一下，你描述的是给他们一个任务，比如“在我们的实际代码库中构建这个功能，与团队合作编写并发布它”，大体上是这样吗？

[00:57:40] Michael Truell

English:

Yes. So we don't use the IP, not shift end-to-end, but it's like a mock... Very often in our code base, "Here's a real mini two-day project. You're going to do it end-to-end." Largely being left alone, there's collaboration too. And then we're a pretty imprisoned company, in almost all cases, it's actually just sitting in office with us too.

中文翻译:

是的。我们不使用他们的知识产权，也不真的发布到生产环境，而是在我们的代码库中进行模拟：“这是一个真实的、为期两天的微型项目，你要端到端地完成它。”大部分时间是独立工作，也有协作。我们是一家非常强调线下办公的公司，所以几乎所有情况下，候选人都是和我们一起坐在办公室里。

[00:58:02] Lenny Rachitsky

English:

And you've been saying that this has scaled to even today, so how big are you guys at this point?

中文翻译:

你说这个流程一直延续到了今天，那你们现在规模有多大？

[00:58:07] Michael Truell

English:

So we are going on 60 people.

中文翻译:

我们快到 60 人了。

[00:58:10] Lenny Rachitsky

English:

So small for the scale and impact. I was thinking it'd be a lot larger than that.

中文翻译:

相对于你们的规模和影响力，这真的很精简。我以为会比这大得多。

[00:58:15] Michael Truell

English:

Yeah.

中文翻译:

是的。

[00:58:16] Lenny Rachitsky

English:

And I imagine the largest percent is engineers?

中文翻译:

我猜大部分是工程师?

[00:58:19] Michael Truell

English:

Yeah. To be clear, a big part of the work ahead of us is building a group of people that is bigger, and awesome, and can continue to make the product better, and the service we give to customers better. And so you don't plan to stay that small for longer, wouldn't hope so. But part of the reason that that number is small is, the percentage of engineering and research and design is very high within the company, and so many software companies when they have roughly 40 engineers would be over 100 people, because there's lots of operational work, and often, they're very, very sales-led from the get-go, and that's just quite labor-intensive. And here, we started from a place of being incredibly lean in product-led, and we now serve lots of our market customers, and it built that out, but there's much more to do there.

中文翻译:

是的。明确地说，我们未来的重要工作是组建一支更大、更棒的团队，持续改进产品和客户服务。所以我们不打算一直保持这么小的规模。但人数之所以少，是因为公司内部工程、研究和设计的比例非常高。很多软件公司如果有 40 个工程师，总人数可能超过 100 人，因为有大量的运营工作，而且通常从一开始就是销售驱动的，那非常耗费人力。而我们是从极度精简的产品驱动模式开始的，虽然现在也服务很多市场客户并建立了相应团队，但还有很多工作要做。

[00:59:10] Lenny Rachitsky

English:

A question I wanted to ask you, there's so much happening in AI, there's things launching every... There's newsletters, many newsletters, whose entire function is to tell you what is happening in AI every single day. Running a company that's at the center, the white-hot center of this space, how do you stay focused, and how do you help your team stay focused, and heads down, and just build and not get distracted by all these shiny things?

中文翻译:

我想问你一个问题：AI 领域发生了太多事情，每天都有新东西发布。有很多时事通讯专门告诉你每天 AI 领域发生了什么。作为一家处于这个领域最核心、最白热化地带的公司负责人，你如何保持专注？你如何帮助团队保持专注、埋头苦干，而不被这些眼花缭乱的新鲜事分散注意力？

[00:59:35] Michael Truell

English:

I think hiring is a big part of it, and if you get people with the right attitude. All of this should be asterisked in, I think we're doing well there, I think that we'd probably be doing better there too, and it's something that we should probably talk even more about as a company. But I think that hiring people with the right disposition, people who are less focused on external validation, more focused on building something really great, more focused on doing really high quality work, and people who are just generally level-headed, and maybe the highs aren't very high, the lows aren't very low. I think hiring can get you through a lot here, and I think that's actually a learning throughout the company, is that for any... You need process, you need hierarchy, you need lots of things, but for any kind of organizational tool that you're introducing into a company, the result you're looking to get from that tool also... You can go pretty far on hiring people with the right behaviors that you want to resolve from that for organizational thing.

