VARUN MOHAN

LENNY'S PODCAST

BILINGUAL TRANSCRIPT

ORIGINAL BY

Lenny Rachitsky

@lennysan • x.com/lennysan

ANALYSIS BY

@Penny777 • x.com/penny777

Generated: 2026-01-16

Varun Mohan - 双语对照

This is the complete bilingual transcript for Lenny’ s Podcast featuring Varun Mohan (CEO of Windsurf/Codeium).

[00:00:00] Varun Mohan

English:

A lot of the bets we're making inside the company are for things that are not three, four weeks away. We should be cannibalizing the existing state of our product every six to 12 months. Every six to 12 months, it should make our existing product look silly. It should almost make the form factor of existing product look dumb.

中文翻译:

我们在公司内部做的很多豪赌，都不是为了三四周后就能实现的目标。我们应该每隔 6 到 12 个月就对现有产品进行一次“自我蚕食”（自我颠覆）。每过 6 到 12 个月，我们就应该让现有的产品显得过时，甚至让现有产品的形态看起来很笨拙。

[00:00:13] Lenny Rachitsky

English:

How do you know when it's time to hire someone?

中文翻译:

你如何判断什么时候该招人了？

[00:00:16] Varun Mohan

English:

I want the company to almost be like this dehydrated entity. Every hire is like a little bit of water, and we only go back and hire someone when we're back to being dehydrated.

中文翻译:

我希望公司几乎像是一个“脱水”的实体。每一次招聘就像是注入一点点水，只有当我们再次回到脱水状态时，我们才会再去招人。

[00:00:24] Lenny Rachitsky

English:

Any other there skills you think people should be investing more in with the rise of AI building more and more of our products?

中文翻译:

随着 AI 越来越多地参与产品构建，你认为人们还应该在哪些技能上投入更多精力?

[00:00:29] Varun Mohan

English:

The engineers are now able to produce more technology. The ROI of building technology has actually gone up. This actually means you hire more. The best thing to do is just get your hands dirty with all of these products. You could be a force multiplier to your organization in ways in which they never even anticipated.

中文翻译:

工程师现在能够产出更多的技术成果。构建技术的投资回报率（ROI）实际上提高了。这其实意味着你应该招更多的人。最好的做法就是亲自动手尝试所有这些产品。你可以通过组织从未预料到的方式，成为公司的“力量倍增器”。

[00:00:47] Lenny Rachitsky

English:

Today my guest is Varun Mohan. Varun is the co-founder and CEO of Windsurf, which has quickly become one of people's favorite AI coding tools, and is basically the main competitor to Cursor with over 1 million users four months in. In our conversation, Varun shares what makes Windsurf unique, why they decided to invest heavily in enterprise sales very early in their history, why agency is going to be the most important skill for engineers and product builders to build, also the story of how they started out as a GPU infrastructure company and realized there was a much bigger opportunity up the stack and the two pivots that got them to where they're today.

中文翻译:

今天的嘉宾是 Varun Mohan。Varun 是 Windsurf 的联合创始人兼 CEO。Windsurf 迅速成为了人们最喜爱的 AI 编程工具之一，基本上是 Cursor 的主要竞争对手，上线四个月用户就突破了 100 万。在我们的对话中，Varun 分享了 Windsurf 的独特之处，为什么他们在发展初期就决定大力投入企业级销售，为什么“自主性”（Agency）将成为工程师和产品构建者最重要的技能。他还讲述了他们如何从一家 GPU 基础设施公司起步，随后意识到应用层（Up the stack）有更大的机会，以及经历两次转型后如何走到今天的故事。

[00:01:20] Lenny Rachitsky

English:

He also gives a live demo, advice for being successful at Windsurf, and so much more. There's so much to learn about where things are heading for engineers and product builders in general in this conversation. And I'm really excited to bring it to you. Thank you to everyone on LinkedIn and Twitter and my newsletter community for suggesting great questions to dig into with Varun. If you enjoy this podcast, don't forget to subscribe and follow it in your favorite podcasting app or YouTube. Also, if you become a yearly subscriber of my newsletter, you get a year free of Perplexity Pro, Notion Plus, Linear, Granola, and Superhuman. Check it out at lennysnewsletter.com. With that, I bring you Varun Mohan.

中文翻译:

他还进行了一场现场演示，分享了在 Windsurf 取得成功的建议等等。这次对话中有很多关于工程师和产品构建者未来走向的干货。我非常激动能把这些内容带给大家。感谢 LinkedIn、Twitter 和我邮件通讯社区的每一位朋友，感谢你们提供的精彩问题。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，如果你成为我邮件通讯的年度订阅者，你将免费获得一年的 Perplexity Pro、Notion Plus、Linear、Granola 和 Superhuman。欢迎访问 lennysnewsletter.com 查看。下面，让我们欢迎 Varun Mohan。

[00:01:59] Lenny Rachitsky (Ad: Brex)

English:

This episode is brought to you by Brex, the financial stack used by one in every three US venture-backed startups. Brex knows that nearly 40% of startups fail because they run out of cash. So they built a banking experience that focuses on helping founders get more from every dollar. It's a stark difference from traditional banking options that leave a startup's cash sitting idle while chipping away at it with fees. To help founders protect cash and extend runway, Brex combined the best things about checking, treasury, and FDIC insurance in one powerhouse account. You can send and receive money worldwide at lightning speed. You can get 20X the standard FDIC protection through program banks. And you can earn industry-leading yield from your first dollar while still being able to access your funds anytime. To learn more, check out Brex at brex.com/banking-solutions.

中文翻译:

本期节目由 Brex 赞助。全美每三家获风投支持的初创公司中就有一家在使用 Brex 的金融栈。Brex 深知近 40% 的初创公司倒闭是因为现金耗尽。因此，他们打造了一种专注于帮助创始人提高每一美元利用率的银行体验。这与传统银行截然不同，传统银行会让初创公司的现金闲置，同时还收取各种费用。为了帮助创始人保护现金并延长生存周期（Runway），Brex 将支票账户、国库券和 FDIC 保险的优势整合到一个强大的账户中。你可以以极快的速度在全球范围内收付款，通过合作银行获得 20 倍于标准额度的 FDIC 保障，并从第一美元起赚取行业领先的收益，同时还能随时提取资金。欲了解更多信息，请访问 brex.com/banking-solutions。

[00:02:57] Lenny Rachitsky (Ad: Productboard)

English:

This episode is brought to you by Productboard, the leading product management platform for the enterprise. For over 10 years, Productboard has helped customer-centric organizations like Zoom, Salesforce, and Autodesk build the right products faster. And as an end-end platform, Productboard seamlessly supports all stages of the product development lifecycle, from gathering customer insights to planning a roadmap, to aligning stakeholders, to earning customer buy-in, all with a single source of truth. And now, product leaders can get even more visibility into customer needs with Productboard Pulse, a new voice of customer solution. Built-in intelligence helps you analyze trends across all of your feedback and then dive deeper by asking AI your follow-up questions. See how Productboard can help your team deliver higher-impact products that solve real customer needs and advance your business goals. For special offer and free 15-day trial visit productboard.com/lenny.

中文翻译:

本期节目由 Productboard 赞助，它是领先的企业级产品管理平台。10 多年来，Productboard 帮助了 Zoom、Salesforce 和 Autodesk 等以客户为中心的企业更快地构建正确的产品。作为一个端到端的平台，Productboard 无缝支持产品开发生命周期的所有阶段——从收集客户洞察到规划路线图，再到协调利益相关者和赢得客户认可，所有这些都基于单一的事实来源。现在，产品负责人可以通过 Productboard Pulse（一种全

新的客户之声解决方案）更清晰地了解客户需求。内置的智能功能可帮助你分析所有反馈中的趋势，并通过向 AI 提出后续问题进行深入探讨。了解 Productboard 如何帮助你的团队交付更高影响力的产品，解决真实的客户需求并推进业务目标。访问 productboard.com/lenny 获取特别优惠和 15 天免费试用。

[00:04:00] Lenny Rachitsky

English:

Varun, thank you so much for being here and welcome to the podcast.

中文翻译:

Varun，非常感谢你能来，欢迎来到我们的播客。

[00:04:03] Varun Mohan

English:

Buddy, thanks for having me. A long time listener.

中文翻译:

老兄，谢谢邀请。我是你的老听众了。

[00:04:05] Lenny Rachitsky

English:

Oh, I really appreciate that. I'm so excited to have you here. I feel like just you guys have become this overnight success, which is definitely not an overnight success, but I feel like I've been hearing about Windsurf more and more as people's favorite AI tool. And I just don't think people know the story behind Windsurf, behind Codeium, the company that you built. So I thought it'd be good to maybe just start there and have you just briefly share the history of Codeium and how Windsurf emerged out of Codeium.

中文翻译:

噢，非常感谢。我也很兴奋你能来。感觉你们就像是一夜成名，虽然肯定不是真的“一夜”之间，但我感觉听到 Windsurf 被称为人们最喜欢的 AI 工具的频率越来越高。我觉得大家可能并不了解 Windsurf 背后，也就是你创办的 Codeium 这家公司的故事。所以我想，也许可以从这里开始，请你简要分享一下 Codeium 的历史，以及 Windsurf 是如何从 Codeium 中诞生的。

[00:04:31] Varun Mohan

English:

Yeah. So the company was actually started close to four years ago. As you know, AI coding was not a thing four years ago. ChatGPT was not out four years ago. At the time, we actually started out building GPU virtualization and compiler software. Before this, I worked in autonomous vehicles. My co-founder, who I had known since middle school, worked on AR/VR at Meta. And for us, we believe deep learning would touch many, many industries. It wouldn't just touch autonomous vehicles. It would touch financial services, defense, healthcare. And we believe these applications were hard to build, these deep learning applications. So we made it possible for you to effectively run these complex applications on computers

without GPUs, and we would handle all the complexity of being able to actually run the workload on the GPUs for you. And we were able to optimize these workloads a ton.

中文翻译:

是的。这家公司实际上成立于将近四年前。如你所知，四年前 AI 编程还不是个事儿，ChatGPT 也没问世。当时，我们最初是做 GPU 虚拟化和编译器软件的。在此之前，我从事自动驾驶汽车行业。我的联合创始人（我从中学就认识他）在 Meta 从事 AR/VR 工作。对我们来说，我们相信深度学习将触及许多行业，而不仅仅是自动驾驶。它会影响金融服务、国防、医疗保健。我们认为这些深度学习应用很难构建。所以我们让用户能够在没有 GPU 的计算机上有效地运行这些复杂的应用程序，而我们负责处理在 GPU 上运行工作负载的所有复杂性。我们能够极大地优化这些工作负载。

[00:05:20] Varun Mohan

English:

And in the middle of 2022 rolled around and we had a couple million in revenue and we were managing upwards of 10,000 sort of GPUs. We had eight people. At the time, we were free cash flow positive. But I think what we felt was once these generative models started to get very good, we sort of felt a lot of what we built was not as valuable. And this was a very, very hard moment for us at the company. We were only eight people at the time, but we felt, "Hey, would people be training these very bespoke sentiment classifier models anymore that were very, very custom models? Or would they just ask GPT-N, is this a positive or a negative sentiment?" Probably it's going to be the latter, right? And in a world in which everyone was going to run generative AI models, why would an infrastructure company be a differentiator? Because everyone is going to run the same kind of infrastructure down the line.

中文翻译:

到了 2022 年中期，我们已经有了几百万美元的收入，管理着超过 1 万个 GPU。当时我们只有 8 个人，而且现金流已经是正的了。但我们感觉到，一旦这些生成式模型开始变得非常强大，我们构建的很多东西价值就没那么大了。这对公司来说是一个非常艰难的时刻。虽然当时只有 8 个人，但我们想：“人们还会去训练那些非常定制化的情感分类模型吗？还是直接问 GPT-N 这段话是正面还是负面的？”大概率是后者，对吧？在一个每个人都要运行生成式 AI 模型的世界里，为什么一家基础设施公司能成为差异化竞争点？因为到头来大家运行的基础设施都会大同小异。

[00:06:06] Varun Mohan

English:

So instead what we decided to do was we believe generative AI was almost going to be like the next internet. And in that case, what we should go out and do is build the next great apps like Google, like Amazon. And we vertically integrated and actually took our infrastructure, our inference infrastructure to go out and build Codeium at the time. And at that time, we were early adopters of GitHub Copilot and we thought the coding space was going to get tremendously disrupted in the next coming years. So we actually took our infrastructure, we ran our own models in massive scale. We even trained our own models.

中文翻译:

所以我们决定，既然我们相信生成式 AI 几乎会成为下一个互联网，那么我们应该去做的是构建像 Google、Amazon 那样的下一代伟大应用。于是我们进行了垂直整合，利用我们的基础设施（推理基础设施）去构建了当时的 Codeium。那时我们是 GitHub Copilot 的早期用户，我们认为编程领域在未来几年将发生翻天覆地的变化。所以我们利用自己的基础设施，大规模运行我们自己的模型，甚至训练我们自己的模型。

[00:06:35] Varun Mohan

English:

In the very beginning it was very, very simple. It was purely an autocomplete model, which basically means that as the user was typing, we'd complete the next one or two or three or four lines of code. But we provided the product entirely for free in all the IDEs that developers coded. That meant to VSCode, JetBrains, Eclipse, Visual Studio, Vim, Emacs. And the reason why we were able to build it for free was because of our infrastructure background. We were able to optimize these workloads a ton.

中文翻译:

最开始它非常简单，纯粹是一个自动补全模型，也就是说当用户打字时，我们会补全接下来的一两行或三四行代码。但我们在开发者使用的所有 IDE（集成开发环境）中完全免费提供这个产品，包括 VSCode、JetBrains、Eclipse、Visual Studio、Vim、Emacs。我们之所以能免费提供，是因为我们的基础设施背景，我们能够极大地优化这些工作负载。

[00:07:04] Varun Mohan

English:

I guess very quickly after that, some large businesses also wanted to work with us. And we built out this enterprise motion to work with these large companies like Dell, JPMorgan Chase. And for them the bigger thing wasn't just, "Hey, could we autocomplete code or could we chat with the code base?" It was, "Could you offer us a secure offering that was also personalized to all the private data inside the company?" So we took our infrastructure and made it so that we invested a ton in making sure that we deeply understood these large companies code bases. And that's what we were working on until six months ago.

中文翻译:

在那之后很快，一些大企业也想和我们合作。于是我们建立了企业级业务，与戴尔（Dell）、摩根大通（JPMorgan Chase）等大公司合作。对他们来说，重点不仅在于“能不能自动补全代码”或“能不能和代码库聊天”，而是“你能不能提供一个安全的方案，并且能针对公司内部的所有私有数据进行个性化定制？”所以我们利用基础设施，投入大量精力确保我们能深度理解这些大公司的代码库。这就是我们直到六个月前一直在做的事情。

[00:07:33] Varun Mohan

English:

It's not that we've stopped working on that, but basically what we realized six months ago was we were getting limited by the IDEs that we were already working in. So VSCode, which is a very popular IDE, had a ceiling for the AI capabilities, we could showcase our users. And because of that, we decided to go out and fork VSCode and build our own IDE with some of these new agentic capabilities. And over time in the last couple of years, the model capabilities have also been growing exponentially year over year. And that's sort of where we are right now. I skipped a lot of pieces there, but that's what we're landed.

中文翻译:

并不是说我们停止了那项工作，而是六个月前我们意识到，我们受到了现有 IDE 的限制。比如 VSCode 这种非常流行的 IDE，它能展示给用户的 AI 能力是有天花板的。因此，我们决定基于 VSCode 进行分叉（Fork），构建我们自己的 IDE，并加入一些新的“智能体”（Agentic）能力。在过去的几年里，模型能力也在逐年呈指数级增长。这就是我们现在的处境。我跳过了很多细节，但大体情况就是这样。

[00:08:05] Lenny Rachitsky

English:

There's so many interesting threads there. One is just, there's always this question of just where value will accrue in AI. And it's so interesting, you guys started almost at the bottom layer of infrastructure GPUs and then you went to what people call a GPT wrapper, not actually. So I guess any just lessons there, just thoughts on just where you think value will end up in this world of AI and the stack of AI tools?

中文翻译:

这里有很多有趣的线索。一个是关于 AI 价值将沉淀在哪里的永恒话题。很有意思的是，你们几乎是从最底层的基础设施 GPU 开始的，然后转向了人们所谓的“GPT 套壳”（虽然实际上不是）。所以我想问，关于 AI 世界和 AI 工具栈中价值最终会落在哪里，你有什么教训或想法吗？

[00:08:28] Varun Mohan

English:

Maybe I can start by just saying one thing about startups that I think are really true, it's very unlikely the first thing that you believe you should go work on is going to be the right thing, which is a very hard thing to kind of wrangle with being a startup founder, right? You need to kind of be irrationally optimistic that what you're going to do is going to be differentially important. Because otherwise, why would you go out and do what you're doing? And if it's obvious, then a bigger company would've already done it, right?

