BRET TAYLOR

LENNY'S PODCAST

BILINGUAL TRANSCRIPT

ORIGINAL BY

Lenny Rachitsky

@lennysan • x.com/lennysan

ANALYSIS BY

@Penny777 • x.com/penny777

Generated: 2026-01-16

Bret Taylor - 双语对照

This is the complete bilingual transcript of Lenny's Podcast featuring Bret Taylor.

[00:00:00] Lenny Rachitsky

English:

You're CTO of Meta. You are a co-CEO of Salesforce, you're chairman of the board at OpenAI. How do you think the AI market is going to play out?

中文翻译:

你曾任 Meta 的 CTO，曾任 Salesforce 的联席 CEO，现在是 OpenAI 的董事会主席。你认为 AI 市场未来会如何发展?

[00:00:07] Bret Taylor

English:

The whole market is going to go towards agents. I think the whole market is going to go towards outcomes-based pricing. It's just so obviously the correct way to build and sell software.

中文翻译:

整个市场都将转向智能体（Agents）。我认为整个市场都将转向基于结果的定价（outcomes-based pricing）。这显然是构建和销售软件的正确方式。

[00:00:16] Lenny Rachitsky

English:

This makes me think about, I had Marc Benioff on the podcast. You guys were co-CEOs. He was extremely agent-pilled.

中文翻译:

这让我想起我曾邀请 Marc Benioff 来过播客。你们曾是联席 CEO。他对智能体简直到了痴迷的程度（agent-pilled）。

[00:00:21] Bret Taylor

English:

It's so hard to sell productivity software, which I learned the hard way.

中文翻译:

销售生产力软件非常困难，这是我通过惨痛教训学到的。

[00:00:24] Lenny Rachitsky

English:

What's a story that comes to mind when you think about your biggest mistake?

中文翻译:

当你想到自己最大的错误时，脑海中会浮现出哪个故事？

[00:00:27] Bret Taylor

English:

I was the product manager for, it was called Google Local had a pretty tough product review with Marissa and Larry, and to not do that well with a link from the Google homepage is embarrassing.

中文翻译:

我当时是 Google Local 的产品经理。我和 Marissa（梅耶尔）以及 Larry（佩奇）进行了一次非常艰难的产品评审。在拥有 Google 首页链接入口的情况下还没做好，这真的很丢脸。

[00:00:37] Lenny Rachitsky

English:

I think it's really empowering for people to hear it's possible to succeed in spite of a massive failure like this.

中文翻译:

我觉得让大家听到在经历如此巨大的失败后仍有可能获得成功，是非常给人力量的。

[00:00:41] Bret Taylor

English:

They gave me another shot to do the V2 of it that resulted in Google Maps. We got about 10 million people using it on the first day.

中文翻译:

他们给了我第二次机会去做 V2 版本，也就是后来的 Google 地图。上线第一天就有大约 1000 万人使用。

[00:00:48] Lenny Rachitsky

English:

What mindset contributed to you being successful in such a variety of roles?

中文翻译:

是什么样的心态让你在如此多样化的角色中都能取得成功？

[00:00:52] Bret Taylor

English:

Waking up every morning, what is the most impactful thing I could do today?

中文翻译:

每天早上醒来都会问：今天我能做的最有影响力的事情是什么？

[00:00:56] Lenny Rachitsky

English:

Today, my guest is Bret Taylor. Bret is an absolute legendary builder and founder. He co-created Google Maps at Google. He co-founded the social network, FriendFeed invented the like button and the real-time newsfeed, which he sold to Facebook. He then became CTO at Facebook. He then started a productivity company called Quip, which he sold to Salesforce for $750 million. He then became co-CEO of Salesforce. He's also currently chairman of the board at OpenAI. At one point he was chairman of the board at Twitter. Today, co-founder and CEO of Sierra an AI started building agent to help companies with customer service sales and more.

(00:01:32):

In our conversation, we cover so much ground, including what skills and mindsets have most helped Bret be so successful in so many roles, why we're all still sleeping on the impact that agents are going to have on the business world. How coding is going to change in the coming years, where the biggest opportunities remain for startups, lessons on pricing and go-to-market in AI, the story behind the like button and so much more. This is a truly epic conversation with a legendary builder.

(00:01:57):

If you enjoy this podcast, don't forget to subscribe and follow it in your favorite podcasting app or YouTube. Also, if you become an annual subscriber of my newsletter, you get a year free of a bunch of incredible products, including Replit, Lovable, Bolt, n8n, Linear, Superhuman, Descript, Wispr Flow, Gamma, Perplexity, Warp, Granola, Magic Patterns, Raycast, ChatPRD, Mobbin and more. Check it out at Lenny'snewsletter.com and click bundle. With that, I bring you Bret Taylor.

中文翻译:

今天的嘉宾是 Bret Taylor。Bret 是一位绝对传奇的构建者和创始人。他在 Google 联合创建了 Google 地图；联合创办了社交网络 FriendFeed，发明了“点赞”按钮和实时新闻流（Newsfeed），后来将其卖给了 Facebook。随后他成为了 Facebook 的 CTO。接着他创办了生产力软件公司 Quip，并以 7.5 亿美元的价格卖给了 Salesforce。之后他成为了 Salesforce 的联席 CEO。他目前还是 OpenAI 的董事会主席，并曾担任 Twitter 的董事会主席。现在，他是 Sierra 的联合创始人兼 CEO，这是一家构建 AI 智能体以帮助公司处理客户服务、销售等业务的初创公司。

(00:01:32):

在我们的对话中，我们涵盖了非常广泛的内容，包括哪些技能和心态最能帮助 Bret 在这么多角色中取得成功；为什么我们仍然低估了智能体将对商业世界产生的影响；未来几年编程将如何发生变化；初创公司最大的机会在哪里；AI 定价和进入市场（GTM）策略的经验教训；“点赞”按钮背后的故事等等。这是一场与传奇构建者的史诗级对话。

(00:01:57):

如果你喜欢这个播客，别忘了在常用的播客应用或 YouTube 上订阅和关注。此外，如果你成为我时事通讯（Newsletter）的年度订阅者，你可以免费获得一年的一系列优秀产品，包括 Replit, Lovable, Bolt, n8n, Linear, Superhuman, Descript, Wispr Flow, Gamma, Perplexity, Warp, Granola, Magic Patterns, Raycast, ChatPRD, Mobbin 等等。请访问 Lennysnewsletter.com 并点击 bundle 查看。下面，让我们欢迎 Bret Taylor。

[00:02:22] Lenny Rachitsky (Sponsor: CodeRabbit)

English:

This episode is brought to you by CodeRabbit, the AI code review platform transforming how engineering teams ship faster with AI without sacrificing code quality. Code reviews are critical, but time-consuming CodeRabbit acts as your AI co-pilot providing instant code review comments and potential impacts of every pull request. Beyond just flagging issues, CodeRabbit provides one click fix suggestions and lets you define custom code quality rules using AST GREP patterns, catching subtle issues that traditional static analysis tools might miss.

(00:02:55):

Coderabbit also provides free AI code reviews directly in the IDE. It's available in VS. Code, Cursor and Windsurf. CodeRabbit has so far reviewed more than 10 million PRs installed on 1 million repositories and is used by over 70,000 open source projects. Get CodeRabbit for free for an entire year at CodeRabbit.ai using code, Lenny. That's CodeRabbit.ai.