And the specific example that comes to mind is, we've been able to get away with not a ton of process yet on the engineering front, and I think we need a little bit more process, but for our size, not a ton of process, by hiring people who I think are really excellent. One is hiring people that are level-headed. I think two is just talking about it a lot. I think three is hopefully leading by example. And for us personally, we've since 2021, 2022 been professionally working on this, and been working on AI, and we've just seen a sea change of the comings and goings of various technologies and ideas of... If you're to transport yourself back to end of 2021, beginning of 2022, this is GPT-3, Instruct GPT doesn't exist, there's no Dolly, there's no stable diffusion. And then we've gone through all of those image technologies existing, ChatGPT and that rise, and GPT-4, all of these new models, all these different modalities, all the video stuff, and only a very small number of these things really kind of affects the business.

So I think we've kind of just built up a little bit of an immune system, and know when an event comes around that actually is really going to matter for us. This dynamic too of there being lots, and lots, and lots of chatter, but then maybe only a few things that really matter, I think has been mirrored in AI over the last decade, where there have been so many papers on deep learning in academia, so many papers on AI in academia, then the amazing thing is there are really a lot of... A lot the progress of AI can be attributed to some very simple elegant ideas that have stayed around, and the vast majority of ideas that have been put out there haven't had staying power, and haven't mattered a ton. And so the dynamic is a little bit mirrored in the evolution of deep learning as a field overall.

中文翻译:

我认为招聘是很大一部分原因，要招到态度正确的人。当然，这一切都要打个星号，我觉得我们做得不错，但可能还可以做得更好。我认为要招那些不那么看重外部认可、更专注于打造伟大产品、更专注于高质量工作的人，以及那些头脑冷静、情绪波动不大的人。招聘能解决很多问题。这是我们公司的一个心得：虽然你需要流程、层级等组织工具，但如果你招到了具备正确行为模式的人，你就能在很大程度上替代这些工具。

一个具体的例子是，在工程方面，我们目前还没有太多的流程（虽然规模大了可能需要一点），但靠着招到极其优秀的人，我们目前运转良好。第一是招冷静的人，第二是多沟通，第三是希望以身作则。对我们个人来说，从 2021、2022 年起我们就专业从事 AI 工作，见证了各种技术和想法的潮起潮落。回想 2021 年底，只有 GPT-3，没有 Instruct GPT，没有 DALL-E，没有 Stable Diffusion。后来经历了图像技术爆发、ChatGPT 崛起、GPT-4、各种模态、视频生成……但真正影响业务的只有极少数。

所以我们建立了一套“免疫系统”，知道什么时候发生的事件才是真正重要的。这种“喧嚣很多，但重要的很少”的动态，其实在过去十年的 AI 学术界也是如此：有无数关于深度学习的论文，但真正推动 AI 进步的往往是少数几个简单、优雅且经得起时间考验的想法。这种动态也反映在深度学习作为一个领域的整体演进中。

[01:02:33] Lenny Rachitsky

English:

Last question. What do you think people still most misunderstand, or maybe don't fully grasp about where things are heading with AI in building in the way the world will change?

中文翻译:

最后一个问题。你认为人们对于 AI 的发展方向、构建方式以及世界将如何改变，目前最普遍的误解或尚未完全理解的是什么？

[01:02:46] Michael Truell

English:

People are still a little bit occupied too much, either end of a spectrum of it's all going to happen very fast, and this is all bluster, and hype, and snake well, and I think we're in the middle of a technology shift that's going to be incredibly consequential. I think it's going to be more consequential than the internet, I think it's going to be more consequential than any shift in tech that we've seen since the advent of computers. And I think it's going to take a while, and I think it's going to be a multi-decade thing, and I think many different groups will be consequential in pushing it forward.