中文翻译:

也许我可以先说一件我认为关于初创公司非常真实的事：你最初认为应该去做的事情，极大概率不是那件正确的事。作为创始人，处理这种心态非常困难。你必须保持某种“盲目的乐观”，相信你正在做的事情具有独特的、差异化的重要性。否则，你为什么要去做呢？如果这件事显而易见，大公司早就做了，对吧？

[00:08:56] Varun Mohan

English:

But then you also need to be really, really realistic because most ideas that are, I guess, non-conventional are usually bad ideas, right? So it's this weird tightrope you need to kind of balance on top of where you're pushing for a future that you believe is true, but all the while you're getting new information. You need to kill the beliefs that you had. And if I were to start with the infrastructure piece, we first went in with the assumption that model architectures were going to be really, really heterogeneous.

中文翻译:

但同时你又必须极其现实，因为大多数非传统的想法通常都是坏主意。所以这就像是在走钢丝，你一方面在推动一个你坚信的未来，另一方面又在不断接收新信息。你需要亲手杀掉自己曾经的信念。以基础设施为例，我们最初的假设是模型架构将会是非常非常异构（Heterogeneous）的。

[00:09:27] Varun Mohan

English:

Working from an autonomous vehicle background, there were many different types of model architectures out there. There were convolutional neural networks, graph neural networks, recurrent

neural networks, LSTMs, sort of lighter neural nets with frustrum point networks. And there were maybe tens of architectures we were dealing with. And at that point we were like, the complexity of this is so high that it's very clear if someone offloaded the complexity, there would be a lot of value.

中文翻译:

基于自动驾驶的背景，当时有很多不同类型的模型架构：卷积神经网络（CNN）、图神经网络（GNN）、循环神经网络（RNN）、LSTM，还有带视锥点网络（Frustrum Point Networks）的轻量级神经网络。我们当时处理的架构可能有几十种。那时我们觉得，这种复杂性太高了，如果有人能把这种复杂性卸载掉，那肯定会有巨大的价值。

[00:09:50] Varun Mohan

English:

Fast-forward to the middle of 2023, everything looks like it's going to be a transformer. So now our hypotheses are just wrong. So at this point then, most of the value is probably not going to accrue at purely the, at least this is our belief, at the infrastructure layer. It's going to accrue somewhere else. Where is the layer that you can actually differentiate on? And we believe the application layer is a very, very deep layer to go out and differentiate on, right? What are the number of ways we can build better user experiences and better workflows for developers? We think there's effectively no ceiling on that, on how much better we can make the lives of developers basically.

中文翻译:

快进到 2023 年中期，看起来所有东西都要变成 Transformer 架构了。所以我们之前的假设就错了。到这一步，至少我们认为，大部分价值可能不会纯粹沉淀在基础设施层，而是会沉淀在别处。那么哪一层是可以真正做出差异化的呢？我们认为应用层是一个可以深入挖掘并做出差异化的层级。我们可以通过多少种方式为开发者构建更好的用户体验和工作流？我们认为，在改善开发者生活质量这件事上，基本上是没有天花板的。

[00:10:22] Lenny Rachitsky

English:

You touched on the second thread that I thought was really interesting here, is just how you guys pivoted from ideas that were working. You were making money, people loved it. You said you had millions of dollars of ARR revenue. And then you're just like, "No, we're going to completely change the business." So the question there is, just like, what have you learned about knowing what to follow? And one thing I heard there that was really interesting is just once your assumptions change about that you built your idea on, it's time to think this idea and maybe try something else.

中文翻译:

你提到了第二个我觉得非常有趣的线索：你们是如何从一个已经奏效的想法中转型的。你们当时在赚钱，用户也喜欢，你提到有数百万美元的 ARR（年度经常性收入）。然后你们却说：“不，我们要彻底改变业务。”所以我想问，关于“知道该追随什么”，你学到了什么？我听到一个很有趣的点是：一旦你构建想法的基础假设发生了变化，就是时候重新审视这个想法，甚至尝试别的东西了。

[00:10:48] Varun Mohan

English:

I think the way we sort of think about this is even when we're working right now, we just accept that we're going to get a lot of things wrong. We're just going to get a lot of things wrong. Obviously that's a very big moment because that was a bet the company moment in the sense that we basically said, told our investors, "Hey, we're making money on this." We had already raised 28 million of capital and we were just like, "Hey, we're just going to pivot entirely from this." And we did that overnight. This wasn't this thing where we just said, "Hey, maybe a quarter, one or two quarters." Because one of the things we knew that's very important for startups is focus.

中文翻译:

我认为我们思考这个问题的方式是，即使是现在，我们也接受自己会犯很多错误。那显然是一个重大时刻，因为那是一个“孤注一掷”的时刻。我们基本上是告诉投资者：“嘿，我们现在这个业务在赚钱。”我们当时已经筹集了 2800 万美元的资金，但我们说：“我们要彻底转型。”而且我们是连夜完成的。这不是那种“哦，我们花一两个季度慢慢转”的事情。因为我们知道，对初创公司来说，专注（Focus）至关重要。

[00:11:20] Varun Mohan

English:

If you're trying to do another thing that you think is big and you're focused on something that you don't believe is valuable, you're guaranteed going to fail at the thing you think is going to be big. So that's a very obvious thing there. But I think once you go in with the assumption that a lot of your hypotheses are going to be wrong, but you will do the most concentrated work possible to go out and validate these hypotheses and you won't be in love with your ideas.

中文翻译:

如果你试图去做另一件你认为很伟大的事，却又把精力分散在一些你不再认为有价值的事情上，那么你注定会在那件伟大的事上失败。这是显而易见的。但我觉得，一旦你带着“很多假设都会出错”的前提去工作，你就会尽一切可能去做最专注的工作来验证这些假设，而且你不会盲目爱上自己的想法。

[00:11:41] Varun Mohan

English:

I think ideas, it's awesome when you have a great idea, but you should never be too in love with your ideas. And you have an organization that is very truth-seeking. I think a lot of people at the company have had their ideas tested over and over again. Even just building Windsurf. That is not a complete company pivot, but that's a big decision that we made at the company. You kind of need to make some bets. And sometimes you're wrong and sometimes you're right. But if you have an organization that comes out and you feel like morale is not going to be low if you made the wrong decision, that's the best, right? That means you have optionality for the rest of time.

中文翻译:

拥有好主意很棒，但永远不要过度迷恋它。你需要一个非常“求真”（Truth-seeking）的组织。公司里的很多人都经历过想法被反复测试的过程。即使是构建 Windsurf，虽然不是公司层面的彻底转型，但也是一个重大决策。你必须做一些豪赌。有时你会错，有时你会对。但如果你有一个组织，即使做错了决定士气也不会低落，那就是最棒的状态。这意味着你永远拥有选择权。

[00:12:14] Varun Mohan

English:

And Lenny, one thing that I try to tell the company about this is, this year the total amount of engineering output we'll have is much larger than the engineering output we've had since the beginning of the company's creation 'till now. So that almost means every year is a new lease on life for us, right? It's almost a new way for us to test out an entirely new set of hypotheses. And maybe we were wrong about our original hypotheses in the first place. What makes us more smart than everyone else to be right more times than that?

中文翻译:

Lenny，我试着告诉公司的一件事是：今年我们的总工程产出将远远超过公司成立以来到现在的总和。这几乎意味着每一年对我们来说都是一次新生。这几乎是让我们测试一整套全新假设的新机会。也许我们最初的假设从一开始就是错的。我们凭什么觉得自己比别人聪明，能比别人对得更多呢？

[00:12:46] Lenny Rachitsky

English:

That's so empowering. It makes me think about... Uri Levine was on the podcast, co-founder of Waze, and he has this phrase that he wears on his shirt. His book is called this Fall in Love with the Problem, Not the Solution. And that feels like that's exactly what you're describing. Okay, so let's talk about Windsurf. What's the simplest way for people to understand what is Windsurf?

中文翻译:

这太有力量了。这让我想起 Waze 的联合创始人 Uri Levine 曾上过这个播客，他把一句话印在衬衫上，他的书名也叫这个：《爱上问题，而非解决方案》（Fall in Love with the Problem, Not the Solution）。感觉这正是你所描述的。好，那我们聊聊 Windsurf。让人们理解“什么是 Windsurf”最简单的方式是什么？

[00:13:04] Varun Mohan

English:

Yeah. So Windsurf is an IDE, right? It's an application to go out and build software and build applications. The crazy thing is a lot of people who use the product don't even probably know what an IDE is, which is crazy. And we'll get into that in a second. But why did we go out and build Windsurf and what is Windsurf maybe? Why couldn't we have just done this on top of conventional IDEs like Visual Studio Code?

中文翻译:

是的。Windsurf 是一个 IDE（集成开发环境），对吧？它是一个用来构建软件和应用程序的工具。疯狂的是，很多使用这个产品的人甚至可能不知道 IDE 是什么，这太不可思议了。我们稍后会详细讨论。但我们为什么要构建 Windsurf？Windsurf 到底是什么？为什么我们不能直接在 VSCode 这种传统 IDE 之上构建呢？

[00:13:22] Varun Mohan

English:

So maybe just to get into this a little bit, as we saw that AI was getting more and more powerful, the way people go out and build technology, we thought the interface for that was going to change remarkably. It was not going to be a conventional pure text editor where the user is writing a handful of lines of code or most of the code and the IDE provides maybe some basic feedback on what the user is doing right or

wrong. And the basic feedback could be, "Hey, there's a bug in your software or compiler error in your software." It could do much more, right? It could actually go out and modify large chunks of code.

中文翻译:

稍微深入一点说，当我们看到 AI 变得越来越强大时，我们认为人们构建技术的方式和界面将会发生显著变化。它不再是一个传统的纯文本编辑器——用户写几行或大部分代码，IDE 只提供一些关于对错的基本反馈（比如“嘿，这里有个 Bug”或“编译器报错”）。它可以做得更多，对吧？它实际上可以去修改大块的代码。

[00:14:04] Varun Mohan

English:

One of the key pieces that we recognized was, with this new paradigm with AI, AI was probably going to write well over 90% of the software, in which case the role of a developer and what they're doing in the IDE is maybe reviewing code. Maybe it's actually a little bit different than what it is in the past. And we'll see this very soon with Windsurf. Maybe when you're using the product, actually a good chunk of the user's time is that you're reviewing what the AI is outputting. So we needed to build custom-review flows into the IDE to actually make it so that it was easier to actually go out and do that, right? Because the developer is not spending all their time writing code.

中文翻译:

我们意识到的一个关键点是，在 AI 的新范式下，AI 可能会编写超过 90% 的软件。在这种情况下，开发者的角色以及他们在 IDE 中做的事情，可能变成了审查代码。这与过去确实有些不同。在 Windsurf 中很快就能看到这一点：用户的大部分时间实际上是在审查 AI 的输出。所以我们需要在 IDE 中构建自定义的审查流程，让这件事变得更容易，对吧？因为开发者不再把所有时间都花在写代码上了。

[00:14:38] Varun Mohan

English:

And this is the fundamental premise on why we built the product. We thought we were going to get limited a ton if we had very, very basic UI out there. And I'll give you even a simple example here. We have this auto-complete product that completes a handful of lines of code. Now we've actually launched this offering called Windsurf Tab that basically shows you refactors as well. And these refactors are almost inline refactors. And we were able to build a custom UI for that in Windsurf.

中文翻译:

这就是我们构建这个产品的基本前提。我们认为如果只用非常基础的 UI，我们会受到极大的限制。举个简单的例子：我们有一个自动补全产品，可以补全几行代码。现在我们推出了一个叫 Windsurf Tab 的功能，它基本上还能向你展示代码重构（Refactors）。这些重构几乎是内联的。我们在 Windsurf 中为它构建了自定义 UI。

[00:15:03] Varun Mohan

English:

But in VSCode, because of the access to the APIs, we needed to dynamically generate images right alongside the user's cursor because we just didn't have access to the capabilities to showcase and edit properly. And what we realized is immediately by porting over to Windsurf, our acceptance rate tripled. Same ML models, it just tripled. So what that, I guess, gave us confidence in is yeah, you could argue technology is very important. And I think technology is very important. But if our users are getting very

little value from the technology we're sort of building, you need to really clarify, "Maybe we do need to build a new surface and interface." And that's what Windsurf is.

中文翻译:

但在 VSCode 中，由于 API 访问权限的限制，我们必须在用户光标旁边动态生成图像，因为我们没有权限以正确的方式展示和编辑。我们发现，一旦迁移到 Windsurf，我们的采纳率（Acceptance rate）翻了三倍。同样的机器学习模型，效果却翻了三倍。这给了我们信心：你可以争论技术很重要，我也认为技术很重要，但如果用户从我们构建的技术中获得的价值很小，你就需要明确：“也许我们确实需要构建一个新的载体和界面。”这就是 Windsurf。

[00:15:39] Lenny Rachitsky

English:

So the big bet you took there just to make this super clear, is you were initially working within existing IDEs that everyone was familiar with. And then it was like, "This isn't going to get us where we need to go. We're going to try to convince people to switch to something completely new because it's going to be so much better. It's our own IDE." I think maybe people may not recognize just how risky that is, convincing engineers to use something completely new. That's a huge deal.

中文翻译:

所以你做的那个豪赌，简单来说就是：你们最初是在大家熟悉的现有 IDE 中工作的，然后你们觉得“这没法带我们去想去的地方，我们要说服大家切换到一个全新的东西，因为它会好得多，这是我们自己的 IDE”。我觉得大家可能没意识到这有多冒险——说服工程师使用全新的工具，这可是件天大的事。

[00:16:02] Varun Mohan

English:

Yeah, no, of course. And one of the key pieces, maybe Lenny, that would be important to share is a lot of our developers do use Visual Studio Code. But there are lots of people that write in languages like Java, sort of C++ and so on and so forth, and they might use the JetBrains family of IDEs that like IntelliJ. And for us, we are actually still committed to building on those platforms, right? We just felt though that one of the dominant IDEs, which was Visual Studio Code, was limiting the sort of user interface that we could give to our actual customers.

中文翻译:

是的，当然。Lenny，有一点很重要：我们的很多开发者确实在使用 VSCode。但也有很多人使用 Java、C++ 等语言，他们可能会使用 JetBrains 系列的 IDE，比如 IntelliJ。对我们来说，我们仍然致力于在这些平台上构建产品。只是我们觉得，作为主流 IDE 之一的 VSCode，限制了我们能提供给客户的用户界面。

[00:16:35] Lenny Rachitsky

English:

What is the current state of traction for Windsurf? You hear all these crazy numbers about all the competitors in your space. What can you share there for folks just to know?

中文翻译:

Windsurf 目前的发展势头如何？在这个领域，竞争对手的数据都很疯狂。你能分享一些数据让大家了解一下吗？

[00:16:43] Varun Mohan

English:

Yeah, so maybe a handful. We launched the product a bit over four months ago. And in that period of time, over a million developers have tried the product. And obviously we have many hundreds of thousands of monthly active users right now.

中文翻译:

好，分享几个数据。我们发布这个产品大概四个多月。在这段时间里，超过 100 万名开发者尝试了该产品。显然，我们现在拥有数十万的月活跃用户（MAU）。

[00:16:54] Lenny Rachitsky

English:

I love how these days like, "oh, a million. Oh, no big deal." It's just the numbers are absurd these days. We're just getting used to just 100 million ARR here, million users in four months there. It's just like, "Oh, of course. How could you not have that?" But that's absurd. It's just like an insane time right now. You touched on something that I wanted to get to later, but I may as well bring it up now, the question of just how engineering will change in the future. You throw out the stat that 90% of code is going to be written by AI in the future. Dario from Anthropic recently said the same thing. You guys have a really interesting glimpse into just how things will look in the future. So I guess the question is just, how do you think coding specifically will look in the next few years, how different will it be from today?

中文翻译:

我喜欢现在这种“哦，一百万啊，没什么大不了”的氛围。现在的数字简直荒谬。我们已经习惯了这里一个亿的 ARR，那里四个月一百万用户。好像在说：“哦，当然了，你怎么可能没有呢？”但这真的很离谱，现在真是一个疯狂的时代。你提到了一个我本来想晚点聊的话题，但现在提也无妨：未来工程领域将如何变化。你抛出了一个数据，说未来 90% 的代码将由 AI 编写。Anthropic 的 Dario 最近也说了同样的话。你们对未来有着非常有趣的洞察。所以问题是，你认为未来几年的编程具体会是什么样子？和今天会有多大不同？

[00:17:39] Varun Mohan

English:

I think when we think about what is an engineer actually doing, it probably falls into three buckets, right? What should I solve for? How should I solve it? And then solving it. I guess everyone who's working in this space is probably increasingly convinced that solving it, which is just the pure, "I know how I'm going to do it" and just going and doing it. AI is going to handle vast majority, if not all of it. In fact, it probably actually, with some of the work that we've done in terms of deeply understanding code bases, how should I solve it is also going to get closer and closer to getting done. If you deeply understand the environment inside an organization, if you deeply understand the code base, how you should solve it, given best practices when the company also gets solved.