中文翻译:

本期节目由 CodeRabbit 赞助。CodeRabbit 是一款 AI 代码审查平台，它正在改变工程团队在不牺牲代码质量的前提下，利用 AI 加快交付速度的方式。代码审查至关重要但非常耗时，CodeRabbit 充当你的 AI 副驾驶，为每个拉取请求（PR）提供即时的审查评论和潜在影响分析。除了标记问题，CodeRabbit 还提供一键修复建议，并允许你使用 AST GREP 模式定义自定义代码质量规则，捕捉传统静态分析工具可能遗漏的细微问题。

(00:02:55):

CodeRabbit 还直接在 IDE 中提供免费的 AI 代码审查，支持 VS Code、Cursor 和 Windsurf。到目前为止，CodeRabbit 已审查了超过 1000 万个 PR，安装在 100 万个代码库中，并被超过 7 万个开源项目使用。使用代码 "Lenny"，在 CodeRabbit.ai 免费获取一整年的 CodeRabbit 服务。

[00:03:20] Lenny Rachitsky (Sponsor: Basecamp)

English:

This episode is brought to you by Basecamp. Basecamp is the famously straightforward project management system from 37signals. Most project management systems are either inadequate or frustratingly complex, but Basecamp is refreshingly clear. It's simple to get started, easy to organize and Basecamp's visual tools help you see exactly what everyone is working on and how all work is progressing. Keep all your files and conversations about projects directly connected to the projects themselves so that you always know where stuff is and you're not constantly switching contexts. Running a business is hard. Managing your project should be easy. I've been a long time fan of what 37signals has been up to and I'm really excited to be sharing this with you. Sign up for a free account at basecamp.com/Lenny. Get somewhere with Basecamp.

中文翻译:

本期节目由 Basecamp 赞助。Basecamp 是来自 37signals 的著名且直观的项目管理系统。大多数项目管理系统要么功能不足，要么复杂得令人沮丧，但 Basecamp 却清晰得令人耳目一新。它上手简单，易于组织，其可视化工具能帮你准确看到每个人在做什么以及工作进展。将所有文件和项目对话直接与项目本身关联，这样你就永远知道东西在哪，而不需要不断切换上下文。经营业务已经很难了，管理项目应该简单点。我一直是 37signals 的粉丝，很高兴能与大家分享。在 basecamp.com/Lenny 注册免费账号。用 Basecamp 助你一臂之力。

[00:04:10] Lenny Rachitsky

English:

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

中文翻译:

Bret，非常感谢你能来，欢迎来到播客。

[00:04:14] Bret Taylor

English:

Thanks for having me.

中文翻译:

谢谢邀请。

[00:04:15] Lenny Rachitsky

English:

My pleasure. There's so much that I want to talk about. You've done so many incredible things over the course of your career. Just boggles the mind, the things that you've done, and we're going to talk about a lot of that sort of stuff, but I want to actually start with the opposite. I want to talk about a time that you messed up, a time that you screwed up in a big way. We have this recurring segment on the podcast called Fail Corner, and so, I thought it'd be fun to just start there before we get into all the great stuff you've done. What's a story that comes to mind when you think about maybe your biggest mistake in building a product?

中文翻译:

我的荣幸。我想聊的内容太多了。在你职业生涯中做了这么多不可思议的事情，简直令人惊叹。我们会聊到很多这类话题，但我其实想从相反的角度开始。我想聊聊你搞砸的时候，也就是你犯大错的时候。我们播客有一个固定环节叫“失败角落”（Fail Corner），所以在进入正题之前，我想先从这里开始。当你想到在构建产品过程中犯过的最大错误时，你会想到哪个故事？

[00:04:43] Bret Taylor

English:

It may not be the biggest, but it was my first prominent mistake as a product manager at Google. So for me, it feels big because it was very formative for me as a product designer. So I joined Google in late 2002, early 2003, and I was one of the earliest associate product managers at the company and first, was working on the search system, essentially expanding our index from 1 billion webpages to 10 billion, which was a big deal at the time. It seems quaint now. Then I did a decent job and so my boss, Marissa Meyer, gave me the opportunity to lead a new product initiative, which was a big bet on me and it was both an opportunity to do something for Google, but I was also being pretty scrutinized just as a young new product manager and the premise given to me was work on local search.

(00:05:40):

At the time, the Yellow Pages was still dominant and well, Google was really good at searching the web. It wasn't really good for finding a plumber or a restaurant just because it wasn't really a huge part of the internet at the time. So this content wasn't necessarily on the internet and even if it was, you didn't really want to find plumbers in Manhattan, you wanted to find plumbers in San Francisco, if you're me. And so it was both a technical problem and a product problem and a content problem.

(00:06:11):

We launched the first version of that product that I was the product manager for, was called Google Local and it was, I'll be a little bit more critical now than I might've been at the time, but it was a little bit of a me too version of Yahoo Yellow Pages, essentially grafting on Yellow Pages search on top of Google Search and with a properly crafted query you could see those listings at the top of your search results but a standalone site at local.google.com. Actually, it was an important enough initiative that actually, there was on the Google homepage, it had Web, Images and Local was up there as well, so it's got top bill.

(00:06:54):

I mean you could put almost any link on the Google homepage and get a lot of traffic to it. And despite that, it didn't do that well and to not do that well with a link from the Google homepage is embarrassing. There's not much one can do more than giving you that kind of traffic to give you an add that as a product leader or a product manager. And the product was fine, it worked, but it really wasn't differentiated. And in many ways, I think, again, I think I've had these reflections more sense than at the time, that I had some of the time, but why use this instead of Yahoo Yellow Pages? But more than anything else, why use this instead of the Yellow Pages? It was a digital version of something that had come before. Had a pretty tough product review with Marissa and Larry and others and it was fine. I wasn't about to get fired or something, but it was, I don't know, the shine on my reputation was waning a little bit.

中文翻译:

这可能不是最大的，但它是我作为 Google 产品经理犯下的第一个显著错误。对我来说，它感觉很大，因为它对我作为产品设计师的成长起到了塑造作用。我在 2002 年底或 2003 年初加入 Google，是公司最早的一批助理产品经理（APM）之一。起初我负责搜索系统，主要是将索引从 10 亿个网页扩展到 100 亿个，这在当时是件大事，现在看来倒显得很平常了。后来我表现不错，我的老板 Marissa Meyer 给了我领导一个新产品项目的机会，这对我来说是一次巨大的赌注。这既是为 Google 做贡献的机会，作为一个年轻的新产品经理，我也受到了严密的审视。给我的前提是：做本地搜索（Local Search）。

(00:05:40):

当时，《黄页》（Yellow Pages）仍然占据主导地位。虽然 Google 非常擅长搜索网页，但不擅长找水管工或餐厅，因为当时这些内容还不是互联网的重要组成部分。这些内容不一定在网上，即使在，如果你在旧金山，你也不想找曼哈顿的水管工。所以这既是一个技术问题，也是一个产品问题和内容问题。

(00:06:11):

我们推出了我担任产品经理的第一个版本，叫 Google Local。现在回过头来看，我会比当时更挑剔一些：它有点像雅虎黄页（Yahoo Yellow Pages）的跟风版，基本上是把黄页搜索嫁接在 Google 搜索之上。通过精心设计的查询，你可以在搜索结果顶部看到这些列表，但它也有一个独立站点 local.google.com。事实上，这是一个

非常重要的项目，以至于在 Google 首页上，除了“网页”、“图片”，还有“本地”选项，它占据了顶级位置。

(00:06:54):

我的意思是，你几乎可以在 Google 首页放任何链接并获得巨大流量。尽管如此，它的表现并不好。在拥有 Google 首页链接的情况下还没做好，这真的很尴尬。作为产品领导者或产品经理，除了给你这种级别的流量，别人没法给你更多支持了。产品本身没问题，能用，但确实没有差异化。在很多方面，我现在回想起来（当时也有一些这种感觉）：为什么要用这个而不是雅虎黄页？更重要的是，为什么要用这个而不是纸质黄页？它只是以前事物的数字化版本。我和 Marissa、Larry 等人进行了一次非常艰难的产品评审。虽然我没到被解雇的地步，但我的名声确实有点受损。

[00:07:52] Bret Taylor

English:

And they gave me another shot to do a V2 of it and I got the impression it wasn't like my last shot, but I certainly was feeling a little dejected from going from a hot shot, new PM to a new thing. So, we spent a lot of time thinking about how can you make something that's just much more compelling and not just a digital version of the Yellow Pages and not just so similar to some of the other products out there. And that ended up being the thread that we pulled that resulted in Google Maps. We had licensed from MapQuest, the ability to put this little map next to the search results, it was always the ugliest part of the product and we always made these backhanded comments about it internally and we spent a lot of time saying, what if we inverted the hierarchy here and made the map the canvas?