To get to a world where computers can increasingly do more, and more, and more for us, there's all of these independent problems that need to be knocked down, and progress needs to be made on them, and some of those are on the science side of things of getting these models to understand different types of data, be faster, cheaper, smarter, conform to the modalities that we care about, take actions in the real world. And then some of it's on how we're going to work with them, and what's the experience that a human should actually be seeing and controlling on a computer, and working with these things.

But I think it's going to take decades. I think that there's going to be lots of amazing work to do. I think that also, one of the most... A pattern of a group that I think will be especially important here, not to talk our own book, but I think is the company that works on automating and augmenting a particular area of knowledge work, builds both the technology under the surface for that, integrating the best parts from providers, sometimes doing it in-house, and then also builds the product experience for that. I think people who do that, and... We're trying to do it in software, people do that in other areas, I think those folks will be really, really, really consequential. Not just for the end value that users see, but then I think as they get to scale, they'll be really important for pushing forward the technology, because I think they'll be able to build... The most successful of them will be able to build very, very big businesses. So, excited to see the rise of other companies like that in other areas.

中文翻译:

人们仍然有点过于纠结于光谱的两端：要么认为一切都会发生得极快，要么认为这全是吹嘘、炒作和骗局。我认为我们正处于一场影响极其深远的技术变革之中。我认为它比互联网的影响更深远，比自计算机诞生以来我们见过的任何技术变革都更深远。但这需要时间，这将是一个长达数十年的过程，需要许多不同的群体共同推动。

为了让计算机能为我们做越来越多的事，有很多独立的问题需要被攻克。有些是科学层面的：让模型理解不同类型的数据，变得更快、更便宜、更聪明，适应我们关心的模态，并在现实世界中采取行动。有些是交互层面的：人类应该如何在计算机上看到、控制并与这些东西协作。

我认为这需要几十年。会有很多了不起的工作要做。此外，我认为一种特别重要的模式（不是为了自夸）是：那些致力于自动化和增强特定知识工作领域的公司，它们既构建底层的技术（整合供应商的精华或自主研发），又构建相应的产品体验。我们试图在软件领域这样做，其他人可以在其他领域这样做。这些人将产生极其深远的影响，不仅是为用户创造价值，而且随着规模扩大，他们将成为推动技术进步的重要力量，并建立起极其庞大的业务。我很期待看到其他领域也出现类似的公司。

[01:04:59] Lenny Rachitsky

English:

I know you guys are hiring. For folks that are interested in, "Hey, I want to go work here, and build this sort of stuff." What kind of roles are you looking for right now? Anyone specifically you're trying... Any roles you're most excited about filling ASAP? What should people know if they're curious?

中文翻译:

我知道你们正在招聘。对于那些感兴趣说“嘿，我想去那里工作，构建这类东西”的人，你们现在在寻找什么样的职位？有没有什么特别急需人才的岗位？如果大家感兴趣，应该了解些什么？

[01:05:12] Michael Truell

English:

There are so many things that this group of people need to do that we are not get equipped to do. Generic across the board, first of all, and so if you don't think we have a role for something, maybe you should reach out, that won't actually be the case. And maybe we can actually learn from you, and decide that we need something that we weren't yet aware of. But by and large, I think that two of the most important things for us to do this year are have the best product in the space, and then grow it. And we're kind of in this land grab mode, where almost everyone in the world is either using no tool like ours, or they're using one that's maybe developing less quickly. So growing Cursor too is a big goal, and I would say, especially always on the hunt for folks who... Excellent engineers, designers, researchers, but then folks all across the business side too.