中文翻译:

我认为当我们思考工程师到底在做什么时，大概可以分为三个部分：我要解决什么问题？我该如何解决它？以及最后去解决它。我想在这个领域工作的每个人可能都越来越相信，“解决它”（即纯粹的“我知道怎么做并把它做出来”）这一步，AI 将承担绝大部分甚至全部。事实上，基于我们在深度理解代码库方面所做的工作，“我该如何解决它”这一步也正变得越来越自动化。如果你深度理解了组织内部的环境和代码库，那么在给定公司最佳实践的情况下，“如何解决”也会迎刃而解。

[00:18:19] Varun Mohan

English:

So I think what engineering kind of goes to is actually what you wanted engineers to do in the first place, which is, what are the most important business problems that we do need to solve? What are the most important capabilities that we need our application, our product to have? And actually going and prioritizing those and actually going and making the right technical decisions to go out and doing it. And I think that's where engineering is probably heading towards.

中文翻译:

所以我认为工程的本质将回归到你最初希望工程师做的事情：我们需要解决的最重要的业务问题是什么？我们的应用程序、我们的产品需要具备哪些最重要的能力？然后去排列优先级，并做出正确的技术决策去实现它们。我认为这就是工程的发展方向。

[00:18:40] Varun Mohan

English:

Now, does that mean that no one needs a CS degree? I think that's maybe a little bit overplayed a little bit just because maybe here's my argument for that. A lot of developers nowadays that build full stack applications, at least until a handful of years ago, they probably went to college and took an operating system course. And in theory, they're not really playing around with the operating system, like the kernel scheduler very frequently. But do those principles help them in understanding why their applications are slow? Do they help them in understanding why some design decisions are better than the other? Yeah, that makes them a much better engineer than another engineer. And I think that idea and the understanding of what's going on at the bottom will make a good engineer even better. But also at the same time, it empowers a bunch of people that never understood all of those things, how to actually build as well, which is another remarkable sort of thing that fell out through this whole process.

中文翻译:

那这是否意味着没人需要计算机科学（CS）学位了？我觉得这种说法可能有点夸张了。我的理由是：很多现在构建全栈应用的开发者，至少在几年前，他们上大学时可能都修过操作系统课程。理论上，他们并不经常直接操作操作系统，比如内核调度器。但这些原理是否能帮助他们理解为什么应用运行缓慢？是否能帮助他们理解为什么某些设计决策优于其他决策？是的，这让他们成为了比别人更优秀的工程师。我认为对底层原理的理解会让优秀的工程师变得更强。但与此同时，它也赋予了那些从未理解过这些底层原理的人构建应用的能力，这也是整个过程中非常了不起的一点。

[00:19:33] Lenny Rachitsky

English:

I don't know if you have kids, but just say you had kids or you had niece or nephew going into college, let's say, would you suggest they do do computer science or would you suggest you're not going to have a

good time if that's the career you choose right now?

中文翻译:

我不知道你有没有孩子，但假设你有孩子，或者你有侄子侄女要上大学，你会建议他们学计算机科学吗？还是你会建议说“如果你现在选这个职业，日子可能不会太好过”？

[00:19:47] Varun Mohan

English:

Yeah. Maybe I think back a little bit. So I went to MIT. A lot of us at the company went to MIT together on the engineering team. I think when I think about what we learned the most for engineering or computer science, it was not exactly like how do you write code. That is almost a given that you can write code after going to college. It's more like the principles of how you think about a problem and how you break it down and how you solve it in an interesting way.

中文翻译:

是的。也许我可以回顾一下。我上的是 MIT，我们公司工程团队的很多人也是 MIT 的校友。当我回想我们在工程或计算机科学中学到最多的东西时，其实并不是“如何写代码”。上完大学能写代码几乎是理所当然的。更多的是关于你思考问题的原则、如何拆解问题以及如何以有趣的方式解决问题。

[00:20:15] Varun Mohan

English:

So an example of a class that I really enjoyed was our distributed systems class. And there, you're kind of reading through literature and understanding how some design decisions were kind of made. And I think it's more like a problem solving kind of course and a major. It's a major of how you solve problems given some constraints of how computers today function, right? Like, here's the speed at which memory sort of operates. Here's the speed at which... Here's how much computation you can do in one cycle or one second. And based on that, you can make some trade-offs and solve a problem.

中文翻译:

比如我非常喜欢的一门课是分布式系统。在那门课上，你会阅读文献，理解某些设计决策是如何做出的。我认为这更像是一门关于“解决问题”的课程和专业。它教你如何在当今计算机运行的约束条件下解决问题，比如：内存运行的速度是多少？一个周期或一秒钟内能完成多少计算？基于这些，你可以做权衡并解决问题。

[00:20:45] Varun Mohan

English:

So I don't know if I would say that you shouldn't go get a computer science degree. I think computer science is almost synonymous with problem solving. In that case, I think it's pretty valuable. Is everything you learn in your computer science degree useful? I'd say a lot of things that I learned in my computer science degree are not useful. I'll give you an example. I took a parallel computing class in Julia. I don't think Julia is a very popular programming language anymore. Am I very sad that I took the class? No. The principles of parallel computing are still very useful, I would say, today.

中文翻译:

所以我不会说你不应该去拿计算机科学学位。我认为计算机科学几乎等同于“解决问题”。从这个角度看，它非常有价值。你在计算机学位中学到的所有东西都有用吗？我会说很多东西都没用。举个例子，我修过一门用 Julia 语言讲的并行计算课。我不觉得 Julia 现在还是什么流行的编程语言。我会因为修了这门课而难过吗？不。并行计算的原理在今天依然非常有用。

[00:21:12] Lenny Rachitsky

English:

So what I'm hearing is, skills that you still want to build, whether it's computer science or maybe some version of computer science, is kind of building the mental model of how computers and systems work. Parallel processing, memory, hard drives, internet, things like that. And then there's just problem solving skills, being able to solve interesting problems. Is there any other skills you think people should be investing more in with the rise of AI building more and more of our products?

中文翻译:

所以我听到的是，你仍然想要培养的技能（无论是通过计算机科学还是某种变体）是建立关于计算机和系统如何运行的心智模型。比如并行处理、内存、硬盘、互联网等等。然后就是解决问题的技能，能够解决有趣的问题。随着 AI 越来越多地构建我们的产品，你认为人们还应该在哪些技能上投入更多？

[00:21:37] Varun Mohan

English:

I think one of the things that's maybe a little bit undervalued is this kind of agency piece. And I think about this a lot, which is, you have a lot of people that could go through college and go through school and they're basically told exactly what to do on a P-set. They're given these very, very, I would say, well-defined paths that they need to take. I think maybe in society and just school, we don't prioritize how do you make sure you get people with real agency that want to build something, right? Their goal is not just to maybe graduate from college and then get a job at a big tech company where they're told exactly what to do or where to put the pixel for this one website.

中文翻译:

我认为有一点可能被低估了，那就是“自主性”（Agency）。我经常思考这个问题：很多人读完大学、读完书，基本上只是被告知在作业（P-set）中该做什么。他们被给予了非常明确的路径。我觉得在社会和学校里，我们并没有优先考虑如何培养出具有真正自主性、想要去创造东西的人。他们的目标不应该只是大学毕业，然后在一家大科技公司找份工作，听从指令去决定某个网站的像素该放在哪儿。

[00:22:27] Varun Mohan

English:

I think that's maybe a skill set that is undervalued just right now, probably in the last maybe 10 years or so. And I think that's going to be really, really important. For a startup, obviously these are skills that we just look for. We look for people that are really high agency because we just recognize that by default, if we don't innovate and do crazy things, we're going to die. The company is just going to die. So we just look for this, right? But I would say for most software engineering jobs, that's probably not the case. Just think about big company X and what they're hiring for on the average software engineering interview. It probably doesn't look like that.

中文翻译:

我认为这可能是目前被低估的一种技能，大概在过去 10 年左右都是如此。我认为这将变得非常非常重要。对于初创公司来说，这显然是我们寻找的技能。我们寻找具有极高自主性的人，因为我们意识到，如果默认不创新、不做疯狂的事，我们就会死。公司就会倒闭。所以我们寻找这种特质。但对于大多数软件工程职位来说，情况可能并非如此。想想那些大公司 X，他们在普通的软件工程面试中招的是什么样的人，大概不是这种。

[00:22:52] Lenny Rachitsky

English:

I love how you phrased that. If we don't do crazy things and innovate, we're going to die. That would be a great title for this podcast episode. And I think, I know, it's 100% true. There's just a lot of crazy things happening and a lot of innovation happening. And if you can't keep up, you'll die. So let's talk about hiring. You have a really interesting approach to hiring. There's a few questions I have here. One is just how do you... I know you try to stay really lean. That's a common theme across all the AI startups these days. How do you know when it's time to hire someone?

中文翻译:

我喜欢你那句“如果我们不做疯狂的事并创新，我们就会死”。这可以作为这期播客的标题。而且我知道这是 100% 正确的。现在发生了太多疯狂的事，有太多的创新。如果你跟不上，你就会死。那我们聊聊招聘吧。你的招聘方式很有趣。我有几个问题，一个是……我知道你试图保持团队精简，这是现在所有 AI 初创公司的共同主题。你如何判断什么时候该招人了？

[00:23:22] Varun Mohan

English:

I love the idea of being a lean company, but I don't idolize it in the way that, "Hey, it is a dream to be a 10% or 20% company that's making 50, 100, 200 million in revenue." That's not, I think, what we idolize inside the company. I think what we idolize is, be the smallest company we can be to satisfy our ambitions. That's what the goal is. And maybe, Lenny, the way I would sort of put that out there is, if I told you, "Hey, I'm going to build an autonomous vehicle," and I said our team is 10 people, you should rightfully say, "Hey Varun, you're not serious. And you'd be right. I'm not serious at that point. So I think the answer is, what is the minimum number of people to go out and build the crazy ambition project that you have?

中文翻译:

我喜欢精简公司的理念，但我并不盲目崇拜它，比如觉得“哇，只有 10 到 20 个人却能创造 5000 万、1 亿甚至 2 亿收入是终极梦想”。这不是我们公司内部崇尚的东西。我们崇尚的是：在能实现我们野心的前提下，保持尽可能小的规模。这就是目标。Lenny，我换个说法：如果我告诉你“我要造一辆自动驾驶汽车”，然后说我们的团队只有 10 个人，你理所当然会说：“Varun，你不是认真的。”你是对的，那时候我确实不是认真的。所以答案是：为了实现你那个疯狂的野心项目，最少需要多少人？

[00:24:04] Varun Mohan

English:

And I think the project we are trying to go out and do, which is completely transform the way software gets built, we've mentioned this [inaudible 00:24:11] the company, our goal is to reduce the time it takes

to build apps and technology by 99%, right? It is a tremendously sort of ambitious goal. And it's not possible for us to be a 10, 20, 30, 40 person engineering team in the long term and actually satisfy that goal. We think there's a very, very high ceiling. So that's maybe the first key piece there. It's like, if we can crack actually being a fairly sizable company but still operate as if we're a startup, that's the dream. That's the dream.

中文翻译:

我们正在尝试做的项目是彻底改变软件构建的方式。我们提到过，公司的目标是将构建应用和技术所需的时间减少 99%，对吧？这是一个极其宏大的目标。从长远来看，我们不可能只靠一个 10、20、30 或 40 人的工程团队来实现这个目标。我们认为这个领域的天花板非常高。所以这是第一个关键点：如果我们能做到既是一家规模可观的公司，又像初创公司一样高效运作，那就是终极梦想。

[00:24:40] Varun Mohan

English:

In terms of hiring philosophy, the way we sort of think about things is, we only hire for a role if we're actually underwater for that function. So let's say we're going out and building inference technology. Unless we're underwater there, we will not go out and hire someone to go out and work for that. And the reason for that is, I actually think this is a feature. When you hire for a role and you already have enough people there, you get a lot of weird politics that ultimately ends up happening. And it's not because people are bad people. I think most people are really well-intentioned. But what happens when you have people that join a company and in reality you didn't really need them? They will go out and manufacture some other thing that they should go work on. They will go out and figure out something else to work on. And realistically, it's not that important, but they will go out and try to convince the rest of the organization that it is important.

中文翻译:

在招聘哲学方面，我们的思考方式是：只有当某个职能部门已经“快溺水了”（忙不过来了），我们才会招人。假设我们要构建推理技术，除非我们在那方面已经忙得不可开交，否则我们不会招人。原因在于，我其实认为这是一种“特性”。当你为一个已经人手充足的职位招人时，最终会产生很多奇怪的政治斗争。这并不是因为大家是坏人，大多数人都是出于好意。但当有人加入公司而实际上你并不需要他们时，会发生什么？他们会去“制造”一些别的事情来做。他们会找点别的东西来研究。现实中，那些事可能并不重要，但他们会试图说服组织的其他成员说那些事很重要。

[00:25:30] Varun Mohan

English:

I just think as a startup, we don't have the bandwidth to go out and deal with that, right? For me, I would like to see everyone just almost be raising their hands up being like, "I'm dying. We need one more person." And that's when we go out and hire someone. And one of the analogies I like to give is, I want the company to almost be this dehydrated entity and every hire is like a little bit of water. And we only go back and hire someone when we're back to being dehydrated.

中文翻译:

我认为作为一家初创公司，我们没有精力去处理这些。对我来说，我希望看到每个人几乎都在举手说：“我不行了，我们需要多一个人。”那时候我们才会去招人。我喜欢用的一个比喻是：我希望公司几乎是一个“脱水”的实体，每一次招聘就像注入一点点水。只有当我们再次回到脱水状态时，我们才会再去招人。

[00:25:57] Lenny Rachitsky

English:

I love this metaphor so much. And it sounds painful. It sounds painful that you need to be underwater and raising your hand, "I'm about to die and dehydrated." But I also know that it's a really exciting way to work. It sounds hard, but if you're in it, it's just like... I guess talk about just that side of it because I think it could sound like, "This is terrible. I don't want to work this way."

中文翻译:

我太喜欢这个比喻了。听起来挺痛苦的，你需要处于“快溺水了”的状态，举手说“我要死了，我脱水了”。但我知道这其实是一种非常令人兴奋的工作方式。听起来很难，但如果你身处其中，感觉会很不一样……我想请你谈谈那一面，因为听起来可能会让人觉得“这太糟糕了，我不想这样工作”。

[00:26:19] Varun Mohan

English:

You know what I actually think, Lenny? It's really good for a handful of reasons, which is that a lot of the... We respect and trust the people that work at the company. So this forces ruthless prioritization. You have a team that's going out and doing something. They will never ask to work on something that's not important. In fact, if there are two things that they're working on, they're just going to just tell me, "Hey, there are two things on my plate. I just don't have the ability to do two. I can only do one." And they will pick the one that's most important.

中文翻译:

Lenny，你猜怎么着？我觉得这其实非常好，原因有几个。首先，我们尊重并信任在公司工作的每一个人。这迫使我们进行“无情的优先级排序”。当一个团队在做某件事时，他们永远不会要求去做不重要的事情。事实上，如果他们手头有两件事，他们会直接告诉我：“嘿，我盘子里有两件事，但我没能力同时做两件，我只能做一件。”然后他们会挑选最重要的那件。

[00:26:45] Varun Mohan

English:

And this actually goes back to one thing that I think is true about startups and just companies in general. You don't win by doing 10 things well. You win by doing one thing really well and maybe you fail nine things. This is the thing that I've told the company, "This is very different than school," right? In school you optimize for your total GPA. But for companies, I just need to get an A+ on the one class that matters. And then I can get an F in all the other classes. And an F in all the other classes doesn't mean just doing illegal things. That basically means you just deprioritize things that don't matter. That actually forces this organizational prioritization that is just really, really good.

中文翻译:

这其实回到了我对初创公司乃至所有公司的一个看法：你不是靠把 10 件事都做好而获胜的，你是靠把一件事情做得极其出色，哪怕其他 9 件事都失败了。我告诉过公司：“这和学校非常不同。”在学校里，你要优化总的 GPA（平均绩点）。但在公司里，我只需要在最重要的那一门课上拿到 A+，其他课拿 F 都没关系。这里的 F 并不是指做违法乱纪的事，而是指你把不重要的事情降级处理。这实际上迫使组织进行优先级排序，这非常棒。

[00:27:22] Varun Mohan

English:

And Douglas and I, Douglas being my co-founder, we can tell the company, "These are the two things that are the most important." But if we go out and tell these are the two things that are the most important to the company and then we put the company has 20% more people than necessary, what's going to ultimately happen? It's almost a forcing function for ruthless prioritization to have fewer people or people that are just underwater internally at the company.

中文翻译:

我和 Douglas（我的联合创始人）可以告诉公司：“这两件事是最重要的。”但如果我们告诉公司这两件事最重要，却又给公司多配了 20% 的人手，最终会发生什么？保持较少的人手，或者让内部人员处于“快溺水”的状态，几乎是实现“无情优先级排序”的一种强制手段。

[00:27:46] Lenny Rachitsky

English:

Everyone listening that works at a big company knows exactly what you mean when you described when there's just too many people, they will all find work to do and they will all be pitching ideas. They all want to show impact, they want to do well in their performance reviews. That's just the nature of too many people at a company. And so I think this all really resonates. To even getting even deeper on just what it looks like when someone's underwater to tell you it's time to hire, is it just someone coming to you, "Varun, I need someone on this team. This is just not possible"? What does that look like even more practically?