(00:08:49):

We ended up finding Lars and Jens Rasmussen who had been working on this Windows mapping product and we got them into the company and started exploring this space and it ended up where, through that exploration we ended up integrating a lot of different products. We ended up integrating mapping local search, driving directions, like all of these products at the time were actually separate product categories and ended up with something that redefined the industry and certainly my career. But it took, I think for me as a product leader, it changed the way I think about product just because there's feature and functionality and then there's like why should I use this thing in the first place? And it was notable, there was a couple of interesting moments.

(00:09:34):

When we launched Google Maps we got about 10 million people using it on the first day, which at that scale of the internet at the time was huge. And then in August of 2005, we integrated satellite imagery from a recent acquisition called Keyhole, which became Google Earth and we got 90 million people using it on the same day. Everyone wanted to look at the top of their house when the imagery came out. And it was really interesting because there's so many subtle product lessons in there. First is, I think as you have these new technologies, rather than literally digitizing what came before, if you can create an entirely new experience, it answers the question for a new customer, why should I give this the time of day? And so, really disassembling the Lego set and reassembling into something new rather than just digitizing what was there before. Certainly, that was the lesson I think in Google Maps, really was native to the platform in a way that a paper map couldn't be and that was a really meaningful breakthrough.

(00:10:35):

And then with satellite imagery, it honestly wasn't the most important part of Google Maps, but it was the sizzle to the steak and it created, I don't think the term viral was a thing people said back then, but it created a viral moment. We were on Saturday Night Live, which is the coolest thing. Andy Samberg in, I think it was called Lazy Sunday rapped about Google Maps and Lars and I were texting each other. We did it. We're on Saturday Live. Mission accomplished. And it was also showing that as you're there thinking about products, there's the why you decide to use a product and then what is the enduring value. And those are deeply related but not all the same thing. And I just learned so many lessons that I took with me for every subsequent product that I worked on.

中文翻译:

他们给了我第二次机会去做 V2 版本。我感觉这不一定是最后一次机会，但从一个备受瞩目的新 PM 变成一个受挫的人，我确实感到有些沮丧。所以，我们花了很多时间思考：如何做出一款更具吸引力的产品，而不仅仅是黄页的数字化版本，也不仅仅是和市面上其他产品雷同的东西。这就是我们最终引出 Google 地图的那条线索。我们当时从 MapQuest 获得了授权，可以在搜索结果旁边放一个小地图，那总是产品中最丑的部分，我们内部经常私下吐槽。我们花了很多时间讨论：如果我们反转层级，把地图作为画布（Canvas）会怎样?

(00:08:49):

我们最终找到了 Lars 和 Jens Rasmussen 兄弟，他们当时正在开发一款 Windows 地图产品。我们把他们招进公司，开始探索这个领域。通过那次探索，我们整合了许多不同的产品：地图、本地搜索、驾车导航——当时这些其实都是独立的产品类别。最终我们做出了一款重新定义行业的产品，当然也重新定义了我的职业生涯。但作为一名产品领导者，它改变了我思考产品的方式：除了功能和特性，还有“为什么我首先要使用这个东西？”。当时有几个有趣的时刻。

(00:09:34):

当我们推出 Google 地图时，第一天就有大约 1000 万用户，这在当时的互联网规模下是巨大的。然后在 2005 年 8 月，我们整合了来自新收购公司 Keyhole 的卫星图像（后来变成了 Google 地球），当天就有 9000 万人使用。图像一出来，每个人都想看看自己家的房顶。这非常有趣，因为其中包含许多细微的产品教训。首先，我认为当你拥有新技术时，与其照搬旧事物的数字化，不如创造一种全新的体验，这回答了新客户的问题：我为什么要关注这个？所以，是拆解乐高积木并重新组装成新东西，而不是仅仅数字化。这正是 Google 地图的教训：它以纸质地图无法实现的方式成为了平台的原生体验，这是一个非常有意义的突破。

(00:10:35):

至于卫星图像，坦白说它不是 Google 地图最重要的部分，但它是“牛排上的滋滋声”（sizzle to the steak，指卖点），创造了一个——我不确定当时人们是否用“病毒式传播”这个词——但它创造了一个爆红时刻。我们甚至登上了《周六夜现场》（SNL），那是超级酷的事情。Andy Samberg 在《Lazy Sunday》里唱了关于 Google 地图的饶舌，我和 Lars 互相发短信说：“我们做到了，我们上 SNL 了，任务完成。”这也表明，当你思考产品时，既有“为什么决定使用产品”的原因，也有“持久的价值”是什么。这两者紧密相关，但并不完全是一回事。我学到了很多教训，并把它们带到了之后参与的每一个产品中。

[00:11:18] Lenny Rachitsky

English:

That is an awesome story. One I think it's really empowering for people to hear, even you Bret, who I'm going to share all the successes you've had, have had a massive failure with the CEO of Google [inaudible 00:11:30] just like Bret, you screwed up. And it was such a big bet. So one, just it's possible to succeed as you have succeeded in spite of a massive failure like this. And then some of the product lessons you shared, just to highlight a few of these things because I think this is great, is just you will often not win if you just make something that's a better copy of something else, what you want to look for is something that is an entirely new experience, something that's differentiated, something that's a lot more compelling. Let's flip to talk about what you've learned from actually being very successful at a lot of things.

(00:12:02):

So I was looking at your resume and you basically have been very successful at every level of the career ladder and in such a huge variety of roles. So let me just read a few of these things for folks that aren't super familiar with your background. You were CTO of Meta, you were co-CEO of Salesforce, you're also CPO and COO at Salesforce. At Google, you joined as an associate product manager where you famously, you didn't mention this but you rebuilt Google Maps on weekend. We're not going to talk about that. You're chairman of the board at OpenAI. You were chairman of the board at Twitter. You've also founded three different companies, one social network, one productivity docs company called Quip, and now, Sierra. Fun fact, at FriendFeed you invented the like button and I don't know if people know that and also just the newsfeed, I'll just throw that out there to give you some credit.

(00:12:50):

So you're basically an associate product manager, an IC product manager, an engineer, CPO, COO, CTO, CEO of three different companies including a public company. Very rare that somebody is successful at all these types of roles and all these levels. So, let me just ask you this question. What mindsets or habits or just ways of working have you worked on building in yourself that you think have most contributed to you being successful in such a variety of roles and levels?

中文翻译:

这是一个很棒的故事。我觉得让大家听到这些非常有力量，即使是你 Bret，在拥有这么多成功之后，也曾经历过被 Google CEO 评价为“Bret，你搞砸了”的巨大失败。而且那还是一个如此重大的赌注。所以第一点，即使经历了这样的惨败，依然可能取得像你这样的成功。然后是你分享的一些产品教训，我想强调其中几点，因为这太棒了：如果你只是做一个比别人更好的复制品，你通常不会赢；你要寻找的是全新的体验、差异化的东西、更具吸引力的东西。现在让我们转而聊聊你从那些非常成功的事情中学到了什么。

(00:12:02):

我看过你的简历，你基本上在职业阶梯的每一个层级、在如此多样的角色中都取得了巨大的成功。让我为那些不太熟悉你背景的听众读一下：你曾是 Meta 的 CTO，Salesforce 的联席 CEO，还担任过 Salesforce 的 CPO 和 COO。在 Google，你以助理产品经理（APM）的身份加入，在那里你最出名的事迹（虽然你没提）是你在周末重写了 Google 地图。你还是 OpenAI 的董事会主席，曾任 Twitter 的董事会主席。你还创办了三家不同的公司：一家社交网络，一家名为 Quip 的生产力文档公司，以及现在的 Sierra。有趣的是，在 FriendFeed，你发明了“点赞”按钮，我不确定大家是否知道这一点，还有新闻流（Newsfeed），我也得提一下，这都是你的功劳。

(00:12:50):

所以你当过 APM、独立贡献者（IC）产品经理、工程师、CPO、COO、CTO，以及三家不同公司（包括一家上市公司）的 CEO。一个人能在所有这些角色和层级上都取得成功是非常罕见的。所以，我想问你：你培养了什么样的心态、习惯或工作方式，让你在如此多样的角色和层级中都能取得成功?