中文翻译:

我们有很多想做的事情，但目前的人手还远远不够。首先，我们全方位招人。如果你觉得我们没有适合你的职位，也许你也应该联系我们，事实可能并非如此。也许我们能从你身上学到东西，并意识到我们需要一个之前没想到的岗位。总的来说，我们今年最重要的两件事是：打造该领域最好的产品，并扩大它的规模。我们现在处于一种“圈地模式”，世界上几乎所有人要么还没用过此类工具，要么在用一个发展较慢的工具。所以扩大 Cursor 的影响力是一个大目标。我们一直在寻找优秀的工程师、设计师、研究员，以及业务端的各类人才。

[01:06:13] Lenny Rachitsky

English:

I can't help but ask this question now that you talk about engineers, there's this question of just, "AI's going to write all our code." But everyone's still hiring engineers like crazy. All the foundational models, so many open roles.

中文翻译:

既然谈到了工程师，我忍不住想问：现在有一种说法是“AI 将编写我们所有的代码”，但大家仍然在疯狂招聘工程师。所有的基础模型公司都有那么多空缺职位。

[01:06:28] Michael Truell

English:

Yeah. We're not out there tooting the horn of, people can learn to code.

中文翻译:

是的。我们并没有在那儿宣扬“人们不用学写代码了”。

[01:06:29] Lenny Rachitsky

English:

Do you think there's going to be an inflection point of engineering roles start to slow down? I know this is a big question, but just... Do you see engineers being more and more needed across all these companies, or do you think at some point there's all these Cursor agents running building for us?

中文翻译:

你认为会出现一个工程师需求开始放缓的转折点吗？我知道这是一个宏大的问题，但你认为各家公司会越来越需要工程师，还是说在某个时间点，会有无数 Cursor 智能体在替我们构建一切？

[01:06:45] Michael Truell

English:

Again, we have the view that there's this both long messy middle of it not jumping to a, just you step back, and you ask for all your stuff to be done, and you have your engineering department. And very much, you want to evolve from programming as it exists today, we want humans to be in the driver's seat, and we think even in the end state, that's giving folks control over everything is really important, and you will need professionals to do that, and decide what the software looks like.

So both I think that, yes, engineers are definitely needed. I think that engineers will be able to do much more. I think the demand for software is very lasting, which is not the most novel thing, but I think it's kind of crazy to think about how expensive and labor-intensive it is to build things that are pretty simple and easy to specify, or it would look like it to the outside observer, and just how hard those things are to do right now.

All of the stuff that exists right now that's justified by the cost and demand that we have now, if you could bring that down by [inaudible 01:07:56], I think you would have tons, and tons, and tons of more stuff that we could do in our computers, tons more tools. And I've felt this, where... One of my early jobs actually was working for a biotechnology company, and it was building internal tools for them, and the off-the-shelf tools that existed were horrible, and did not fit their use case at all. And then the internal

tools I was building, there was definitely a ton of demand there for things that could be built, and that far outstripped just the things that I could build in the time that I was with them.

The physics of working on computers are so great, you should be able to basically just move everything around, do everything that you want to do. There's still so much friction, I think that there's much more demand for software than what we can build today with things costing like a blockbuster movie to make simple productivity software. And so I think long into the future, yes, there will actually be more demand for engineers.

中文翻译:

再次强调，我们认为会有一个漫长且复杂的过渡期，不会直接跳跃到“你往后一躺，下个指令，工程部门就帮你搞定一切”的状态。我们希望从现有的编程模式进化，让人类掌握主动权。我们认为即使在最终状态，赋予人们对一切的控制权也是极其重要的，你需要专业人士来操作并决定软件的样子。

所以，是的，工程师绝对是需要的。我认为工程师将能做更多的事情。对软件的需求是持久的，这虽然不是什么新鲜观点，但想想现在构建一些看起来很简单、很容易描述的东西竟然如此昂贵且耗费人力，这确实很疯狂。

如果能大幅降低构建成本，我认为我们在计算机上能做的事情、能拥有的工具会增加成千上万倍。我有过切身体会：我早期的工作之一是为一家生物技术公司构建内部工具。当时市面上的现成工具非常糟糕，完全不符合他们的需求。而我构建内部工具的需求量极大，远远超出了我当时能完成的范围。

在计算机上工作的物理特性非常棒，你理应能随心所欲地移动和构建任何东西。但现在摩擦力仍然很大，构建简单的生产力软件成本高得像拍大片。所以我认为在遥远的未来，对工程师的需求实际上会更多。

[01:08:51] Lenny Rachitsky

English:

Is there anything that we didn't cover that you wanted to mention? Any last nugget wisdom you wanted to leave listeners with? You could also say no, because we've done a lot.