中文翻译:

在大公司工作的听众肯定明白你的意思。当人太多时，大家都会给自己找活干，都会去推销自己的想法。大家都想展示影响力，想在绩效评估中表现出色。这就是人太多的必然结果。所以我觉得这很有共鸣。再深入一点，当有人“快溺水”并告诉你该招人时，具体是什么样子的？是有人跑来跟你说：“Varun，我这个团队需要人，这活儿没法干了”吗？实际操作中是什么样的？

[00:28:16] Varun Mohan

English:

Yeah, I think it's basically along those lines. It's that, "Hey, there's some pressure to get something done in a short period of time." By the way, one of the things that we do believe though for software, if you want to do great things, it's not possible to just say, "Hey, I want to get it done in one month" if it is like... Because you have to think about it from this perspective. If a software project could get built in two to three weeks, what does that really mean about the true complexity and differentiation of what you built? It's probably not very high, unless you believe you are way smarter than everyone else. But I think that's hubris, right? I think we actually have a very exceptional engineering team. But also at the same time, I don't think our engineering team is so exceptional that we can do things in three weeks that the rest of the world can't do in six to nine months. That's kind of stupid to believe that.

中文翻译:

是的，基本上就是那样。比如，“嘿，要在短时间内完成某件事压力很大。”顺便说一下，我们对软件开发有一个信念：如果你想做伟大的事情，你不能只是说“我想在一个月内完成”。因为你得从这个角度想：如果一个软件项目两三周就能做出来，那它真正的复杂性和差异化能有多高？除非你觉得自己比全世界都聪明。但我觉

得那是狂妄自大。我认为我们的工程团队非常出色，但我不认为他们出色到能在三周内完成全世界其他人在六到九个月内都做不出来的事情。那样想太蠢了。

[00:28:57] Varun Mohan

English:

So I think basically it comes down to that person coming out and being like, "Hey, look, I don't have enough time to do X." Us having a conversation to be like, "Okay, what can you do then?" And if the answer is, "I can only do less than that," then maybe we make a decision actually, "Oh wow, that's great. Maybe we actually should deprioritize Y." Because this is actually also another thing that's very hard even for people like me and my co-founder. It's that we also want to do a lot of things. There's an urge to do a lot of things. But if we are forced to make a decision constantly on like, "We cannot do X," it's very clarifying. It's very clarifying because our engineering interview process is also extremely low acceptance rate. So it's not very easy for us to very quickly spin up people and have them join the company really, really quickly either.

中文翻译:

所以基本上就是那个人跑来说：“嘿，我没时间做 X 了。”然后我们对话：“好，那你现在能做什么？”如果答案是“我只能做比那更少的事”，那我们可能会决定：“噢，太好了，那我们也许应该把 Y 的优先级降低。”这对我和联合创始人来说也很难，因为我们也想做很多事，有一种想做很多事的冲动。但如果我们被迫不断做出“我们不能做 X”的决定，这会让我们变得非常清醒。这非常清醒，因为我们的工程面试通过率极低，所以我们也不可能迅速招到人并让他们立刻入职。

[00:29:43] Varun Mohan

English:

So I think it's clarifying for everyone. It's clarifying for the person that wants more people. We can just tell them, "Hey look, we don't believe you should be doing this other thing." And it's also clarifying for us because we can also get on the same page with them. And sometimes we just kind of agree, "Hey..." Our teams are very flexible that, "Hey, actually we do need to get something done." And one of the things that we've kind of tried to make sure is true on our engineering team is, people's value to the company does not have anything to do with the size of their team. There are projects inside the company, there are directly responsible individuals for these projects inside the company. And if we feel like one project is very important, then people can move from one project to the next.

中文翻译:

所以我觉得这对每个人都是一种澄清。对想要更多人手的人来说，我们可以告诉他：“嘿，我们觉得你不应该去做那件别的事。”对我们来说也是一种澄清，因为我们可以和他们达成共识。有时我们会达成一致：“嘿，实际上我们确实需要完成某件事。”我们在工程团队中努力确保的一点是：一个人的价值与他管理的团队规模无关。公司内部有各种项目，每个项目都有直接负责人（DRI）。如果我们觉得某个项目非常重要，人们可以从一个项目转移到另一个项目。

[00:30:21] Varun Mohan

English:

There's no notion of someone owning people at the company. That is a very bad and gnarly idea. In fact, the person that is the most valuable at the company is the person that can do the most crazy sort of

project out there with as few people as possible. And that's what you should be rewarding internally.

中文翻译:

公司里没有“某人拥有某些下属”的概念。那是一个非常糟糕且棘手的想法。事实上，公司里最有价值的人是那个能用最少的人手完成最疯狂项目的人。这才是内部应该奖励的行为。

[00:30:35] Lenny Rachitsky

English:

How many people do you have at coding at this point?

中文翻译:

目前 Codeium 有多少人?

[00:30:37] Varun Mohan

English:

So we have close to 160 people and the engineering team is over 50 people right now.

中文翻译:

我们现在有接近 160 人，工程团队目前超过 50 人。

[00:30:42] Lenny Rachitsky

English:

Awesome. Oh, what's the other bigger functions? So [inaudible 00:30:46]-

中文翻译:

太棒了。那其他比较大的职能部门是?

[00:30:46] Varun Mohan

English:

We have go-to-market. We have a... Yeah.

中文翻译:

我们有市场开拓（Go-to-market）团队。我们有……是的。

[00:30:48] Lenny Rachitsky

English:

Oh, right. Okay. I want to talk about that, the sales learning that you guys had. Okay. But let's close out this hiring conversation. So we talked about what you look for... To tell you it's the time to hire, what do you look for in the people that you interview and hire?

中文翻译:

噢，对。好，我想聊聊你们在销售方面的经验。不过我们先结束关于招聘的话题。我们谈到了什么时候该招人，那么在面试和招聘时，你最看重候选人的什么特质？

[00:31:01] Varun Mohan

English:

One of the key pieces that we look for, we have a very high technical bar. Assuming that they actually meet the technical bar, I think we sort of look for people that are really, really passionate about the mission of what we're actually trying to solve and people that are willing to work very hard. I think one of the things that we don't try to do is convince people, "Hey look, we are a very chill company and it's great to work here." I think, no, this is a very exciting space. It's very competitive. You should expect us to lose if the people at the company are not kind of... They're not working very hard. And I think one of the biggest dog whistles I hear is, when I ask people how hard are you willing to work, some people actually ultimately say, "Hey, I work very smart." And I basically ask them a question, "If we have many smart people at our company that also work hard, what's the differentiator going to be? Are you just going to pull them down?"

中文翻译:

我们看重的一个关键点是技术门槛，我们的技术门槛非常高。假设他们达到了技术门槛，我们会寻找那些对我们正在解决的使命充满激情，并且愿意非常努力工作的人。我们不会试图去说服别人说：“嘿，我们是一家非常轻松的公司，在这里工作很棒。”不，这是一个非常令人兴奋的领域，竞争极其激烈。如果公司的人不努力工作，你就应该预料到我们会输。我听到的最大的“暗示”之一是，当我问人们愿意多努力工作时，有些人最终会说：“嘿，我工作很聪明（Work smart）。”我基本上会反问他们：“如果我们公司有很多既聪明又努力的人，你的差异化在哪？你是不是只会拖累他们？”

[00:31:48] Varun Mohan

English:

Because I think one of the things that's true about companies is it's like this massive group project. And I think the thing about a person that is not pulling their weight that's bad. It's not the productivity, right? At some point when the company becomes many hundreds of engineers, I'm not going to be thinking about the one engineer that's not pulling their weight. It's the team of people they work with that are almost basically saying, "Is this the bar internally at the company? Is this the expectation?" And I guess, Lenny, if I told you you have a team of five people and the four other people you're working with just don't care, how much are you going to feel like you should care?

中文翻译:

因为我认为公司就像一个巨大的小组作业。一个不尽责的人最糟糕的地方不在于他的产出。当公司有几百名工程师时，我不会去想那个不尽责的个体。糟糕的是和他一起工作的团队，他们会想：“这就是公司的内部标准吗？这就是期望吗？”Lenny，如果我告诉你，你有一个五人团队，而其他四个人根本不在乎，你还会觉得自己应该在乎吗？

[00:32:21] Lenny Rachitsky

English:

Not too much.

中文翻译:

不会太在乎。

[00:32:22] Varun Mohan

English:

Exactly. So for us, I think that's what we more care about. We have a culture where it's very collaborative. It's not an individual sport, but people feel like they can rely on other people to get complex sort of tasks done.

中文翻译:

没错。所以对我们来说，这是我们更看重的。我们有一种非常协作的文化。这不是一项个人运动，人们觉得可以依靠其他人来完成复杂的任务。

[00:32:35] Lenny Rachitsky

English:

So the question you asked there just basically is, how hard are you willing to work? How hard do you want to work? And I know some people, there's this whole group of folks that are just like work-life balance, "How dare you ask me to work crazy hours?" And I love just the filter upfront of, "If you work here, you will work really hard. You'll work a lot of hours. It's a crazy space to be in. And we will win by working smart and also really hard."

中文翻译:

所以你问的问题基本上就是：你愿意多努力工作？你想多努力工作？我知道有一群人非常看重工作与生活的平衡，会觉得“你怎么敢要求我疯狂加班？”我喜欢这种前置的筛选：“如果你在这里工作，你会非常努力，会工作很长时间。这是一个疯狂的领域。我们将通过既聪明又极其努力的工作来获胜。”

[00:33:02] Varun Mohan

English:

Yeah.

中文翻译:

是的。

[00:33:03] Lenny Rachitsky

English:

You said at some point earlier that your engineering pass rate, as you said, it was like 0.6% of candidates, something like that.

中文翻译:

你之前提到过，你们工程职位的通过率大概是 0.6% 左右，是吗？

[00:33:10] Varun Mohan

English:

Yeah, it's probably post or take home. It's probably that actually. So the take home itself filters probably another 10, 15X on top of that.

中文翻译:

是的，大概是笔试或家庭作业之后的通过率。家庭作业环节本身可能又会刷掉 10 到 15 倍的人。

[00:33:19] Lenny Rachitsky

English:

Here's a question that I've been hearing more and more, is just, how do you do interviews these days with tools like Windsurf out there that solve all your problems?

中文翻译:

这是一个我听得越来越多的问题：既然现在有了像 Windsurf 这样能解决所有问题的工具，你们现在是怎么做面试的?

[00:33:25] Varun Mohan

English:

We are okay with people using the tools because I think one of the worst things is like, if someone comes here and doesn't like using these tools, we believe there are massive productivity improvements. We do bring people into the company on site so we can actually see how they think through problems on a whiteboard and all these other pieces. So we do want to see how they think on their feet and hopefully they're not just taking what we're saying, putting it in a voice translator and sticking it into ChatGPT and getting the answer out.

中文翻译:

我们允许人们使用这些工具，因为我觉得最糟糕的事情之一是，如果有人来这里却不喜欢用这些工具，那就不行，因为我们相信这些工具能带来巨大的生产力提升。我们会邀请候选人来公司现场，这样我们可以看到他们在白板上如何思考问题。我们确实想看他们的临场思考能力，希望他们不只是把我们说的话放进语音翻译器，然后塞进 ChatGPT 拿答案。

[00:33:51] Varun Mohan

English:

So there is a way to do this. My viewpoint on this is the tools are really, really important, but I do think we still look for some problem solving ability. If the only way you can solve a hard problem is put it into ChatGPT, I think that's a concern to us.

中文翻译:

所以是有办法应对的。我的观点是，工具非常非常重要，但我认为我们仍然在寻找某种解决问题的能力。如果你解决难题的唯一方法就是把它扔进 ChatGPT，那我们会很担心。

[00:34:07] Lenny Rachitsky (Ad: Coda)

English:

Today's episode is brought to you by Coda. I personally use Coda every single day to manage my podcast and also to manage my community. It's where I put the questions that I plan to ask every guest that's coming on the podcast. It's where I put my community resources, it's how I manage my workflows. Here's how Coda can help you. Imagine starting a project at work and your vision is clear, you know exactly who's doing what and where to find the data that you need to do your part. In fact, you don't have to waste time searching for anything because everything your team needs from project trackers and OKRs to documents and spreadsheets lives in one tab all in Coda. With Coda's collaborative all in one workspace, you get the flexibility of docs, the structure of spreadsheets, the power of applications, and the intelligence of AI all in one easy to organize tab.

中文翻译:

本期节目由 Coda 赞助。我个人每天都在使用 Coda 来管理我的播客和社区。我会把计划问每位嘉宾的问题放在那里，还有我的社区资源和工作流管理。Coda 是这样帮助你的：想象你在公司开始一个项目，愿景清晰，你确切知道谁在做什么，以及在哪里可以找到你需要的数据。事实上，你不需要浪费时间去搜索任何东西，因为团队需要的一切——从项目追踪器、OKR 到文档和表格——都存在于 Coda 的一个标签页中。通过 Coda 的协作式全能工作空间，你可以在一个易于组织的标签页中获得文档的灵活性、表格的结构、应用程序的能力以及 AI 的智能。

[00:34:54] Lenny Rachitsky (Ad: Coda)

English:

Like I mentioned earlier, I use Coda every single day. And more than 50,000 teams trust Coda to keep them more aligned and focused. If you're a startup team looking to increase alignment and agility, Coda can help you move from planning to execution in record time. To try it for yourself, go to coda.io/lenny today and get six months free of the team plan for startups. That's C-O-D-A, .io/lenny to get started for free and get six months of the team plan. coda.io/lenny.

中文翻译:

正如我之前提到的，我每天都用 Coda。超过 5 万个团队信任 Coda，用它来保持团队的一致性和专注度。如果你是一个寻求提高一致性和敏捷性的初创团队，Coda 可以帮助你以创纪录的速度从规划转向执行。亲自尝试一下，今天访问 coda.io/lenny，即可获得六个月免费的初创团队计划。即 C-O-D-A, .io/lenny，免费开始并获得六个月的团队计划。coda.io/lenny。

[00:35:24] Lenny Rachitsky

English:

Okay. Let's talk about this go-to-market sales experience that you guys had. So you started obviously like most people, started building without sales team. And then you realized, from what I hear, that that was a huge miss and a big opportunity to talk about there because that's really unique, I think, that you guys have a large sales team and go-to-market team.

中文翻译:

好。我们聊聊你们的市场开拓（GTM）和销售经验。显然，你们像大多数人一样，刚开始构建时没有销售团队。然后据我所知，你们意识到这是一个巨大的缺失和机会。这真的很独特，因为你们现在拥有一支庞大的销售和市场开拓团队。

[00:35:40] Varun Mohan

English:

Yeah, we actually made this decision pretty early in the company's history, I would say. We hired our VP of sales over a year ago actually. And the go-to-market team is now over 80 people inside the company. So it's a pretty sizable function inside the company. Yeah. Maybe a little bit of a backstory here. So when we started the company, actually we had a handful of angels that actually were operators, go-to-market operators. So an example of one was Carlos Delatorre who used to be the CRO of MongoDB. And I think for us, we never viewed enterprise sales and sales as a very negative thing. I think this is a interesting thing that technical founders sometimes don't really like. They think sales is a very negative part of the process. Everything should be product-led growth.

中文翻译:

是的，我们在公司历史上很早就做出了这个决定。实际上，我们在一年多前就聘请了销售副总裁。现在公司的市场开拓团队已经超过 80 人。所以这是一个相当庞大的职能部门。这里有个背景故事：当我们创办公司时，我们有几位天使投资人实际上是市场开拓方面的运营专家。比如 Carlos Delatorre，他曾是 MongoDB 的首席营收官（CRO）。对我们来说，我们从未将企业级销售视为负面的东西。我觉得这很有趣，因为技术出身的创始人有时不太喜欢销售，觉得销售是流程中负面的部分，认为一切都应该是产品驱动增长（PLG）。

[00:36:37] Varun Mohan

English:

I think it's not that black and white. I think enterprise sales is really valuable. But maybe when we were a GPU virtualization company and we were an infrastructure company, the reason why we never hired a salesperson is, I didn't know how to scale the function. I was the one who was selling the product. So ultimately speaking, if it was hard for me to sell the product incrementally, I didn't know how we could make that into a process that we could then go and scale. I didn't know how we could take the revenue of the business from a couple million to hundreds of millions and let alone even tenths. So if I didn't know how to do that, how could I go out and hire someone and make them scale it out?

中文翻译:

我认为这并非非黑即白。企业级销售非常有价值。但当我们还是 GPU 虚拟化公司和基础设施公司时，我们之所以没招销售，是因为我不知道如何规模化这个职能。当时是我自己在卖产品。说到底，如果连我都很难递增式地卖出产品，我就不知道如何将其转化为一个可以规模化的流程。我不知道如何将业务收入从几百万做到几亿，甚至连做到几千万都不知道。如果我不知道怎么做，我怎么能招个人来让他去规模化呢？

[00:36:58] Varun Mohan

English:

On the other hand, for Codeium, very quickly, a lot of large enterprises reached out to us. And from that alone in the middle of 2023, we started, I guess, me and a handful of other folks at the company started selling the product and we were doing tens of pilots concurrently with large enterprises and we were very quickly able to understand that there was a large enterprise motion that needed to be built in this space. So by the end of 2023, we actually hired our VP of sales. And very quickly after that, scaled our sales team. Yeah, I mean look, if you want to sell to the Fortune 500, it is very hard to do that purely by swiping a credit card.