[00:13:19] Bret Taylor

English:

Yeah. It's actually something I am proud of. I like the fact I've worn different hats. It's actually amusing when I meet colleagues that I've known from one of those jobs, they'll often think of me through the lens of that job. And so, I'll go to meet folks from Facebook and they think of me largely as an engineer. I'll meet folks from Google, they think me largely as a product person. At Salesforce, a lot of the folks there interacted with me as like a, lack of a better word, a suit, the boss. And I'm not sure they think of me as an engineer at all, even though I was still probably coding on the weekends for fun. And one of the things that is a principle for me is to have a really flexible view of my own identity. I probably would self-describe as an engineer, but more broadly I think of myself as a builder and I like to build products and I think companies are one of the most effective ways to build products.

(00:14:17):

There's also things like open source, but I think I'm a huge believer in the confluence of technology and capitalism to produce just incredible outcomes for customers. And as a consequence, I think to really build something of significance, I think to be a great founder, you really need to be able to not have such a ossified view of your identity that you can't transform into what the company needs you to be at that point. And every founder you'll talk to, one day, I think selling is a big part of being a founder. You have to sell investors on wanting to invest in your company. You have to sell candidates on wanting to work at your company. You have to sell customers to want to use the product that your customer produces. You have to have good design taste, not just for your product but for your marketing and essentially, soliciting your customers.

(00:15:09):

You have to have good engineering. If you're building a technology company, the technology comes first. It's why this industry is so transformative. I probably credit, and I've told this story before, but I'm very grateful for her, but I probably credit Sheryl Sandberg for really changing the way I approach new jobs. The story, and I might be embellishment a little bit, but I think it's broadly accurate. So I had just become the chief technology officer of Facebook and when I first got the job, it was the flavor of CTO or that relatively small group reporting into me, but contributed almost as a very senior architect on a number of projects. And then at some point, Mark Zuckerberg reorganized the company and split it into a bunch of different groups. I ended up with a very large group, 100 under me and I was essentially running our platform and mobile groups, products, design engineering.

(00:16:10):

So I went from a handful of reports to, I don't know, over 1,000 or something. It was a big group. And it was the largest management job. I had become a manager at Google, but a modest team. And so, I was doing okay but not great. And I had this moment where Sheryl saw me, I think I was editing a presentation for a partner just because the presentation I got didn't meet my quality bar and I was editing it and griping about it. She pulled me into a room and gave me talking to a little bit about holding my team to as high of a standard as I have. If someone wasn't meeting my expectations, what was my plan to manage them out of the company or... Just giving me management one-on-one.

(00:16:57):

And she's a remarkable mentor in the sense she can give you feedback that's very direct and often, a bit uncomfortable, but you know she cares about you. And so it was the type of feedback you listen to. I went home that night and I was stewing on it and not very happy. I was like, you get naturally a little defensive in those moments. Is that really true? Am I really fucking it up or is she overreacting? And then I woke up the next day, I was like, "No, she's right." And I had realized this subconscious limiter that was limiting my success in the job, which is, I was trying to conform the job to the things I thought I liked to do. So, I was spending a lot of my time on some product and technology things that I was passionate about, thinking I'm the boss. I should focus on what I want to focus on instead of thinking about, okay, I'm running the mobile and platform teams at Facebook. What's the most important thing to do today to make our mobile and developer platform successful?

(00:18:03):

And when I reframed the job that way, I did different things. And the thing that was the biggest pleasant surprise to me was I liked it. I thought I liked engineering and product, but in fact, when I changed an organization and it turned out to be more successful, I derived a great deal of joy from seeing that success. Our developer platform had a lot of partners and when there was an issue there and I'd spend time on partnerships and it worked and our platform became healthier, the partner became more successful. I took pride in that success and then I just started being better at my job and I realized that the actual act of engineering or product design or all the things I thought I liked, what I really liked is impact.

(00:18:49):

And so, that conversation led to my waking up every morning, sometimes literally, but certainly, in the broadest sets of the words saying, "What is the most impactful thing I can do today?" And really thinking almost like, if you had an external board of advisors telling you what are the things where if you focus on them, you can maximize the likelihood that what you're trying to achieve will happen? Sometimes, it's recruiting, sometimes, it's product, sometimes, it's engineering, sometimes, it's sales. And I've become much more self-reflective just about what is important to work on. And I have become much more receptive to doing things that I previously would've said aren't my favorite things to do because I derive so much joy from having an impact that I enjoy a lot more things now. And so, I really credit Sheryl, I'm so grateful. And actually, it's interesting, I think a lot about this when I give feedback to people now, just those moments that can change the trajectory of your career. I give her all the credit for it.

中文翻译:

是的，这确实是我引以为豪的一点。我喜欢尝试不同的角色。有趣的是，当我遇到以前工作中认识的同事时，他们通常会通过那个职位的视角来看待我。比如，我遇到 Facebook 的人，他们大多认为我是个工程师；遇到 Google 的人，他们认为我是个产品人；在 Salesforce，很多人把我当成——找不到更好的词——一个“穿西装的”，也就是老板。我不确定他们是否认为我是个工程师，尽管我周末可能还在为了好玩而写代码。

对我来说，一个原则是：对自己的身份保持灵活的看法。我可能会自称为工程师，但更广泛地说，我认为自己是一个“构建者”（Builder）。我喜欢构建产品，而我认为公司是构建产品最有效的方式之一。当然还有开源等方式，但我坚信技术与资本主义的结合能为客户创造不可思议的结果。因此，要真正构建有意义的东西，要成为一名伟大的创始人，你不能对自己的身份有僵化的看法，以至于无法转型为公司在那个阶段需要你成为的角色。

你遇到的每一位创始人都会告诉你，销售是创始人工作的重要组成部分。你必须向投资者推销，让他们想投资你的公司；你必须向候选人推销，让他们想来你的公司工作；你必须向客户推销，让他们想用你的产品。你必须有良好的设计品味，不仅是为了产品，也是为了营销和吸引客户。你必须有扎实的工程能力。如果你在创办一家技术公司，技术永远是第一位的。这就是为什么这个行业如此具有变革性。

我可能要归功于 Sheryl Sandberg（谢丽尔·桑德伯格），我以前讲过这个故事，我非常感激她，她真正改变了我对待新工作的方式。故事大概是这样的（可能有点艺术加工，但大体准确）：我刚成为 Facebook 的 CTO，起初这个职位更像是一个高级架构师，管的人很少。后来 Mark Zuckerberg 重组了公司，我最终接管了一个巨大的团队，负责平台和移动部门，包括产品、设计和工程。我从管几个人变成了管 1000 多人。这是我做过最大的管理工作。虽然我在 Google 当过经理，但团队规模很小。所以，我当时做得还可以，但谈不上出色。

有一次 Sheryl 看到我正在修改一个给合作伙伴的演示文稿，因为我觉得拿到的版本没达到我的质量标准，我一边改一边抱怨。她把我拉进房间，跟我谈了谈：要像要求自己一样高标准要求团队；如果有人没达到预期，我的计划是管理他们离开还是……基本上是给我上了一堂管理入门课。她是一位了不起的导师，她能给你非常直接、甚至让人不舒服的反馈，但你知道她是关心你的。所以那是你会听进去的反馈。

那天晚上我回到家，心里一直在琢磨，不太高兴。在那种时刻，人自然会产生防御心理：那是真的吗？我真的搞砸了吗？还是她反应过度了？但第二天醒来，我想：“不，她是对的。”我意识到一个潜意识里的限制因素，它限制了我在工作中的成功，那就是：我试图让工作去适应我自认为喜欢做的事情。我把大量时间花在我热衷的产品和技术上，心想“我是老板，我应该关注我想关注的事情”。但我没有思考：“我现在负责 Facebook 的移动和平台团队，为了让这些业务成功，今天我能做的最重要的事情是什么？”