中文翻译:

还有什么我们没聊到但你想提的吗？或者想留给听众的最后一点智慧锦囊？你也可以说没有，因为我们已经聊了很多了。

[01:09:00] Michael Truell

English:

We think a lot about how you set up a team to be able to make new stuff, in addition to continuing to improve the stuff that you have right now. And I think if we were to be successful, IDE is going to have to change a ton, [inaudible 01:09:18] looks like is going to have to change a ton going into the future. And if you look around, the companies we respect, there are definitely examples of companies that have continued to really ride the wave of many leapfrogs, and continue to actually push the frontier. But they're kind of rare too, it's a hard thing to do. So part of that is just kind of thinking about the thing, and trying to reflect on it in our good days, and the first principle side of things, part of it's also trying to get in and study past examples of greatness here, and that's something that we think about a lot too.

中文翻译:

我们经常思考如何组建一支既能持续改进现有产品，又能创造全新事物的团队。我认为如果我们想成功，IDE 的形态和未来的样子必须发生巨变。环顾四周，我们尊敬的那些公司中，确实有一些能持续抓住多次跨越式发展

的机会并不断推向最前沿。但这很罕见，也很难做到。所以，部分工作是思考事物的本质，从第一性原理出发进行反思；另一部分是研究过去那些伟大的先例。这也是我们经常思考的问题。

[01:10:00] Lenny Rachitsky

English:

Yeah. Yeah. Before we started recording, you had all these books behind you, and I was like, "What's that over there?" It's the history of some old computer company that was influential in a lot of ways that I've never heard of. And I think that says a lot about you of, where a lot of this innovation comes from, is studying the past, and study history, and what's worked and what hasn't.

中文翻译:

是的。在我们开始录音前，我看到你身后有很多书，我问：“那是什么？”那是某家我从未听说过但在很多方面都很有影响力的老牌计算机公司的历史。我觉得这很能说明你的风格：很多创新其实源于对过去的研究，研究历史，研究什么是有效的，什么是无效的。

[01:10:19] Lenny Rachitsky

English:

Okay. Where can folks find you online if they want to reach out and maybe apply? You said that there may be roles they may not even be aware of, where do they go find that, and then how can listeners be useful to you?

中文翻译:

好的。如果大家想联系你或者申请职位，可以在哪里找到你？你提到可能有一些大家还没意识到的岗位，他们去哪里查看？听众们能为你提供什么帮助？

[01:10:28] Michael Truell

English:

Yeah. If folks are interested in working on this stuff, would love to speak, they can find... If they go to cursor.com, they can kind of both find the product and find out how to reach us.

中文翻译:

好的。如果大家有兴趣从事这方面的工作，我很乐意交流。大家可以访问 cursor.com，在那里既可以找到产品，也可以找到联系我们的方式。

[01:10:41] Lenny Rachitsky

English:

Easy. Michael, thank you so much for being here. This was incredible.

中文翻译:

太棒了。Michael，非常感谢你能来。这次对话太精彩了。

[01:10:44] Michael Truell

English:

It was wonderful. Thank you.

中文翻译:

非常愉快。谢谢。

[01:10:46] Lenny Rachitsky

English:

Bye, everyone. Thank you so much for listening. If you found this valuable, you can subscribe to the show on Apple Podcasts, Spotify, or your favorite podcast app... [Closing remarks omitted]

中文翻译:

再见，各位。非常感谢收听。如果你觉得有价值，可以在 Apple Podcasts、Spotify 或你喜欢的播客应用上订阅本节目……（结语略）