中文翻译:

另一方面，对于 Codeium，很快就有大量大型企业主动联系我们。仅凭这一点，在 2023 年中期，我和公司里的几个人就开始销售产品，我们同时与大型企业进行着几十个试点项目（Pilots）。我们很快意识到，在这个领域需要建立一套大型企业销售机制。所以到 2023 年底，我们聘请了销售副总裁，并在此后迅速扩大了销售团队。我是说，如果你想把产品卖给财富 500 强公司，纯靠刷信用卡是非常困难的。

[00:37:35] Lenny Rachitsky

English:

Let's talk about Cursor. I don't want to spend too much time with competitors, but that's what everyone's always thinking about when they think of you guys. You guys are kind of the leading players, I think, in the space also. There's Copilot, but that's different. So what's the simplest way to understand how you guys are different from Cursor and also just how you think you win in the space long-term?

中文翻译:

我们聊聊 Cursor。我不想在竞争对手上花太多时间，但大家想到你们时总会想到它。你们都是这个领域的领先者。虽然还有 Copilot，但那是不同的东西。理解你们与 Cursor 区别最简单的方式是什么？以及你认为长期来看你们如何在这个领域获胜？

[00:37:53] Varun Mohan

English:

So I think maybe a handful of things that I could share. So on the product side, I think we've invested a lot in making sure code-based understanding for very large code bases is really high quality. And that's just because of where we started. We worked with some of the worlds are just companies like Dell, JPMorgan Chase. Companies like Dell have singular code bases that are over 100 million lines of code. So being able to understand that really, really quickly to make large scale changes is something that we've spent a lot of time doing. And that requires us actually building our own models that can consume large chunks of their code base in parallel across thousands of GPUs and almost rank them to be able to find out what the most important snippets of code are for any question that are asked about the code base. So we've gone out and built large distributed systems based on our infrastructure background to go ahead and do that. That's maybe one.

中文翻译:

我想我可以分享几点。在产品方面，我们投入了大量精力确保对超大型代码库的理解达到极高质量。这源于我们的起点：我们与戴尔、摩根大通等世界级大公司合作。像戴尔这样的公司，单个代码库就超过 1 亿行代码。因此，能够极快地理解这些代码并进行大规模更改，是我们投入大量时间研究的课题。这需要我们构建自己的模型，能够跨数千个 GPU 并行处理大块代码库，并进行排序，以便针对代码库的任何问题找到最重要的代码片段。基于我们的基础设施背景，我们构建了大型分布式系统来实现这一点。这是第一点。

[00:38:38] Lenny Rachitsky

English:

Let me actually follow that thread because I think people may underestimate just how big of a deal that is. So when we talk about, we had the founders of Bolt and Lovable on the podcast, so those products, they build something from scratch, they built, they write the code for you. So that versus just loading, say,

Windsurf on your million line code base, say, at Airbnb or Uber. Like, understanding what the hell you have and how it works and where to go change things without breaking it is insanely hard. And so what I'm hearing is that's kind of a big differentiator as you guys started there actually. And then Windsurf is now building up that advantage.

中文翻译:

我想追问一下，因为我觉得人们可能低估了这件事的重要性。我们之前请过 Bolt 和 Lovable 的创始人，那些产品是从零开始构建的，它们为你写代码。但这与在 Airbnb 或 Uber 的百万行代码库上加载 Windsurf 是完全不同的。理解你到底有什么、它是如何运作的、以及在不破坏它的前提下该去哪里修改，这是极其困难的。所以我听到的是，这其实是一个巨大的差异化优势，因为你们就是从这里起步的，而 Windsurf 现在正在巩固这一优势。

[00:39:15] Varun Mohan

English:

That's right. Yeah. So that's a big thing that we spent a lot of time on, which is just understanding what the code base is doing. And actually one of the other things is, what are all the user interactions with respect to the code base? And happy to show that also in a bit here.

中文翻译:

没错。这是我们投入大量时间的一件大事，即理解代码库在做什么。另一件事是，用户与代码库之间的所有交互是什么样的？稍后我很乐意演示一下。

[00:39:31] Lenny Rachitsky

English:

Awesome.

中文翻译:

太棒了。

[00:39:31] Varun Mohan

English:

The second key piece probably is we're not only tied to Windsurf actually. This is probably a weird statement given even we are talking about Windsurf, which is that actually we're pretty focused on supporting IDEs like JetBrains. JetBrains or IntelliJ has over 70 to 80% of all Java developers coding in JetBrains based IDEs, right? The reason why we don't feel as big a need to almost build a competing product to JetBrains is JetBrains is actually a very sort of extensible product in a way that VSCode is not. VSCode is not very extensible.

中文翻译:

第二个关键点可能是，我们实际上并不只绑定在 Windsurf 上。考虑到我们正在谈论 Windsurf，这听起来可能有点奇怪，但实际上我们非常专注于支持像 JetBrains 这样的 IDE。JetBrains 或 IntelliJ 拥有超过 70% 到 80% 的 Java 开发者，对吧？我们之所以觉得没那么有必要去构建一个 JetBrains 的竞争产品，是因为 JetBrains 本身是一个非常可扩展的产品，而 VSCode 在这方面做得不够。

[00:40:05] Varun Mohan

English:

So I think for us, our goal here is not only just to satisfy a subset of users that can actually switch onto our IDE, but we want to give this agentic sort of experience to every sort of developer out there. And if that means there are Java developers that write in JetBrains, that's fine. We work with a lot of large enterprises that have 10 plus thousand developers where over 50% of the developers are on JetBrains. It's a very large product. And by the way, that company itself is a privately held company that makes many hundreds of millions of dollars a year. So it's a very, very large company. So for us, that's another key piece. We actually want to meet developers sort of where they are. And if they use a different platform, we'll work on that too.

中文翻译:

所以对我们来说，目标不仅是满足那一小部分可以切换到我们 IDE 的用户，我们还想把这种“智能体”式的体验带给每一位开发者。如果这意味着有 Java 开发者在用 JetBrains，那没问题。我们与许多拥有上万名开发者的企业合作，其中超过 50% 的开发者使用 JetBrains。这是一个非常庞大的产品。顺便说一下，JetBrains 本身是一家年收入数亿美元的私有公司，规模非常大。所以对我们来说，另一个关键点是：我们要去开发者所在的地方。如果他们使用不同的平台，我们也会在那个平台上提供支持。

[00:40:42] Varun Mohan

English:

The third key piece, and this probably sounds another key piece for enterprises, is we work in a lot of very secure environments. We have FedRAMP compliance, which means we can sell to very large government entities. We have a hybrid mode of actually using the product, which means that all the code that lives that is indexed, it actually lives on the tenant of the user, right? Code is one of the most important pieces of IP for the company. So I think just if you were to look at it from a big company perspective, there are many reasons why over the years of just building an enterprise product, we've handled a lot of complexities that large companies want to see. But that's part of it is because of the history of how we got here in the first place.

中文翻译:

第三个关键点（这对企业来说可能非常重要）是，我们在许多非常安全的环境中运行。我们拥有 FedRAMP 合规认证，这意味着我们可以向大型政府机构销售。我们还有一种混合模式，这意味着所有被索引的代码实际上都存储在用户的租户（Tenant）中。代码是公司最重要的知识产权之一。所以从大公司的角度来看，在多年构建企业级产品的过程中，我们处理了许多大公司看重的复杂性。这在很大程度上归功于我们一路走来的历史背景。

[00:41:21] Lenny Rachitsky

English:

Okay, Varun, enough teasing. Let's do a live demo of Windsurf so folks can see what it's like. And then I'm just going to ask you a bunch of questions as we're going through it. So I'll let you pull up a little shared screen where you have Windsurf pulled up.

中文翻译:

好，Varun，别卖关子了。让我们来一场 Windsurf 的现场演示，让大家看看它是什么样的。演示过程中我会问你一堆问题。请分享一下你的屏幕，打开 Windsurf。

[00:41:33] Varun Mohan

English:

Great. So some context, this is a very basic React project. There's nothing in it right now. So if you were to open any sort of file, it's the default React app project. I have this basic image here. You can pass Windsurf images of what you'd like the project to look like, of what I would like an Airbnb for dog's website to kind of look like.

中文翻译:

太好了。先交代一下背景，这是一个非常基础的 React 项目，现在里面什么都没有。如果你打开任何文件，它就是默认的 React 应用模板。我这里有一张基础图片。你可以把图片传给 Windsurf，告诉它你希望项目长什么样，比如我希望这个“狗狗版 Airbnb”网站长什么样。

[00:41:55] Lenny Rachitsky

English:

Beautiful. Beautiful mock-up by the way. I love that this is like all you need.

中文翻译:

漂亮。顺便说一下，这原型图画得真不错。我喜欢这种“只要这张图就够了”的感觉。

[00:41:59] Varun Mohan

English:

This is all you need. This is all you need. So basically what we're going to do is we're going to say, "Hey..." One of the cool parts about Windsurf is it can actually work in an existing project already. So I can basically say, "Hey, change this React app to show an Airbnb for dog's website based on this image and preview it." So now it'll just go out and start executing code, reading through the repository. Obviously, it doesn't know what the current code base actually looks like. And it'll go out and analyze the code base to actually find out the set of changes necessary. So we'll go out and wait and see what it's going to do. But while we're doing that, let's continue the conversation.

中文翻译:

这就够了。基本上我们要做的就是说：“嘿……”Windsurf 的酷炫之处在于它可以在现有项目中运行。所以我可以简单地说：“嘿，根据这张图片修改这个 React 应用，展示一个狗狗版 Airbnb 网站，并预览它。”现在它就会开始执行代码，读取仓库。显然，它还不知道当前代码库长什么样。它会去分析代码库，找出必要的更改。我们先等等看它会做什么。在等待的时候，我们继续聊。

[00:42:45] Lenny Rachitsky

English:

Awesome. Okay, so first of all, so you open up Windsurf. You had a boilerplate React project ready to go. And Windsurf had never really seen this code before. You ask it to do stuff on your code base, which is just like, "Change this to Airbnb for dogs using this design." Amazing.

中文翻译:

太棒了。首先，你打开 Windsurf，准备好一个 React 模板项目。Windsurf 之前从未见过这些代码。你要求它在你的代码库上操作，比如“根据这个设计把它改成狗狗版 Airbnb”。太神奇了。

[00:43:03] Varun Mohan

English:

That's right. That's exactly right.

中文翻译:

没错，正是如此。

[00:43:04] Lenny Rachitsky

English:

Yeah. Okay, cool. So we'll let it run and we'll talk. Let me ask you this question that I've been asking everyone that comes on that is building a product that helps engineers build products and product managers build products and designers. Say you could sit next to every single new user that opens up Windsurf and whisper a couple tips in their ear to help them be successful with the product. What would be a couple tips you'd share?

中文翻译:

好，酷。让它跑着，我们继续聊。我想问你一个我问过所有嘉宾的问题，他们都在构建帮助工程师、产品经理和设计师构建产品的工具。假设你可以坐在每一位打开 Windsurf 的新用户旁边，在他们耳边低声说几个建议，帮助他们成功使用这个产品。你会分享哪几个建议？

[00:43:25] Varun Mohan

English:

Tip number one is just be a little bit patient and both patient and explicit. When you ask the application to go out and make some changes, it could actually go out and make many irrelevant changes. One of the things that I think prevents this the most is just be really, really explicit or as explicit as possible. And one of the things I ask people to do is in the beginning, start by making smaller changes. If there's a very large directory, don't go out and make it refactor the entire directory because then if it's wrong, it's going to basically it destroy 20 files.

中文翻译:

第一个建议是：保持耐心，并且要表达明确。当你要求应用进行更改时，它可能会做出许多无关的更改。我认为防止这种情况最有效的方法就是尽可能地明确、具体。我建议人们刚开始时先做一些小的改动。如果有一个非常大的目录，不要一上来就让它重构整个目录，因为如果它弄错了，基本上会毁掉 20 个文件。

[00:44:00] Varun Mohan

English:

And I think from there, one of the key pieces I think that comes from the users that use the product is they sort of learn what the hills and valleys of the product are. The analogy I like to give are kind of similar to autocomplete. When you use a product like autocomplete, you would think a product that is suggesting

things but only getting accepted 30% of the time would be really, really annoying. But the reason why it's not very annoying is actually because you've actually learned that, hey, 70% of the time, I don't need to accept this. And the times that I do, I know to get value from it. And you also know beforehand if a sort of command that you write is very complex, you just expect, "Hey, the autocomplete is not going to work for it." So I think it's almost like a, understand what the hills and valleys of the product are.

中文翻译:

我认为关键的一点是，用户在使用过程中会逐渐了解产品的“高山和低谷”（优缺点）。我喜欢用自动补全来打比方。当你使用自动补全产品时，你可能会觉得如果它的建议只有 30% 被采纳，那一定会很烦人。但实际上它并不烦人，因为你已经学会了：嘿，70% 的时间我不需要采纳它。而在采纳的时候，我知道能从中获得价值。你也会预先知道，如果你写了一个非常复杂的命令，自动补全可能就不管用了。所以，这就像是去理解产品的边界在哪里。

[00:44:45] Varun Mohan

English:

The crazy thing is, every three months that kind of gets changed and reevaluated. It almost becomes the case that it becomes materially better than it was in the past. So I think maybe patience and being explicit are maybe the two important key pieces I would tell users.

中文翻译:

疯狂的是，每隔三个月，这些边界就会发生变化并被重新评估。它几乎总是变得比过去好得多。所以，耐心和明确，可能是我会告诉用户的两个最重要的建议。

[00:45:00] Lenny Rachitsky

English:

And I think something that was kind of between the lines there is get a gut feeling of what the model is capable of, like how specific to be versus how abstract it can be. And there's kind of this gut feeling you start to build over time.

中文翻译:

我觉得你话里有话的意思是：要对模型的能力建立一种直觉，比如该具体到什么程度，或者可以抽象到什么程度。随着时间的推移，你会建立起这种直觉。

[00:45:12] Varun Mohan

English:

That's right. Yeah. And with that, it feels like we have an actual preview. Guess what? We have a nice-

中文翻译:

没错。说到这，看来我们已经有一个实际的预览了。你瞧，我们有了一个漂亮的——

[00:45:20] Lenny Rachitsky

English:

Cute dogs.

中文翻译:

可爱的狗狗。

[00:45:21] Varun Mohan

English:

A nice dog app. And one of the cool parts is that we've also done beyond just modifying code is actually being able to point to different pieces. And I guess I could just kind of say... I could point to different elements and say, "Hey, make the background..." This is not great design, but I could basically say, if I took this element, "Make this background red and just take a particular element and just change it and make it red." And it should go out and be able to go out and do this.

中文翻译:

一个很棒的狗狗应用。除了修改代码，我们还做了一件很酷的事，就是能够指向不同的部分。我想我可以试着……我可以指向不同的元素并说：“嘿，把背景改成……”虽然这设计不咋地，但我可以选中这个元素说：“把这个背景改成红色”，选中一个特定元素并把它变红。它应该能够完成这个操作。

[00:45:52] Varun Mohan

English:

The preview aspect of the product of being able to showcase the app while it's getting built helps in that, now actually you can live entirely in app world. You don't even maybe even need to look at the code. Granted this looks hideous, but in some ways if I wanted to, I could go out and do that, right?

中文翻译:

产品中的预览功能可以在应用构建过程中实时展示，这很有帮助。现在你实际上可以完全生活在“应用世界”里，甚至可能都不需要看代码。虽然现在这看起来很丑，但如果我想的话，我是可以这么做的，对吧?

[00:46:09] Lenny Rachitsky

English:

This is what happens when there's no more designers. Like, [inaudible 00:46:11]-

中文翻译:

这就是没有设计师后的后果。就像……

[00:46:11] Varun Mohan

English:

Yeah. When there's no more designers. Sure. Maybe the answer is like, when you ask me what should people be doing, they should study great taste. Having great taste. Because I think taste is also a very, very hard, right?

中文翻译:

是的，当没有设计师的时候。当然，也许答案是：当你问我人们应该做什么时，他们应该去研究“好品味”。拥有好品味。因为我认为品味也是一件非常非常难的事情，对吧?

[00:46:22] Varun Mohan

English:

But maybe the other key piece, Lenny, that I wanted to showcase here is obviously you could keep going here. I could take different components and kind of change them. We have a lot of plans here that are beyond just point and click changing components. But one of the cool pieces is the AI. There's an AI review flow as well, which is kind of like what I was saying. The goal of AI has now changed a lot in that it is now modifying large chunks of code for you. And the job of a developer now is to actually review a lot of the code that the AI has generated. And granted right now during this podcast, I'm not going to review all the code that's getting generated.