当我以这种方式重新定义工作时，我开始做不同的事情。最让我惊喜的是，我竟然喜欢上了这些事。我原以为我只喜欢工程和产品，但事实上，当我改变一个组织并让它变得更成功时，我从这种成功中获得了巨大的快乐。我们的开发者平台有很多合作伙伴，当出现问题时，我花时间处理伙伴关系，结果平台变得更健康，伙伴也更成功。我为这种成功感到自豪。我开始更擅长我的工作，并意识到：我真正喜欢的不是工程或产品设计本身，而是"影响力"（Impact）。

所以，那次谈话让我养成了每天早上醒来问自己"今天我能做的最有影响力的事情是什么？"的习惯。这就像是有一个外部顾问委员会告诉你：如果你专注于这些事情，你实现目标的可能性就会最大化。有时是招聘，有时是产品，有时是工程，有时是销售。我变得更加自省，去思考什么才是重要的。我变得更愿意去做那些以前我会说"不是我最喜欢做"的事情，因为我从产生影响力中获得了太多快乐，以至于我现在喜欢的事情变多了。我真的非常感激 Sheryl。现在当我给别人反馈时，我经常想到这一点——那些能改变职业轨迹的瞬间。我把这一切都归功于她。

[00:19:53] Lenny Rachitsky

English:

There's so many people that share stories of Sheryl Sandberg giving them advice and that changing their life. What a mensch.

中文翻译:

有很多人都分享过 Sheryl Sandberg 给他们建议并改变他们生活的故事。她真是个了不起的人（mensch）。

[00:20:01] Bret Taylor

English:

Yeah.

中文翻译:

是的。

[00:20:01] Lenny Rachitsky

English:

My biggest takeaway from this, which is this question of what is the most impactful thing I could do today? Such a powerful heuristic just to keep in mind. To your point, you may realize you don't want to be doing sales or hiring, but if that's the most impactful thing and you end up doing it, you may realize I like this and I'm good at this and have thought about-

中文翻译:

我最大的收获是这个问题："今天我能做的最有影响力的事情是什么？"这是一个非常强大的启发式思考。正如你所说，你可能意识到自己不想做销售或招聘，但如果那是最有影响力的事情，当你去做了，你可能会发现自己其实喜欢并擅长这些，而且思考过——

[00:20:18] Bret Taylor

English:

Can I double click on that though for a second?

中文翻译:

我能就这一点再深入探讨一下吗?

[00:20:19] Lenny Rachitsky

English:

Absolutely.

中文翻译:

当然可以。

[00:20:20] Bret Taylor

English:

I think it's really hard. One of the dangers for founders and product managers, but I think particularly for founders is incorrect storytelling. People don't like my product because of X. And if you tell that to yourself and you tell it to your team, all of a sudden, it goes from being an intuition to being a fact. Well, you better hope you're right because if you orient your strategy around fixing a problem and you're wrong, your company's going to fail. So why did you lose a deal? You could talk to the salesperson who is on the account or perhaps maybe a product manager was involved in the conversation. It's very important to have intellectual honesty in those moments because you could say something like, "Oh, they didn't buy it because the platform cost too much." And that's something a salesperson might say.

(00:21:16):

Maybe the real reason is they didn't actually see much value in your platform. So it was communicated to the salesperson as it was too expensive. But in fact, the problem was product differentiation. And you could end up going into a discussion on pricing when in fact, there was a much deeper, much harder problem to solve there. But just like when you break up with someone, you don't say, it's because I don't like you anymore. You say it's not you, it's me. You say all these pleasantries because we're all social animals and you want to be pleasant with the people around you. So literally taking what a customer says or what a user says in a focus group or a usability study is rarely correct. It often is related to what the truth is, but it's very important to get right. And so, I think one of the things I've observed with first time founders in particular is you're often a single issue voter based on your skillset.

(00:22:14):

So if you're a great engineer, the answer to almost every problem in your business is engineering. If you're a product designer, the answer almost to the proverbial redesign, I joke, it's like the dead cap balance of a consumer product like this next redesign will fix all of our problems. I don't know if it's ever, ever worked. And then I met a lot of entrepreneurs who come from a business development background, they're always thinking about partnerships and oh, if we just get this partnership done for this distribution channel, everything's going to change. And I think it's really important when you're a founder to be self-aware that you will naturally, subconsciously pick the thing that is your strength, your superpower as a solution to more problems. And in fact, if you think that's a solution to your problem, it may be right, but you probably by default should question it.

(00:23:03):

If you think the thing that you've been doing your whole career is the way to fix your problem, it's at least 30% likely that you've chosen that because of comfort and familiarity not truth. And so, I think one of the skills I think is, it really goes around to do you have a good co-founder? Do you have a good leadership team? If you're a product manager, you're a partner in engineering, you're a partner in marketing, you really want to have very real conversations to ensure that you're actually working on the actual correct thing. And I think it's easy to say what's the most impactful thing to do today? My guess, if a lot of people try that, they'll lie to themselves more often than not. And it's a very challenging question to answer. The question's interesting. Being able to answer it accurately is actually the hard part.

中文翻译:

我觉得这其实很难。对于创始人和产品经理，尤其是创始人来说，一个危险在于“错误的叙事”。比如，“人们不喜欢我的产品是因为 X”。如果你对自己和团队这么说，突然之间，这就会从一种直觉变成一个“事实”。你最好祈祷你是对的，因为如果你围绕修复一个错误的问题来制定策略，你的公司就会失败。

比如，你为什么丢掉了一个订单？你可能会问负责该账户的销售人员，或者参与谈话的产品经理。在这些时刻，保持“理性的诚实”（intellectual honesty）非常重要。你可能会说：“哦，他们没买是因为平台太贵了。”这是销售人员常说的话。但真实原因可能是他们根本没看到你平台的价值。所以他们对销售说太贵了，但实际上问题在于产品缺乏差异化。你可能会陷入关于定价的讨论，而实际上有一个更深层、更难解决的问题。

就像分手时，你不会说“因为我不喜欢你了”，你会说“不是你的问题，是我的问题”。你会说这些客套话，因为我们都是社交动物，想对周围的人表现得友善。所以，字面上听取客户在焦点小组或可用性研究中说的话很少是完全正确的。它通常与真相有关，但准确把握真相至关重要。

我观察到，尤其是初次创业者，往往会根据自己的技能背景成为“单议题投票者”。如果你是个优秀的工程师，你业务中几乎所有问题的答案都是工程；如果你是产品设计师，答案几乎总是“重新设计”——我开玩笑说，这就像消费产品的“死猫反弹”，觉得下一次重新设计就能解决所有问题，但我不知道这是否真的奏效过。我还见过很多商务拓展（BD）背景的企业家，他们总是在想合作伙伴关系，“哦，只要我们搞定这个渠道的合作，一切都会改变”。

作为创始人，意识到这一点非常重要：你会自然而然地、潜意识地选择你的强项、你的超能力作为解决更多问题的方案。事实上，如果你认为那是解决方案，它可能是对的，但你默认应该质疑它。如果你认为你职业生涯一直在做的事情就是解决当前问题的方法，那么至少有 30% 的可能性是因为舒适和熟悉，而不是因为真相。

所以，我认为一项关键技能是：你是否有好的联合创始人？是否有好的领导团队？如果你是产品经理，你和工程、营销部门是伙伴，你需要进行非常真实的对话，以确保你们处理的是真正正确的事情。说“今天最有影响力的事情是什么”很容易，但我猜如果很多人尝试这样做，他们大多时候会欺骗自己。这是一个非常具有挑战性的问题。问题本身很有趣，但能准确回答它才是最难的部分。

[00:23:50] Lenny Rachitsky

English:

This feels like such an important lesson you've learned. Is there an example that comes to mind where you learned this the hard way where you actually ended up-

中文翻译:

这听起来是你学到的非常重要的一课。有没有什么例子让你记忆犹新，是你通过惨痛教训学到的，结果你最终——

[00:23:57] Bret Taylor

English:

Oh, yeah. You just want to spend this whole thing on my failures, but I'm fine with that.