中文翻译:

但 Lenny，我想展示的另一个关键点是，显然你可以继续操作。我可以选取不同的组件并更改它们。我们在这里有很多计划，不仅仅是点击更改组件。但最酷的部分之一是 AI。还有一个 AI 审查流程，就像我之前说的。AI 的目标现在发生了很大变化，它现在是为你修改大块代码。而开发者的工作现在实际上是审查 AI 生成的大量代码。当然，在播客期间，我不会去审查所有生成的代码。

[00:46:57] Varun Mohan

English:

But let's say I want to go out and modify some of this code. And this is where if you're an actual developer that actually wants to go modify, maybe I don't like my variable name being called title. I want it to be called Title String instead, like this. And if I wanted to go out and make that change and change to go out and say Title String and that's what I'm going to do, I'm just going to tell the AI to continue.

中文翻译:

但假设我想去修改其中一些代码。如果你是一个真正想去修改代码的开发者，比如我不喜欢我的变量名叫做 `title`，我想把它改成 `TitleString`。如果我想做这个改动，把它改成 `TitleString`，我只需要告诉 AI 继续。

[00:47:18] Varun Mohan

English:

The cool part about this is Windsurf not only knows about what the agent has done. It also knows everything that the user has done. Our goal here is to have this almost flow-like state where everything the user has done, the AI also knows. And it is able to predict the intent. And as you can see, it said, "I noticed that the interface property title was changed to Title String." And then it now has gone out and modified all the locations within the app from title to Title String. And now it no longer says that.

中文翻译:

最酷的地方在于，Windsurf 不仅知道智能体（Agent）做了什么，它还知道用户所做的一切。我们的目标是建立一种几乎像“心流”一样的状态，用户做过的每一件事，AI 都知道，并且能够预测意图。如你所见，它说：“我注意到接口属性 `title` 被改成了 `TitleString`。”然后它自动修改了应用中所有从 `title` 到 `TitleString` 的位置。现在它不再报错了。

[00:47:45] Varun Mohan

English:

So this is where even if I'm writing software and I want to go and make point changes, the AI can go out and quickly make these changes on the user's path. Imagine doing a refactor or a migration and you just change one part of the code. You can just tell the AI to continue the rest. And because it deeply understands the code base, it should go out and find all the corresponding places to go out and make the change. And obviously now when I reload my app, there's no bug in the app. It still loads properly. I could obviously tell it to do even cooler things like make the app retro. I don't know what that means, but I guess I could do that. And it should go out and make the change correspondingly for me.

中文翻译:

所以，即使我在写软件并想做一些局部修改，AI 也能在用户的路径上快速完成这些更改。想象一下你在做重构或迁移，你只改了一部分代码，你可以直接告诉 AI 把剩下的做完。因为它深度理解代码库，它会找到所有相应的地方进行修改。显然，现在当我重新加载应用时，没有 Bug，它依然正常加载。我当然还可以让它做更酷的事，比如“让应用变得复古”。我不知道那具体意味着什么，但我猜我可以这么做，它会为我做出相应的改变。

[00:48:23] Varun Mohan

English:

But yeah, that's maybe the high level parts there where the AI is not only able to operate entirely in app space but also on the code space of the users going out and modifying code and to bridge the gap between the two. So it should add leverage not only non-developers that are just purely building apps, but also developers that are just hands-on keyboard too.

中文翻译:

是的，这就是高层面的部分：AI 不仅能完全在应用空间操作，还能在用户修改代码的代码空间操作，并弥合两者之间的鸿沟。所以它不仅能为纯粹构建应用的非开发者提供杠杆，也能为那些亲自动手写代码的开发者提供杠杆。

[00:48:44] Lenny Rachitsky

English:

Amazing. By the way, if you're not on YouTube, you can't see, but you can just select any element of the page and then reference that in your ask of, "Here's what I want changed." I didn't know that was a feature. And that is extremely cool.

中文翻译:

太神奇了。顺便说一下，如果你没在看 YouTube 视频，你可能看不到：你可以直接选择页面上的任何元素，然后在你的请求中引用它，说“这就是我想改的地方”。我以前不知道还有这个功能，这真的非常酷。

[00:48:57] Lenny Rachitsky

English:

So interestingly, so having just looked at Lovable and Bolt and Replit and apps like that, it's basically doing all the things those apps do. Oh, wow. There's the retro version. That's good. I like that it built on your red and made it really nice actually.

中文翻译:

有趣的是，我刚看过 Lovable、Bolt 和 Replit 之类的应用，Windsurf 基本上能做那些应用能做的所有事。噢，哇，复古版出来了。不错，我喜欢它在你那个红色的基础上把它变得挺好看的。

[00:49:11] Varun Mohan

English:

Actually the red looks way better now.

中文翻译:

事实上，现在这个红色看起来好多了。

[00:49:12] Lenny Rachitsky

English:

Yeah, a little green button. This is great. Okay.

中文翻译:

是的，还有一个绿色的小按钮。太棒了。好。

[00:49:14] Varun Mohan

English:

Cool.

中文翻译:

酷。

[00:49:16] Lenny Rachitsky

English:

So I don't think people realize this, but apps like Windsurf, that it could actually do a lot of agentic work for you where you just tell it, "Here. I want you to do this" versus it's auto completing code for you. The big difference is you need to start it with some code base so you have this kind of boilerplate React project. Is there a reason you guys aren't taking that step and just doing that automatically for you? Is it because you're targeting engineers and they don't need that or is there other reasons?

中文翻译:

我觉得大家可能没意识到这一点：像 Windsurf 这样的应用实际上可以为你做很多“智能体”式的工作，你只需要告诉它“我想让你做这个”，而不仅仅是自动补全代码。最大的区别在于你需要先给它一个代码库，比如你刚才那个 React 模板项目。你们为什么不更进一步，把那一步也自动化了呢？是因为你们的目标用户是工程师，他们不需要那个，还是有其他原因？

[00:49:39] Varun Mohan

English:

Lenny, the interesting thing is the base app that you saw for this was also generated by Windsurf. The reason why we sort of didn't generate it is installing all the dependencies takes like three or four minutes. And for the demo, I didn't want to wait. But totally, actually most of the users of the product, probably zero-to-one build these apps.

中文翻译:

Lenny，有趣的是，你刚才看到的那个基础应用其实也是由 Windsurf 生成的。我们之所以没在演示中生成它，是因为安装所有依赖项需要三四分钟，我不想在演示时干等着。但完全可以做到，实际上大多数用户都是用它从零到一构建应用的。

[00:49:57] Varun Mohan

English:

And if I can say one interesting thing is, when we launched Windsurf, actually we tasked everyone at our company to go out and build an app with Windsurf. That included our go-to market team and our sales team. There was a crazy stat that I think people would find surprising, but we've saved over half a million dollars of SaaS products we were going to buy because our go-to-market team has now built apps instead of buying them. Our head of partnerships, instead of buying a partner portal product, has actually built its own partner portal. He had never built software in the past. We've actually come up with ways inside the company to deploy these apps easily in a secure way. And we're actually now building very, very custom software for our company to operate more efficiently, which is, I would not have expected this probably six months ago.

中文翻译:

如果我可以分享一件趣事：当我们发布 Windsurf 时，我们要求公司里的每一个人都用 Windsurf 构建一个应用，包括我们的市场开拓团队和销售团队。有一个疯狂的数据我觉得大家会感到惊讶：我们节省了超过 50 万美元原本打算购买 SaaS 产品的预算，因为我们的市场开拓团队现在自己构建应用，而不是去买。我们的合作伙伴负责人没有购买合作伙伴门户产品，而是自己构建了一个。他以前从未写过软件。我们甚至在公司内部找到了一些方法，可以轻松、安全地部署这些应用。我们现在正在构建非常定制化的软件，让公司运作更高效。这在六个月前我是绝对预料不到的。

[00:50:44] Lenny Rachitsky

English:

That is incredibly interesting. You don't need to name company names, but I guess what's a space you're least bullish on that you think is going to have the most problem here with people building their own version of these sorts of products?

中文翻译:

这太有意思了。你不需要点名，但我猜，你最不看好哪个领域？你认为哪个领域在“人们自己构建这类产品的版本”趋势下会遇到最大的麻烦？

[00:50:56] Varun Mohan

English:

I think maybe my viewpoint are these very, very verticalized niche products I think are going to get... They're going to get competed down a ton. And I think sales products are an example of one of these things. And maybe this is a... I don't want to be very negative, but it's very hard inside a company like ours to task our best engineers to build a best in class sales product. There's not enough interest to do that. Or to build a best in class legal software product or finance software product. It's very, very hard for us to. And actually that's a very big moat for these companies that built these products that they were able to come out, have an opinionated stance on how to do this, hire good enough engineers to go out and build the software. Our company is unwilling to do that. So previously, we would go out and buy the technology because there would be no alternative.

中文翻译:

我认为那些非常垂直、小众的产品将会面临巨大的竞争压力。销售类产品就是一个例子。我不想太悲观，但在像我们这样的公司内部，很难让最优秀的工程师去构建一个顶级的销售产品，大家没那个兴趣。或者去构建顶级的法律软件或财务软件，这很难。实际上，这正是那些构建此类产品的公司的巨大护城河：他们能提出一套有见地的方案，招募足够好的工程师来开发。而我们公司不愿意把精力花在那上面。所以以前，我们会去买这些技术，因为没有替代方案。

[00:51:48] Varun Mohan

English:

But now one of the crazy things is that the domain specialists now have access to build the tools that they ultimately wanted, which is actually crazy. If you think about why were these software companies able to exist these vertical software companies, the reason is because they had many features. The kitchen sink of features worked for a lot of companies, but each individual company only wanted 10% of the features. But the problem is, each individual company was not capable of maintaining a piece of software or building the custom piece of software for 10% of the features, but that has now changed entirely. Now they can.

中文翻译:

但现在疯狂的是，领域专家们现在有能力去构建他们最终想要的工具了。如果你想想为什么这些垂直软件公司能够存在，原因在于他们有很多功能。这种“大杂烩”式的功能对很多公司都适用，但每家公司其实只需要其中 10% 的功能。问题在于，以前每家公司都没有能力为了那 10% 的功能去维护或构建定制软件。但现在情况完全变了，他们可以自己做了。

[00:52:22] Lenny Rachitsky

English:

Yeah. There's always been a story of like, "Why would I spend any time building my own software if I could just..." But now it's like five minutes of time.

中文翻译:

是的。以前总有一种说法：“如果我能直接买，为什么要花时间自己写软件？”但现在，这只需要花五分钟。

[00:52:29] Varun Mohan

English:

Five minutes and maybe even more custom to your system. How many times have you bought a software and you're almost like, "Why is there no integration to X? And I actually use X." How annoying is that? That actually makes the software less useful to you.

中文翻译:

五分钟，而且可能更贴合你的系统。你有多少次买了软件后心想：“为什么它没有 X 的集成？我明明在用 X 啊。”那多烦人啊，这实际上降低了软件对你的价值。

[00:52:43] Lenny Rachitsky

English:

So I think what's cool is when you go back, if someone zooms back to the beginning of when you started the demo, it's basically a PM talking to an engineer, "Hey, build me a Airbnb for dogs. Here's a stupid mock that I made with some boxes." That's almost like a bad PM talking to an engineer and it just actually works. That's what's insane about this. And so that's why this example you're sharing of go-to-market folks, building their own things, it's like they don't need to know anything about product building. It's just describe it in some ridiculous way and draw a couple boxes of what you want it to look like and it makes something for you.

中文翻译:

我觉得很酷的一点是，如果有人回看你演示的开头，那基本上就是一个产品经理（PM）在跟工程师说话：“嘿，给我做一个狗狗版 Airbnb。这是我用几个方框画的简陋原型。”这简直就像一个糟糕的 PM 在跟工程师沟通，但它居然真的做出来了。这就是疯狂之处。所以你分享的那个市场开拓人员自己构建工具的例子，就像是他们不需要懂任何产品构建知识，只需要用某种随意的方式描述一下，画几个框，它就能为你做出来。

[00:53:20] Varun Mohan

English:

Which shows that agency is what matters. If you have a product manager that has an idea, there's no reason for why that idea cannot be more well fleshed out. How many times do you have a product manager that just continualize ideas, but it just feels like they are extremely unsure on how to execute on it? They just want to say things for sake of saying things? But for the people that have ideas and a lot of, I guess, agency, they can go out and prove out what they want without any sort of external resources.

中文翻译:

这说明“自主性”（Agency）才是最重要的。如果一个 PM 有一个想法，没有理由不把它具象化。有多少次 PM 只是在不断抛出想法，但感觉他们对如何执行完全没谱？他们只是为了说而说。但对于那些有想法且有极强自主性的人来说，他们可以在不需要任何外部资源的情况下，去证明自己想要的东西。

[00:53:47] Lenny Rachitsky

English:

I think even more acutely for product folks listening to this, it's the salesperson coming to you being like, "Hey, I want this thing. It's going to help me with my sales team." And you're like, "I don't have a million things to build. I don't have time for this." And so that problem goes away, which I think will make a lot of product leaders really happy. The model that this is sitting on, is it Sonnet?

中文翻译:

我觉得对于听这个播客的产品人来说，更切身的体会是：销售跑来跟你说：“嘿，我想要这个功能，它能帮到我的销售团队。”而你会说：“我手头有一万件事要建，没时间管这个。”现在这个问题消失了，我想这会让很多产品负责人非常开心。这个工具背后的模型是 Sonnet 吗?

[00:54:08] Varun Mohan

English:

Yeah. So just to break down how it ultimately works, we have a model that does planning. And I would say right now Sonnet is a really, really good planning model. I think OpenAI's GPT-4o is also good. But the crazy thing is what we try to do is we try to make the Anthropic based model or Sonnet model try to do as much of the high level planning as possible. And then what we try to do internally is run all the models necessary to do high quality retrieval for the agent. As you could see, the agent needed to understand what the rest of the code base ultimately did. We actually make sure we run models to actually chunk up the entire code base and understand the code base so that obviously it would not be a good idea if we had a 100 million line code base to send that entire code base to Anthropic.

中文翻译:

是的。简单拆解一下它的运作方式：我们有一个负责规划（Planning）的模型。我会说目前 Sonnet 是一个非常出色的规划模型，OpenAI 的 GPT-4o 也不错。但疯狂的是，我们尝试让基于 Anthropic 的模型（即 Sonnet）尽可能多地承担高层规划工作。然后我们在内部运行所有必要的模型，为智能体进行高质量的检索（Retrieval）。如你所见，智能体需要理解代码库的其他部分在做什么。我们确保运行模型来对整个代码库进行分块（Chunking）和理解。显然，如果我们有一个 1 亿行的代码库，把整个代码库都发给 Anthropic 是不可行的。

[00:54:49] Varun Mohan

English:

First of all, you couldn't do that. That's over 1.5 billion tokens of code. So obviously that would be three or four orders of my actually larger than the largest context lens right now. But you also wouldn't want to do that from a cost and latency standpoint too. So that's one. And the second piece that you saw was the model is able to very quickly make edits to the software as well. We have custom models that we built that are post trained on top of popular open source models that can make these edits really, really quickly to the code base. And the reason why you would want to do that is it's A, faster, and B, also that model can actually have more of the code base in context too. So it can be better at applying changes than even Anthropics model too.

中文翻译:

首先，你做不到，那是超过 15 亿个 Token 的代码，比目前最大的上下文窗口还要大三四个数量级。其次，从成本和延迟的角度来看，你也不想这么做。这是其一。其二，如你所见，模型能够非常快速地对软件进行编辑。我们构建了自定义模型，是在流行的开源模型基础上进行后期训练（Post-trained）的，可以极快地对代码库进行修改。这样做的好处是：第一，速度更快；第二，该模型实际上可以在上下文中包含更多代码库内容，因此在应用更改方面甚至可能比 Anthropic 的模型做得更好。

[00:55:28] Varun Mohan

English:

So I think the way we like to think about it is, our only goal is how do we build the best product possible? How do we build the best product possible and how do we make the ceiling as high as possible? And we will go out and build models and train models wherever necessary. But if we're not going to be good at a task and we think the open source is better or Anthropic's better, we'll go and just use the open source or Anthropic.

中文翻译:

所以我们思考的方式是：我们唯一的目标是如何构建最好的产品？如何把天花板做得尽可能高？必要时我们会去构建和训练模型。但如果我们不擅长某项任务，而我们认为开源模型或 Anthropic 的模型更好，我们就会直接使用它们。

[00:55:47] Lenny Rachitsky

English:

And so the models you guys are building, those are built on open source models that people are releasing?

中文翻译:

所以你们构建的那些模型，是基于别人发布的开源模型构建的吗？

[00:55:51] Varun Mohan

English:

Yeah. Interestingly, the one that does retrieval is actually completely pre-trained in-house that actually does that. But yeah, for a lot of different pieces, it's based on open source. Interestingly for the one that does the edits and auto-complete, that is also in-house. As you're typing, we actually do some auto-complete related stuff. I'm happy to show that, but I think a lot of users are familiar with that capability. So I think the way we like to look at it is like, what could we be best at and we will go out and trade. But if we're not going to be best at it, we should not just, for the sake of ego, go out and trade something.

中文翻译:

是的。有趣的是，负责检索的模型实际上是完全由我们内部预训练（Pre-trained）的。但在很多其他部分，它是基于开源模型的。有趣的是，负责编辑和自动补全的模型也是内部开发的。当你打字时，我们会做一些自动补全相关的工作。我很乐意演示，但我猜很多用户已经熟悉这个功能了。所以我们的看法是：我们要看自己在什么方面能做到最好，然后去投入。但如果我们做不到最好，我们不应该为了面子而去强行开发。

[00:56:23] Lenny Rachitsky

English:

This may be getting too technical, but just, is there anything interesting around what you train on?

中文翻译:

这可能有点太技术化了，但关于你们训练模型的数据，有什么有趣的地方吗？

[00:56:27] Varun Mohan

English:

Yeah, so one of the interesting things that we have from our users, and this is where we try to think like, "Why would we be any better?" is that, actually every hour, we get probably tens of millions of pieces of feedback from our users. We get a lot of feedback on what they like and what they don't like. For something like autocomplete, we get a lot of preference data, a lot of preference data. And the preference data is weird. It doesn't look like data that you find on the internet. It's like data as the user is typing. Imagine you're typing some code in a code base, the code's going to be incomplete as you're typing it, right? It's not going to be in a full-fledged form. It's not like it is on GitHub. But we have a lot of data that looks like this.

中文翻译:

是的，我们从用户那里获得了一些有趣的东西。当我们思考“为什么我们会做得更好”时，事实是：每小时我们可能会收到数千万条来自用户的反馈。我们收到了大量关于他们喜欢什么、不喜欢什么的反馈。对于自动补全之类的功能，我们获得了大量的偏好数据（Preference data）。这些偏好数据很奇特，它不像你在互联网上能找到的数据。它是用户打字过程中的数据。想象一下你在代码库里写代码，打字时的代码是不完整的，对吧？它不是最终形态，不像 GitHub 上的代码那样。但我们有很多这种类型的数据。

[00:57:06] Varun Mohan

English:

So we are uniquely well-positioned to actually build a good model that can complete code even when it's in an incomplete state when the models that are out there, the frontier models have consumed very little code that looks like this. So for that case we're like, "Hey, we can go out and do a much better job potentially." And we'll go out and train models on all the preference data we have. The same is kind of true on retrieval, right? There's a way to find out, are we retrieving the right data? Did the user accept the code change after that? Was the retrieval actually a good retrieval a signal that we can get? So basically the way we like to look at it is, if something is just purely code planning, there's not a great reason why we would be the best at that. I can't come up with a coherent argument for that. But for something that looks more along the lines of, "Hey, here's an intermediate code base that is very gnarly and here are some changes that need to get made" and we know the evolution of the code or we've seen the evolution of code across millions of users, we feel like we can do a great job of that.

中文翻译:

因此，我们处于一个独特的优势地位，可以构建一个即使在代码不完整状态下也能进行补全的好模型，而目前市面上的前沿模型（Frontier models）很少接触过这类数据。在这种情况下，我们觉得：“嘿，我们有可能做得更好。”我们会根据所有的偏好数据来训练模型。检索也是如此，对吧？有办法查明我们检索的数据是否正确：用户之后是否接受了代码更改？检索是否提供了一个好的信号？所以基本上，如果只是纯粹的代码规划，我找不出理由说我们会是做得最好的。但如果涉及到“这是一个非常棘手的中间态代码库，需要进行某些更改”，而我们了解代码的演变，或者我们见过数百万用户的代码演变过程，我们觉得我们可以做得非常出色。

[00:58:03] Lenny Rachitsky

English:

I think what's interesting about this is another differentiator/moat for companies that end up winning in this space, is you just have more and more of that data than other companies if you're ahead.

中文翻译:

我觉得有趣的是，对于在这个领域最终获胜的公司来说，另一个差异化优势或护城河在于：如果你处于领先地位，你拥有的这类数据就会比其他公司越来越多。

[00:58:14] Varun Mohan

English:

Yeah. This is sort of why maybe at a high level we like the zero-to-one app building product space. I think it's really... It's a good product space. But ultimately I think it needs to boil down to you understanding the code, because otherwise, you're living at too high a plane where it's not clear why you would be able to be the best at that compared to everyone else. It's not really clear.

中文翻译:

是的。这就是为什么从高层面来看，我们喜欢“从零到一构建应用”的产品空间。这是一个很好的产品空间。但归根结底，我认为它必须落实到对代码的理解上。否则，你所处的层面太高了，很难解释为什么你能比别人做得更好。这并不明确。

[00:58:35] Lenny Rachitsky

English:

As a company, you mean?

中文翻译:

你是说作为一家公司？

[00:58:36] Varun Mohan

English:

As a company.

中文翻译:

作为一家公司。

[00:58:36] Lenny Rachitsky

English:

Versus as a user.

中文翻译:

而不是作为用户。

[00:58:37] Varun Mohan

English:

It feels like it might get competitive in a way that it's not clear where you would continue to differentiate over and over with time.

中文翻译:

感觉竞争会变得非常激烈，以至于不清楚随着时间的推移，你该如何持续保持差异化。

[00:58:45] Lenny Rachitsky

English:

I see. Because if they're just sitting on top of Sonnet and just doing what every other Sonnet wrapper is doing, there's not a lot of differentiation or moat.

中文翻译:

我明白了。因为如果他们只是基于 Sonnet 开发，做着和其他 Sonnet 套壳工具一样的事，那就没有太多的差异化或护城河。

[00:58:54] Varun Mohan

English:

It depends on how you do it. But maybe if I was to say this, if the inputs you're consuming are just web elements, extremely high level web elements, then the interface might be high level enough that it's hard to maybe get better than maybe what the frontier models are doing just across the board. You are just better off just plugging in Sonnet for everything.

中文翻译:

这取决于你怎么做。但如果我这么说：如果你处理的输入只是 Web 元素，即极高层面的 Web 元素，那么界面可能已经足够高层，以至于很难做得比前沿模型本身更好。那样的话，你还不如直接在所有地方都接入 Sonnet。

[00:59:14] Lenny Rachitsky

English:

Got it. Awesome. One thing I wanted to come back to that I wrote down that I think is really important for people to understand, you talked about how with Windsurf it's not necessarily... There's a boilerplate code base that you want to start with because it's actually... Because it's not an abstracted zero-to-one app builder. It's an actual IDE you're coding in. And you talked about how has to install dependencies, which is kind this painful thing. And the reason it has to do that is because running locally on your machine versus in the cloud, like, say, Lovable and Replit and all these guys, although I think Bolt runs in your browser in this really cool way. So that's an important distinction. This is like you're running this locally in your machine and has all the libraries you need to actually run it.

中文翻译:

明白了。太棒了。我想回到我记下的一点，我认为这对大家理解很重要：你提到 Windsurf 并不一定……你想要从一个模板代码库开始，因为它实际上……因为它不是一个抽象的“从零到一应用构建器”，而是一个你正在其中写代码的真实 IDE。你提到它必须安装依赖项，这确实挺烦人的。它之所以必须这么做，是因为它是运行在你的本地机器上，而不是像 Lovable、Replit 那样运行在云端（虽然我觉得 Bolt 以一种很酷的方式运行在浏览器里）。这是一个重要的区别：你是在本地运行它，它拥有实际运行所需的所有库。

[00:59:54] Varun Mohan

English:

No, I think that's important. I think we believe a lot of people sort of build software in what are called code spaces and things in a remote machine. I just think it's that a lot of developers like building locally

for what you said. Like if you're doing things that are more than just full stack applications, you might have dependencies on your machine that are just system dependencies that are just gnarly to install. Let's imagine you're building a GPU-based application and the Nvidia drivers, they're necessary. You just want to give people the flexibility to build where they can build. And I think the IDE and building locally has been a thing that people have done for decades, so probably it's not going to go away in the next couple of years.

中文翻译:

是的，我认为这很重要。虽然很多人在所谓的 Code Spaces 或远程机器上构建软件，但我认为很多开发者还是喜欢本地构建，原因正如你所说。如果你做的不仅仅是全栈应用，你机器上可能会有一些非常难安装的系统依赖项。想象一下你在构建一个基于 GPU 的应用，Nvidia 驱动是必需的。你只想给人们提供灵活性，让他们在能构建的地方构建。而且我认为 IDE 和本地构建已经存在了几十年，在未来几年内大概不会消失。

[01:00:29] Lenny Rachitsky

English:

I love that your sales folks now are running local host servers.

中文翻译:

我喜欢你们的销售人员现在居然也在运行本地主机服务器（Localhost）。

[01:00:34] Varun Mohan

English:

Well, with the browser previews, it's easier, right? You kind of just open it up on the side.

中文翻译:

有了浏览器预览，这就容易多了，对吧？你只需要在旁边打开它就行。

[01:00:37] Lenny Rachitsky

English:

Yeah. Yeah. Oh my god. Okay. I have a few more questions just about how you think and operate at Codeium. So you guys are kind at the forefront of how product teams are going to operate. You're seeing the future every day. And so I'm curious if there's ways you guys have structured your teams, engineers, product design that might be different from how other companies are doing it or have tried stuff that has worked really well or tried stuff that's a huge disaster?

中文翻译:

是的。天呐。好。关于你在 Codeium 的思考和运作方式，我还有几个问题。你们处于产品团队运作方式的最前沿，每天都在预见未来。所以我很好奇，你们在组织工程师、产品和设计团队方面，是否有一些不同于其他公司的方式？或者你们尝试过哪些非常奏效的方法，或者哪些是彻底的灾难？

[01:01:02] Varun Mohan

English:

One interesting decision that we kind of have for core engineering is that we don't have pure product managers for the core engineering side of the company. And by the way, that's purely because we build for developers and our product is built by developers. So I think the intuition from our own developers is hopefully valuable. If not, we might be hiring the wrong type of people. So I think our developers are, in some sense, flexing to be more product conventional product managers.

中文翻译:

我们在核心工程方面做了一个有趣的决定：在公司的核心工程端，我们没有纯粹的产品经理。顺便说一下，这纯粹是因为我们的产品是为开发者构建的，而且是由开发者构建的。所以我认为我们自己开发者的直觉非常有价值。如果不是这样，那说明我们可能招错人了。所以我觉得我们的开发者在某种程度上也在兼任传统产品经理的角色。

[01:01:32] Varun Mohan

English:

Now on the other hand, if we were building something that looked more like Uber or the persona was very different and we didn't ourselves understand it, I think the organization wouldn't look the way it looks. For the enterprise side of the company, because we do work with a lot of large enterprises where the requirements are not something that our engineers would automatically understand, I don't think our engineers wake up and they're like, "We need FedRAMP." This is probably something that a lot of customers come to us with and tell us. We have people that flex in this product strategy role that understand what the customer wants and understands the technical capabilities that we have to best build a product that would help them at scale.

中文翻译:

另一方面，如果我们构建的是像 Uber 那样的东西，或者用户画像非常不同，而我们自己并不理解，那么组织的架构就不会是现在这样。在公司的企业端，因为我们与许多大型企业合作，那里的需求并不是我们的工程师能自动理解的。我不认为我们的工程师一觉醒来会想：“我们需要 FedRAMP 认证。”这通常是客户告诉我们的。我们有一些人兼任“产品策略”角色，他们理解客户的需求，也了解我们的技术能力，从而构建出能大规模帮助客户的最佳产品。

[01:02:12] Varun Mohan

English:

So I think we have an interesting organization in this regard, but mostly I would say because we are a developer-based product, I would say that's true. And then also kind of like what you said for the engineering team itself, the team structure is, it's fairly flat. We try to go with two pizza teams, teams that are fairly small just because I think the problem is when a team gets too big, the person leading the team is no longer able to get in the weeds of the technology itself. And I think in a space that's moving this quickly, I think it's dangerous to have leaders that don't understand the technology deeply and are not building. It's very, very dangerous because there's too much armchair quarterbacking. And so I think that's maybe one other decision we made.

中文翻译:

所以我觉得我们在这方面有一个有趣的组织架构，但主要还是因为我们是一个面向开发者的产品。对于工程团队本身，我们的结构非常扁平。我们尝试采用“两个披萨团队”（Two-pizza teams），即规模相当小的团队。因为我认为问题在于，当团队变得太大时，团队负责人就无法深入到技术细节中去了。在一个变化如此之快的领

域，如果领导者不深入理解技术且不亲自动手构建，那是很危险的。这非常危险，因为会产生太多的“纸上谈兵”（Armchair quarterbacking）。所以这可能是我们做的另一个决定。

[01:02:56] Varun Mohan

English:

And then teams are very, very flexible. So if we decide something is a new priority, we're very quick to change the way a team looks. And it's very centrally planned in this regard.

中文翻译:

此外，团队非常灵活。如果我们决定某件事是新的优先级，我们会非常迅速地调整团队构成。在这方面，它是高度集中规划的。

[01:03:08] Lenny Rachitsky

English:

The two pizza team concept, I saw a tweet long ago where someone from India, was like, there's always talk about two pizza teams, but pizzas in India are much smaller. And so the teams end up being smaller and they're like, "Why can't we build as much of these teams in the US?"

中文翻译:

关于“两个披萨团队”的概念，我很久以前看过一条推特，一个印度人说：大家总在谈论两个披萨团队，但印度的披萨要小得多。所以印度的团队最终规模更小，他们会问：“为什么美国的团队不能也建得这么小？”

[01:03:22] Varun Mohan

English:

Oh man.

中文翻译:

噢，天呐。

[01:03:23] Lenny Rachitsky

English:

Okay. So how many PMs do you have? So you said you have 150 employees, something like that?

中文翻译:

好。那你们有多少个 PM？你说你们有大概 150 名员工？

[01:03:28] Varun Mohan

English:

Yeah. So in terms of the product strategy function, we have three people in that role right now.

中文翻译:

是的。在产品策略职能方面，我们目前有三个人。

[01:03:34] Lenny Rachitsky

English:

I see. So it's like product... They're in their titles is product strategy, not necessarily product management?

中文翻译:

我明白了。所以他们的头衔是“产品策略”，而不一定是“产品管理”？

[01:03:41] Varun Mohan

English:

That's right.

中文翻译:

没错。

[01:03:41] Lenny Rachitsky

English:

Interesting. And then 50 engineers, you said 80-ish sales folks?

中文翻译:

有趣。然后是 50 名工程师，你说还有 80 多个销售人员？

[01:03:45] Varun Mohan

English:

Yes, that's right. And then obviously we have functions like recruiting parts of G&A, like finance. We have marketing at the company. So some other functions internally as well.

中文翻译:

是的，没错。当然我们还有招聘、行政（G&A）如财务、市场营销等职能部门。公司内部还有一些其他职能。

[01:03:56] Lenny Rachitsky

English:

It's interesting. And this is something that you hear all the time with companies like Dario for example, from Anthropic talking about how 90% of code is going to be written by AI. But all at the same time, all you guys are hiring engineers like crazy.

中文翻译:

这很有趣。你经常听到像 Anthropic 的 Dario 这样的人说 90% 的代码将由 AI 编写。但与此同时，你们所有人都在疯狂招聘工程师。

[01:04:08] Varun Mohan

English:

Yeah. Is that contradictory?

中文翻译:

是的。这矛盾吗?

[01:04:10] Lenny Rachitsky

English:

It's that contradictory, will there be an inflection point of like, "All right. Now we don't need them anymore."

中文翻译:

这确实挺矛盾的，未来会不会有一个拐点，比如“好了，现在我们不再需要他们了”？

[01:04:15] Varun Mohan

English:

I think it really comes down to, do you get incremental value by adding more engineers internally? I'm going to take... First of all, maybe just to set the record straight, if AI is writing over 90% of the code, that doesn't mean engineers are 10X as productive. Engineers spend more time than just writing code. The review code, test code, debug code, design code, deploy code, right? Navigate code. There's probably a lot of different things that engineers do. There's this one famous law in parallel computing, it's called Amdahl's Law. I don't know if you've heard about it. But it basically says if you have a graph of tasks and you have this critical path and you take any one task and parallelize it a ton, which is make it almost take zero amount of time, there's still a limit of the amount of how much faster it made the whole process go.

中文翻译:

我认为这归根结底在于：增加更多内部工程师是否能带来增量价值？首先，我要澄清一点：即使 AI 编写了超过 90% 的代码，也不意味着工程师的生产力提高了 10 倍。工程师花在写代码之外的时间更多：审查代码、测试代码、调试代码、设计代码、部署代码，对吧？还有阅读代码。工程师要做的事情有很多。并行计算中有一个著名的定律叫阿姆达尔定律（Amdahl's Law）。它基本上是说，如果你有一系列任务和一条关键路径，你把其中一个任务无限并行化（使其耗时几乎为零），整个过程的加速比仍然是有上限的。

[01:05:09] Varun Mohan

English:

So maybe put simply, let's say you have 100 units of time and only 30 units of time is being spent writing software and I took the 30 and made it three, I only took the 100 and made it 73. It's only a 27% improvement in the grand scheme of things. So I think look, we are definitely seeing over 30, maybe close to 40% productivity improvements. But I think for the vision that we're solving for, even if I were to say the company in the long tail had 200 engineers, it'd probably be too low still at that point.