中文翻译:

噢，是的。你就是想把整期节目都花在我的失败上，不过我没意见。

[00:24:02] Lenny Rachitsky

English:

You've had too much success.

中文翻译:

因为你太成功了。

[00:24:04] Bret Taylor

English:

Frontier was my first company. At our peak we had 12 employees, 12 of the best people I've ever worked with. Started the company with Jim Norris, who's an engineer I've known since Stanford and Paul Buhite and Sanjeev Singh who Paul started Gmail. Sanjeev was the first engineer at Gmail, so we had the Google Maps people and Gmail people. It was a pretty awesome founding team. We made a social network, as you said. We invented a lot of concepts that became popular in the newsfeed. We invented the like button. It was really neat. It was a fun time. We were only really popular in Turkey, Italy and Iran, and at one point, we were blocked in Iran, so we were only popular in Turkey and Italy and Silicon Valley. To this day, actually a lot of folks in Silicon Valley are like, "I love FriendFeed." I'm like, that's awesome.

(00:24:51):

It wasn't really a successful business. We were a follower-oriented social network, not a friendship-oriented social network, which meant a lot of our content was more like X or Twitter than it is Facebook in that respect. And a lot of sharing newspaper articles, interests, scientific communities, things like that. And there was a period when Twitter, which was one of our competitors at the time, that there was a lot more social networks at the time. I am probably screwing this up a little bit. I think Obama, Ashton Kutcher and Oprah Winfrey all went on Twitter in a summer and we just got our ass kicked.

(00:25:31):

And it was a great example of you... I think 11 of those 12 people were engineers and we were just making product and I think it was Biz Stone. If you talk to the Twitter folks, they could give you the history on this, but I think Biz was really focused on getting celebrities and public figures onto Twitter, which is totally obvious. If you have a social service that's oriented towards following people, put some people on there worth following and instead, we were exclusively focused on polishing the product.

(00:26:00):

And we actually, I think at our peak of popularity, we were very confident just, I think it was a time when Twitter had the fail whale and it was down half the time and people couldn't even use it. And our product, we were innovating faster, we had more features, people liked it and we were up 100% of the time and we totally lost for no reason related to product at all. And it was an example of, I think somewhat famously not a lot of great entrepreneurs have come out of Google because Google was so successful, I think it's hard as a product manager to see distribution and product design and even business model when you have AdWords and money's raining from the sky. There wasn't as much scrutiny and I think folks like the

PayPal Mafia I think learned a lot more about entrepreneurialism than a typical PM at Google. So, we're just getting punched in the face and learning this the hard way.

(00:26:57):

And so, that was probably the most prominent example of it and I think we probably did have a... I can tell you all the flaws of that product, but I don't think that was the reason why we lost. There's a lot of reasons. I think there was a lot of flaws of the product, but it was a lot of other stuff. And so, I've learned, accumulated these skills over time, but I say the hard part of that question is answering it correctly, is it's hard when you don't have experience in something, than to have intuition in it. So I think if there's probably a structural flaw, it wasn't that... I don't know if I could have figured out how to reach out to Ashton Kutcher [inaudible 00:27:28], it's not he's on my Rolodex. But I probably wasn't soliciting advice from the right people.

(00:27:36):

I think that what's great about the technology industry is there's a lot of advice. Choosing whom you listen to is actually quite difficult, but I think we're somewhat myopic. We're in our own little world, creating this product and we weren't asking people from the outside in to say, what are you seeing that could go wrong? What are you seeing that could go right? What are you seeing in the industry that we're not doing that you think we might want to do? And this is why boards are important. This is why finding the right advisors, the advisors will actually tell you what you not [inaudible 00:28:09] want to hear, but you need to hear. I think that was probably the missing part. I'm not sure I was great at market at the time, but if I had solicited the right advice, I could have learned that that was a shortcoming. And I think that was a deep lesson I took from that. I'm a huge believer in boards and getting good advice.

中文翻译:

FriendFeed 是我的第一家公司（注：应为 FriendFeed，Bret 口误说成 Frontier）。鼎盛时期我们有 12 名员工，是我合作过最优秀的 12 个人。我和 Jim Norris（斯坦福校友）、Paul Buchheit（Gmail 创始人）以及 Sanjeev Singh（Gmail 第一位工程师）一起创办了它。所以我们有 Google 地图的人，也有 Gmail 的人，创始团队非常强大。正如你所说，我们做了一个社交网络，发明了很多后来在新闻流中流行的概念，比如点赞按钮。那是一段非常有趣的时光。我们当时只在土耳其、意大利和伊朗非常流行，后来在伊朗被封锁了，所以只在土耳其、意大利和硅谷流行。直到今天，硅谷还有很多人跟我说：“我爱 FriendFeed。”我说：“那太棒了。”

(00:24:51):

但它并不是一个成功的商业项目。我们是一个以“关注”为导向的社交网络，而不是以“友谊”为导向的，这意味着我们的内容更像现在的 X（Twitter）而不是 Facebook。有很多报纸文章分享、兴趣小组、科学社区等等。当时 Twitter 是我们的竞争对手之一。我可能记不太清了，但我记得奥巴马、阿什顿・库彻和奥普拉都在一个夏天加入了 Twitter，然后我们就彻底被打败了。

(00:25:31):

这是一个极好的例子。我们 12 个人里有 11 个是工程师，我们只顾着做产品。而 Twitter 的 Biz Stone 当时非常专注于邀请名人和公众人物加入 Twitter，这其实非常显而易见：如果你有一个以关注为导向的社交服务，那就得放一些值得关注的人在上面。相反，我们却全身心地投入在打磨产品上。

(00:26:00):

在人气最旺的时候，我们非常自信。当时 Twitter 经常出现“宕机鲸鱼”（fail whale），有一半时间无法使用。而我们的产品创新更快，功能更多，用户更喜欢，而且 100% 在线。但我们彻底输了，原因跟产品一点关系都没有。这是一个例子——众所周知，Google 并没有走出很多伟大的创业者，因为 Google 太成功了。当你有 AdWords 这种像天上掉钱一样的业务时，作为产品经理很难看清分发、产品设计甚至商业模式。当时没有那么

多审视，我觉得像“PayPal 黑帮”那样的人比 Google 的典型 PM 学到了更多关于创业的知识。所以，我们当时就像被人迎面痛击，通过惨痛教训学习。

(00:26:57):

这可能是最显著的例子。我可以告诉你那个产品的所有缺陷，但我认为那不是我们失败的原因。原因有很多，产品缺陷只是其中之一，更多的是其他方面。我随着时间的推移积累了这些技能，但我说那个问题的难点在于“准确回答”：当你对某件事缺乏经验时，很难产生直觉。所以，如果说有什么结构性缺陷，倒不是说我能搞定阿什顿・库彻（他不在我的联系人名单里），而是我没有向正确的人寻求建议。

(00:27:36):

科技行业的优点是建议很多，但选择听谁的其实很难。我们当时有点短视，沉浸在自己的小世界里做产品，没有请外部的人来看看：你们觉得哪里可能出问题？哪里可能做对？行业里有哪些我们在做而你们觉得我们应该考虑的事情？这就是为什么董事会很重要，为什么寻找正确的顾问很重要——顾问会告诉你那些你不想听但必须听的话。我觉得那是当时缺失的部分。我不确定我当时是否擅长市场，但如果我寻求了正确的建议，我本可以意识到那是我的短板。这是我学到的深刻教训。我坚信董事会和获取好建议的价值。

[00:28:26] Lenny Rachitsky

English:

Any heuristics or advice for people to know whose advice to listen to? What do you pay attention to when you're like, okay, ignore this person but listen to this person?

中文翻译:

对于如何判断该听谁的建议，你有什么启发式方法或建议吗？当你决定“忽略这个人，听那个人的”时，你会关注什么？

[00:28:35] Bret Taylor

English:

Yeah, that one's tough. It does come down to good judgment and being judge of people's character. One thing that is particularly hard is there's not a strong correlation between the confidence with which someone expresses an opinion and the quality of that opinion. I don't want to say it's inversely correlated, but that's funny, with all the podcasts out now, if there's topics I know a lot about, sometimes the most eloquent, confident statements about things I know a lot about, are the least accurate and it sounds extremely persuasive. And so, it does require very good judgment. One thing is I think, not just asking for advice but asking people, who should I talk to get good advice? And you'll find some common answers there and that's often a really strong signal of good judgment.

(00:29:26):

And then one thing I found is when you ask for advice, don't just ask what to do but why. Be an obnoxious two-year-old kid, why? Why? Why? Why? And really try to understand the framework that someone is using to give you advice. The interesting thing about advice is people are often extrapolating from relatively few experiences. So they will say, never do this or always do that. And it's because they had one experience where something backfired or something could have gone better if they had done it. So it's a useful anecdote, but if you don't ask why and understand they had one experience and here's what happened, it can come across as a rule when in fact, it's [inaudible 00:30:08] data and if you ask advice of three people and they all have very similar interactions, you can create a first principles framework from which that advice emerges. And when you start applying it, you're applying it with a degree of nuance that you couldn't if you're just following a rule. So, I think one is, it does come down to good judgment, I think. I don't know how to teach that.

(00:30:30):

I'm a huge believer in good judgment. It's one of the things I hire for. I just think that's something that probably comes from a mix of self perfection. You really need to hold yourselves accountable as an entrepreneur, as a product manager. If you made a bad decision, spend time reflecting on it, number one. And really, try to understand why and try to always improve your judgment. I think at the end of the day, that is why you are a good entrepreneur, a good product manager. And number two, when you get advice, really understand where it's coming from and why so that you can create your own independent view of where that advice came from and recognize that no one's advice is statistically significant or very rarely is it. If you're getting advice on investing from Warren Buffett, yeah, okay, it's statistically significant, but most advice is like something happened to you once and you have regrets.

中文翻译:

是的，这很难。这确实归结为良好的判断力和对人品的判断。特别难的一点是：一个人表达观点的自信程度与该观点的质量之间并没有很强的相关性。我不想说它们是负相关的，但很有趣的是，现在有这么多播客，对于一些我非常熟悉的领域，有时那些最雄辩、最自信的言论反而是最不准确的，尽管听起来极具说服力。

所以这确实需要非常好的判断力。第一点，我认为不仅要寻求建议，还要问：“我应该找谁去获得好的建议？”你会发现一些共同的答案，这通常是良好判断力的强信号。

第二点，我发现当你寻求建议时，不要只问“该做什么”，要问“为什么”。像个讨人厌的两岁小孩一样问：为什么？为什么？为什么？真正去理解对方给出建议的思维框架。关于建议，有趣的一点是人们往往是从极少数的经验中推断出来的。他们会说“永远不要做这个”或“总是要做那个”，这是因为他们有过一次搞砸了的经历，或者如果做了某事结果会更好的经历。这是一个有用的轶事，但如果你不问“为什么”，不理解他们只是经历过那一次，它听起来就像一条铁律，而实际上它只是一个数据点。

如果你向三个人请教，他们都有类似的互动经历，你就可以建立一个产生这些建议的“第一性原理”框架。当你开始应用它时，你会带着一种细微的差别去应用，这是你仅仅遵循一条规则所无法做到的。所以，我认为这最终归结为良好的判断力，我不知道该如何教这个。

我坚信良好判断力的价值，这是我招聘时看重的一点。我认为这来自于自我完善。作为企业家或产品经理，你真的需要对自己负责。第一，如果你做了一个错误的决定，花时间反思它，真正理解为什么，并不断提高你的判断力。我认为归根结底，这就是你成为优秀企业家或产品经理的原因。第二，当你得到建议时，真正理解它的来源和原因，这样你就能对建议的来源建立独立的看法，并意识到没有人的建议具有统计学意义（或者极少有）。如果你从沃伦・巴菲特那里得到投资建议，那确实具有统计学意义，但大多数建议只是“某件事曾发生在我身上，我感到后悔”而已。

[00:31:28] Lenny Rachitsky

English:

I love that you're like, I don't know if I have a great answer then you just give us an incredible answer to this question. I want to go in a different direction. You mentioned that you describe yourself as an engineer. I know I heard you code to relax still. Let me just ask you this question, something a lot of people in college are thinking about. Do you think it still makes sense to learn to code? Do you think this will significantly change in the next few years?

中文翻译:

我喜欢你先说“我不知道是否有好答案”，然后给出了一个精彩的回答。我想换个方向。你提到你把自己描述为一名工程师，我听说你现在还会通过写代码来放松。我想问你一个很多大学生都在思考的问题：你认为现在

学习编程还有意义吗？你认为未来几年这会发生重大变化吗？

[00:31:47] Bret Taylor

English:

I do still think studying computer science is a different answer than learning to code, but I would say I still think it's extremely valuable to study computer science. I say that because I think computer science is more than coding. If you understand things like Big O notation or complexity theory or study algorithms and why a randomized algorithm works and why two algorithms with the same Big O complexity, one can then practice perform better than others and why a cache miss matters and just all these little... There's a lot more to coding than writing the code. The reason I think that is I do think the act of creating software is going to transform from typing into a terminal or typing into Visual Studio code to operating a code-generating machine. I think that is the future of creating software. But I think operating a code-generating machine requires systems thinking and I think that computer science, there are other disciplines as well, but computer science is a wonderful major to learn systems thinking and at the end of the day, AI will facilitate creating this software.

(00:33:08):

We may do a lot more in the next few years we can't even imagine, but your job as the operator of that code-generating machine is to make a product or to solve a problem and you really need to have great systems thinking and you're going to be managing this machine that's doing a lot of the tedious work of making the button or connecting to the network. But as you're thinking of the intersection of a technology and a business problem, you're trying to affect a system that will solve that problem at scale for your customers and that systems thinking is always the hardest part of creating products.

(00:33:40):

I'll just give you, it's this cheesy simple example, but I think it's representative. At Facebook, we spent a lot of time designing the newsfeed and if you ever had a really, really good designer and they showed you at the time, a Photoshop mock-up of the newsfeed, it was just all as beautiful. The photos, the family was happy and the photo was a perfect photo and the posts were all perfectly grammatically correct and of a completely normal length and the comments and there was the like... Everything was just perfect. And then you'd implement that design and you'd look at your own newsfeed and it looked like shit because it turns out not everyone's photos were made by a professional photographer. The posts were all these different lengths. The comments were like, you suck and... All that stuff.

(00:34:29):

And then all of a sudden you realize that designing a newsfeed, Photoshop is the easy part. You need to actually design a system that produces both in content and visual design, like a delightful experience given input you don't control. And that's a system, it's sort of a design and it's just, what we did practically, I am sure it's changed a lot since I left in 2012, but we made a system so designers had to show their newsfeed designs with real newsfeed data that was messy rather than anything artificial because I think it forced the process to be more realistic.

(00:35:10):

But I say that because I think that whether AI is writing code or doing the design or doing all these other things, you need to learn how to have a system in your head. You need to understand the basics of what's hard and what's easy and what's possible and what's impossible. And AI can help you do that too, by the way. But I do think that's a really useful skill. I think in general, with the advent of AI agents and AI approaching super intelligence in certain domains, I think the tools with which we do our job will change a lot. I think it's very important to have a very loose attachment to the way we do our jobs and that story that we won't talk about when I rewrote Google Maps, everyone talks about that story and I think it's because of Paul [inaudible 00:35:57] who told it on some podcasts and that's where it made the rounds.

(00:36:01):

I think that's going to end up this vestige of the past, almost like the human calculators at NASA before the computers were invented, like wow, a person was a calculator? Whoa, that's fun. Tell me that story. I think just what I was good at will no longer be useful in the future or certainly not valuable in the future and that's okay. So I think we need to have a really loose view of it, but the idea that you shouldn't study these disciplines, it's like people say, I don't want to study math because I'm not going to use it in my career for X. Well, studying maths is quite important. It teaches you how to think. It teaches you how the world works, physics, math, and I think computer science especially, at least the foundations of it, will continue to be the foundations of how we build software and understanding that when you're interacting particularly with something that's smarter than you, producing code you might not completely understand how you constrain it and how you get it to produce these outcomes. I think it will require a lot of sophistication actually.