中文翻译:

简单来说，假设你有 100 个单位的时间，其中只有 30 个单位花在写软件上。如果我把这 30 变成了 3，我只是把 100 变成了 73。从全局来看，这只有 27% 的提升。所以，我们确实看到了超过 30% 甚至接近 40% 的生产力提升。但对于我们正在努力实现的愿景，即使公司长远来看拥有 200 名工程师，到那时可能仍然嫌少。

[01:05:43] Varun Mohan

English:

The question is, how much more productivity do you get per person? Actually, maybe just to even say one of those thing for some of these large companies, let's say you took the CIO of a company like JPMorgan Chase, right? Her budget on software every year is $17 billion and there's over 50,000 engineers inside the company and you told her, "Hey, each of these engineers are now able to produce more technology." That's effectively what you've done, right? The right calculus that JPMorgan Chase or any of these companies will make is the ROI of building technology has actually gone up. So the opportunity cost of not investing more into technology has gone up, which means that you should just invest even more. And maybe in the short term you have even more engineers, right?

中文翻译:

问题在于，每个人能多出多少生产力？实际上，对于一些大公司来说，比如摩根大通的 CIO，她每年的软件预算是 170 亿美元，公司内部有超过 5 万名工程师。如果你告诉她：“嘿，现在这些工程师中的每一个都能产出更多的技术成果。”这实际上就是你所做的，对吧？摩根大通或任何这类公司会做出的正确计算是：构建技术的投资回报率（ROI）实际上提高了。因此，不加大技术投入的机会成本也增加了，这意味着你应该投入更多。也许在短期内，你甚至需要更多的工程师，对吧？

[01:06:08] Varun Mohan

English:

Now, that's not true across the board. There are some companies that are happy with the amount of technology they're building and there's a ceiling on the amount of technology they want to build. But for companies that actually have a very high technology ceiling, this doesn't mean you stop. This actually means you hire more.

中文翻译:

当然，这并非对所有公司都成立。有些公司对目前构建的技术量已经很满意了，他们想要构建的技术量是有天花板的。但对于那些技术天花板非常高的公司来说，这并不意味着停止招聘，反而意味着你要招更多的人。

[01:06:22] Lenny Rachitsky

English:

This is a great bull case for engineers. I feel like the canary in the coal mine for the engineering profession is when companies like yours slow down on hiring engineers.

中文翻译:

这对工程师来说是一个很好的利好消息。我觉得工程行业的“煤矿里的金丝雀”（预警信号）应该是像你们这样的公司什么时候开始放缓招聘工程师。

[01:06:30] Varun Mohan

English:

Yep.

中文翻译:

是的。

[01:06:31] Lenny Rachitsky

English:

That's not happening.

中文翻译:

而现在并没有发生。

[01:06:32] Varun Mohan

English:

[inaudible 01:06:32]. It seems like Anthropic is also hiring a lot to get it done.

中文翻译:

看起来 Anthropic 也在大量招人来实现目标。

[01:06:35] Lenny Rachitsky

English:

Yeah. Everyone is. So I think that's really promising. I think if you're in college still, makes sense to get into engineering at this point. Okay. Let me ask you this question as kind of a final question maybe. What's maybe the most counterintuitive thing you've learned about building AI products, building Windsurf and just being in a space?

中文翻译:

是的，大家都在招。所以我觉得这非常令人振奋。如果你还在上大学，现在进入工程领域仍然是有意义的。好，让我问你最后一个问题。在构建 AI 产品、构建 Windsurf 以及身处这个领域的过程中，你学到的最反直觉的事情是什么？

[01:06:54] Varun Mohan

English:

I think one of the weird things is online, everyone is very excited about the short-term wins that we are making, right? Like what we're putting out maybe weekly. We do these waves every couple of weeks. But actually a lot of the bets we're making inside the company are for things that are not three, four weeks, maybe three, six months, nine months away. That's what we're working on internally. Because I think this is kind of, Lenny, what I was mentioning to you before. One of the goals that I tell everyone at our company is we should be cannibalizing the existing state of our product every six to 12 months. Every six to 12 months, it should make our existing product look silly. It should almost make the form factor of our existing product look dumb.

中文翻译:

我觉得奇怪的一点是，在网上，大家对我们取得的短期胜利非常兴奋，比如我们每周发布的东西。我们每隔几周就会发布一波更新。但实际上，我们在公司内部做的很多豪赌，都是为了三六个月甚至九个月后的目标。这就是我们内部正在做的事情。因为正如我之前提到的，Lenny，我告诉公司每个人的目标之一是：我们应该每隔 6 到 12 个月就对现有产品进行一次“自我颠覆”。每过 6 到 12 个月，我们就应该让现有的产品显得过时，甚至让现有产品的形态看起来很笨拙。

[01:07:31] Varun Mohan

English:

So there's this weird tension where you want to have a product in market and you want to incrementally iterate and listen to users and keep making it better and better. But I would say we were the first identical IDE product out there. That's what we landed with. And I think the value of that is going to depreciate very quickly unless we continue to re-prove ourselves. And we will need to re-prove ourselves in ways in which our users are not even asking. So there's this tension here, where incremental feels very safe, right? Add this one more button. Users say, "Hey, I would like to be able to have this drop down to do X." But that is not the reason why we're going to win. That's almost table sticks. Yeah, we'll decide to do some of these. We might not decide to do a lot of these things. But it's these longer term efforts inside the company that almost disrupt the existing product that are ultimately the reason why we're going to succeed.

中文翻译:

所以这里有一种奇妙的张力：你既想让产品在市场上运行，通过增量迭代、听取用户反馈来不断完善它；但我想说，我们是第一个推出这类 IDE 产品的。如果我们不持续证明自己，它的价值会贬值得非常快。我们需要以用户甚至都没有要求过的方式来重新证明自己。这里存在一种张力：增量更新感觉非常安全，对吧？多加一个按钮，用户说“嘿，我想要一个下拉菜单来做 X”。但这并不是我们获胜的原因。那几乎只是基本要求。是的，我们会决定做其中的一些，但也可能决定不做很多。但公司内部那些几乎会颠覆现有产品的长期努力，才是我们最终成功的真正原因。

[01:08:21] Varun Mohan

English:

It's this weird tension that you need to have in your head of, you can't also not listen to your users at all because they're the reason you exist.

中文翻译:

你脑子里必须保持这种奇妙的平衡：你不能完全不听用户的，因为他们是你存在的理由。

[01:08:29] Lenny Rachitsky

English:

This reminds me of a recent podcast guest. We had Gara from captions on the podcast and he told us that he has two roadmaps. They have two roadmaps at the company. They have the real roadmap, like the typical roadmap based on feature requests and user feedback and data and things like that. And then they have the secret roadmap, which is completely not informed by users or data/ it's just them making bets on where they think the world is going.

中文翻译:

这让我想起最近的一位嘉宾，Captions 的 Gara。他告诉我们公司有两个路线图：一个是“真实路线图”，即基于功能请求、用户反馈和数据等的典型路线图；另一个是“秘密路线图”，完全不受用户或数据影响，纯粹是他们对世界走向的豪赌。

[01:08:52] Varun Mohan

English:

That's right.

中文翻译:

没错。

[01:08:52] Lenny Rachitsky

English:

And I love that he calls it the secret roadmap just to make it very mysterious and-

中文翻译:

我喜欢他把它叫做“秘密路线图”，听起来非常神秘。

[01:08:56] Varun Mohan

English:

That's smarts. That's very smart.

中文翻译:

那很聪明，非常聪明。

[01:08:57] Lenny Rachitsky

English:

Okay. I have one more question. I apologize. What's one thing that you wish he had known before starting Codeium?

中文翻译:

好。我还有一个问题，抱歉。有什么事是你希望在创办 Codeium 之前就知道的?

[01:09:04] Varun Mohan

English:

Honestly, I wish I had... Maybe humility is the wrong term, but this idea of just being okay with being wrong faster. I always think about things on when we make decisions. Me and my co-founder, we always talk about it. We're almost like, "Hey, I wish we had made the decision to do this a couple months earlier." We always talk about this. And the weird thing is outside looking and everyone's like, "Wow, actually the

decision was made at the right time." But in my head I'm always banging my head being like, "What if we had made it a couple months earlier?"

中文翻译:

老实说，我希望我能……也许“谦逊”这个词不太准确，但应该是“更坦然地接受更快速地犯错”。我总是在思考我们做决策的时候。我和联合创始人经常讨论：“嘿，我希望我们能早几个月做这个决定。”我们总是在聊这个。奇怪的是，外界看来大家都觉得：“哇，这个决定做得正是时候。”但在我脑子里，我总是在撞墙想：“如果我们早几个月做决定会怎样？”

[01:09:40] Varun Mohan

English:

I think part of that is I waxed poetically about like, "Oh, you need to be irrationally optimistic and uncompromisingly realistic." But it's very hard to do this in practice because you drink your own Kool-Aid too. Because if you're not drinking your own, you won't get up out of bed. The answer is already solved. It's not actually any of these startups. The answer is Microsoft is going to be the winner in any software category. Isn't that the answer? Just because of distribution, resources and capital, they're going to commoditize every space.

中文翻译:

我觉得部分原因是我之前夸夸其谈地说什么“你需要盲目乐观且极其现实”。但在实践中很难做到，因为你也会被自己的愿景洗脑。如果你不相信自己，你根本没法起床。如果答案已经注定，那就不是这些初创公司了。答案本该是：微软将在任何软件类别中获胜。难道不是吗？仅仅凭借分发渠道、资源和资本，他们就能让每个领域都变得平庸化。

[01:10:06] Varun Mohan

English:

So I think in some ways this amount of just understanding that, hey, re-evaluate your hypotheses and get into an uncomfortable space way more frequently is something I need to remind myself even to this day. And probably something that I didn't know coming in and starting the company. We started the company at peak zero time. At that time, probably everything seemed like it was going to moon. And there was probably a lot of irrational confidence, I would say, that we shouldn't have had.

中文翻译:

所以在某种程度上，意识到“嘿，要更频繁地重新评估你的假设并进入不舒适区”，是我直到今天仍需提醒自己的。这可能是我在创办公司之初并不了解的。我们在零利率时代的巅峰期创办了公司，那时似乎一切都会一飞冲天。我想那时可能有很多我们不该有的盲目自信。

[01:10:36] Lenny Rachitsky

English:

Varun, we covered so much ground. What an incredible conversation. I learned so much just sitting here listening and asking you questions. Is there anything else that you wanted to share I leave listeners with, any last piece of nuggets or wisdom before I let you go?

中文翻译:

Varun，我们聊了很多。这是一次非常精彩的对话。坐在这里听你讲并向你提问，我学到了很多。在结束之前，你还有什么想和听众分享的吗？最后的一点建议或智慧？

[01:10:51] Varun Mohan

English:

To be honest, I could give predictions about the space. Probably most of them are going to be wrong. I think the best thing to do is just get your hands dirty with all of these products. And I think one of the most obvious things that's going to happen is, in the next year, there will be a tremendous amount of alpha for anyone that is able to take maximum advantage of these tools. Just imagine how many of your coworkers just don't even know the existence of these tools, don't know what they can do and how much less productive they will be. And I would just say get your hands as dirty as possible, as quickly as possible.

中文翻译:

老实说，我可以对这个领域做一些预测，但大概率大部分都是错的。我认为最好的做法就是亲自动手尝试所有这些产品。我认为明年最显而易见的一件事是：任何能够最大限度利用这些工具的人，都将获得巨大的“超额收益”（Alpha）。想象一下，你有多少同事甚至不知道这些工具的存在，不知道它们能做什么，他们的生产力会低多少。我只想说，尽可能快地、尽可能深入地去动手尝试。

[01:11:24] Lenny Rachitsky

English:

And when you say get your hands dirty, basically it's like download Windsurf, start coding. Ask it to build things for you.

中文翻译:

你说的“亲自动手”，基本上就是下载 Windsurf，开始写代码，让它为你构建东西。

[01:11:29] Varun Mohan

English:

Yeah, build apps. Build apps. Start using it for maybe even making mocks, modifying your existing code base. There's probably ways in which you could be a force multiplier to your organization and ways in which they never even anticipated, right? Imagine if you were a product manager that could actually very quickly make edits to the code base and just start pushing changes yourself. You probably get a tremendous amount of respect from your own engineering peers. You could probably get way more done because of that. I feel like there's no sort of ceiling at that point.

中文翻译:

是的，构建应用。开始用它做原型，修改现有的代码库。你可能会以组织从未预料到的方式成为“力量倍增器”。想象一下，如果你是一个 PM，能够非常快速地修改代码库并亲自推送更改，你可能会赢得工程同事的极大尊重。你也能因此完成更多工作。我觉得在那一点上，是没有天花板的。

[01:12:00] Lenny Rachitsky

English:

I think this is such an underestimated point you're making here. There's apps that can build things from scratch and then there's apps like this that can edit your existing code base if you're a PM at... What's the largest company you work with, people-wise?

中文翻译:

我认为你提出的这一点被严重低估了。有些应用可以从零开始构建，而像这样的应用可以修改你现有的代码库。如果你是……你们合作的人数最多的公司是哪家？

[01:12:15] Varun Mohan

English:

Publicly, let's just say JPMorgan Chase.

中文翻译:

公开资料显示，就说是摩根大通吧。

[01:12:16] Lenny Rachitsky

English:

Okay.

中文翻译:

好。

[01:12:19] Varun Mohan

English:

They have over 50,000 developers.

中文翻译:

他们有超过 5 万名开发者。

[01:12:20] Lenny Rachitsky

English:

Okay. So you could be a PM at JPMorgan Chase and be like, "I have a problem I need to solve. I want to move this metric. I want to change the step in the signup flow." You just open up Windsurf and tell it to do the thing you want. And then can you push straight to GitHub and do a-

中文翻译:

好。所以你可能是摩根大通的一个 PM，你想：“我有一个问题要解决，我想提升这个指标，我想改变注册流程中的这一步。”你只需要打开 Windsurf，告诉它你想做什么。然后你能直接推送到 GitHub 并做一个——

[01:12:37] Varun Mohan

English:

Yeah. Actually, you could do that too.

中文翻译:

是的，实际上你也可以这么做。

[01:12:39] Lenny Rachitsky

English:

... merge [inaudible 01:12:39]-

中文翻译:

……合并……

[01:12:39] Varun Mohan

English:

Yeah.

中文翻译:

是的。

[01:12:39] Lenny Rachitsky

English:

Okay. PR?

中文翻译:

好。PR（拉取请求）？

[01:12:40] Varun Mohan

English:

Yeah, it could make a PR for you.

中文翻译:

是的，它可以为你创建一个 PR。

[01:12:41] Lenny Rachitsky

English:

Oh, my God. This is insane. Folks, future is out of control. Okay. Man, that was such an important point at the end there because I think people may not realize this. They see all these other apps, they're like, "Oh, [inaudible 01:12:51], prototypes," but this is legitimately something a PM can actually do work with.

中文翻译:

天呐，这太疯狂了。各位，未来已经失控了。伙计，最后那一点太重要了，因为我觉得大家可能还没意识到。他们看到其他那些应用，会觉得“哦，只是原型”，但这个工具是 PM 真正可以用来工作的。

[01:12:55] Varun Mohan

English:

Yeah. When you think about the people, at least that, I don't know, Lenny, who you respect the most, they're the people that somehow, despite their title, their level of agency and just output just all the way down to the weeds to the highest level strategy is just perfection, right? They know when to go all the way down. And I think sometimes you see people that talk about roles and they irrationally feel like, "Oh, because I'm this role, I'm not allowed to touch this." Well now everything's open season, right? And I think this is an opportunity to almost go all the way down to the weeds and all the way up to the top and just be effective on every level.

中文翻译:

是的。当你想到那些你最尊敬的人时，Lenny，我不知道你最尊敬谁，但他们通常是那些无论头衔如何，其自主性和产出从最细微的细节到最高层的战略都追求完美的人，对吧？他们知道什么时候该深入底层。我觉得有时你会看到人们谈论角色，并盲目地觉得“哦，因为我是这个角色，我不被允许碰那个”。现在，一切都开放了，对吧？我认为这是一个机会，让你既能深入细节，又能统揽全局，在每个层面上都发挥效力。

[01:13:29] Lenny Rachitsky

English:

Unbelievable. All right. Well with that, we'll leave folks. Varun, thank you so much for being here.

中文翻译:

不可思议。好，我们就聊到这里。Varun，非常感谢你能来。

[01:13:35] Varun Mohan

English:

Awesome. Thanks a lot, Lenny.

中文翻译:

太棒了。非常感谢，Lenny。

[01:13:36] Lenny Rachitsky

English:

What an incredible conversation. Thanks, Varun. 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. Also, please consider giving us a rating or leaving a review as that really helps other listeners find the podcast. You can find all past episodes or learn more about the show at lennyspodcast.com. See you in the next episode.

中文翻译:

多么精彩的对话。谢谢 Varun。大家再见。非常感谢大家的收听。如果你觉得内容有价值，可以在 Apple Podcasts、Spotify 或你喜欢的播客应用中订阅本节目。此外，请考虑给我们评分或留下评论，这能极大地帮助其他听众发现这个播客。你可以在 lennyspodcast.com 找到所有往期节目或了解更多信息。下期节目再见。