中文翻译:

我认为"学习计算机科学"和"学习编程"是两个不同的答案，但我认为学习计算机科学仍然极具价值。我这么说是因为计算机科学不仅仅是编程。如果你理解大 O 表示法、复杂度理论，或者研究算法以及为什么随机化算法有效，为什么两个大 O 复杂度相同的算法在实践中表现不同，为什么缓存缺失（cache miss）很重要……编程中有很多东西比写代码本身更重要。

我之所以这么想，是因为我认为创建软件的行为将从在终端或 Visual Studio Code 中打字，转变为"操作一台代码生成机器"。我认为这是创建软件的未来。但操作代码生成机器需要"系统思维"（systems thinking）。计算机科学（当然还有其他学科）是学习系统思维的绝佳专业。归根结底，AI 会辅助创建软件。

(00:33:08):

未来几年我们可能会做很多现在无法想象的事情，但你作为代码生成机器的操作员，你的工作是做出产品或解决问题。你真的需要具备出色的系统思维。你会管理这台机器，让它去做那些枯燥的工作，比如做一个按钮或连接网络。但当你思考技术与业务问题的交集时，你是在试图影响一个系统，为你的客户大规模地解决问题。而这种系统思维永远是创建产品中最难的部分。

(00:33:40):

我给你举个简单甚至有点俗气的例子，但它很有代表性。在 Facebook，我们花了很多时间设计新闻流（Newsfeed）。如果你找一个非常优秀的设计师，他们会给你展示一个 Photoshop 做的原型，那简直太美了：照片里的家庭很幸福，构图完美，帖子语法正确且长度适中，评论也很和谐……一切都完美无缺。但当你实现那个设计并看自己的新闻流时，它看起来像垃圾一样。因为事实证明，不是每个人的照片都是专业摄影师拍的，帖子的长度千奇百怪，评论里可能还有人在骂街。

(00:34:29):

然后你突然意识到，设计新闻流时，Photoshop 是最简单的部分。你实际上需要设计一个系统，在输入内容不可控的情况下，依然能产生内容和视觉设计上都令人愉悦的体验。这就是一个系统。我们当时的实际做法是（我确信自 2012 年我离开后已经变了很多）：我们建立了一个系统，要求设计师必须用真实的、混乱的新闻流数据来展示他们的设计，而不是用任何人工数据，因为这迫使过程变得更真实。

(00:35:10):

我之所以说这些，是因为无论 AI 是在写代码、做设计还是做其他事情，你都需要学会在脑子里构建一个系统。你需要理解什么是难的，什么是简单的，什么是可能的，什么是不可能的。顺便说一下，AI 也可以帮你做到这

些。但我认为这确实是一项非常有用的技能。总的来说，随着 AI 智能体的出现以及 AI 在某些领域接近超人工智能，我们工作的工具会发生巨大变化。我认为对我们工作方式保持一种“松散的依恋”非常重要。

(00:36:01):

那个关于我在周末重写 Google 地图的故事（大家都爱聊这个，可能是因为 Paul Buchheit 在播客里讲过），我认为那最终会成为过去的遗迹，就像 NASA 在发明计算机之前的“人类计算员”一样。人们会说：“哇，以前居然有人充当计算器？真有意思，讲讲那个故事。”我以前擅长的东西在未来可能不再有用，或者不再有价值，这没关系。所以我们需要保持开放的心态。但那种“不该学习这些学科”的想法，就像有人说“我不想学数学，因为我以后的职业用不到”一样。学习数学很重要，它教你如何思考，教你世界如何运作。我认为计算机科学，尤其是它的基础，将继续成为我们构建软件的基石。当你与比你更聪明的东西互动，让它生成你可能不完全理解的代码时，如何约束它、如何让它产生预期的结果，实际上需要非常高的素养。

[00:36:59] Lenny Rachitsky

English:

That's such a great answer. There's this always sense of this binary, should I learn to code or not? And your point here is learn to understand how engineering works and how systems work and what your code does and how it all interconnects, but the way you actually do the coding at your desk will change significantly. This reminds me of something you mentioned on a podcast recently. This idea that you think there's going to, or there should be a new programming language that is more designed for LLMs versus humans. Can you just talk about that because I think a lot of people aren't thinking about that?

中文翻译:

这个回答太棒了。人们总是在纠结“该不该学编程”这种二元对立的问题。而你的观点是：去学习工程如何运作、系统如何运作、代码的作用以及它们如何互联，但你实际坐在桌前写代码的方式会发生巨大变化。这让我想起你最近在播客中提到的一个观点：你认为将会出现、或者应该出现一种更多是为大语言模型（LLM）而非人类设计的编程语言。你能聊聊这个吗？我觉得很多人还没想到这一点。

[00:37:27] Bret Taylor

English:

I don't know if it's a language, I would call it a programming system because I think language might be too limited. My reductive version of the past, what 40 years of computers maybe more, is we created the hardware for computers, then we created punch cards, which is the way in the late '70s you would tell a computer what to do or maybe mid to late '70s. Then we invented early operating systems and time-sharing systems from the invention of things like Unix at Bell Labs and Berkeley, you ended up with the C programming language, Fortran and a lot of higher level programming languages. I think Fortran and then C.

(00:38:15):

And we moved up the layers of abstraction. No one does punch cards anymore, obviously. Few people write assembly language. Some people write C, some people write REST. But a lot of people write Python and TypeScript and things like that. And as we've invented more and more abstractions, we've made it easier to do high-leverage things. So, if you look at how remarkable Google was back in the day or Google Maps, you could probably give a lot of react programmers the task of make a draggable map now and I think a lot of people could do it. That was true RND back in the day.

(00:38:54):

When Salesforce was created in 1998, just putting a database in the cloud was hard and that alone was a technical moat that is now trivial with Amazon Web Services and that technical moat is comically narrow, but the product moat is quite large. I think that if the act of writing code is going from something that is very costly to the marginal cost of that going to zero, how many of the abstractions that we've built are based on human program or productivity? I think a ton. I always laugh that I assume Python is probably the most common generated code just because of how much it's in the training data and data scientists love Python and I love Python too. It's such a comically bad thing for AI to generate just because it's one of the most inefficient programming languages of all time. If you know the global interpreter lock and just slow. And I've written a lot of high-scale web services and it's just quite slow and it's very hard to verify.

(00:40:00):

It's not as bad as Perl, but if you have a big Python program, how many errors will you find at runtime versus before releasing it? So, Python was designed to be very ergonomic, almost looked like pseudo code for humans, for me to write code in a delightful way. That's why data scientists love it so much. So as we move to a world where let's just postulate, and I'm not sure this will be completely true, that we're not going to write a lot of code as people. We're going to be operating these code-generating machines. We probably don't care how ergonomic the programming language is. What we care about is when this machine generates code, do we know that it did what we wanted it to do? And if it doesn't do we want it to do, can we change it easily? I think there's a lot of insights in programming languages that could serve this.

(00:40:48):

So Rust I think, is interesting because if I asked you to look at a C program and say, "Does it leak memory?" You probably couldn't do it that well just because it's really hard and if it's a very, a million line C program, it's going to be very, very hard. If I asked you to verify that a Rust program doesn't leak memory, you would just have to compile it. And because it has compile time, memory safety, just the act of compiling successfully tells you that's true. I think we need more things like that because if an AI is generating this code, by definition, if you have to read every line that is going to be the limiting factor for producing the code or worse, you're just not going to read every line and you're going to emit a bunch of unsafe unverified code into the wild.

(00:41:36):

And so, the question is how do you enable humans to have as much leverage as possible? Which means using computers to do the work on your behalf. You could have obviously the simplest form of this, is AI supervising AI and doing code reviews and that's great. Certainly, self-reflection is a really effective way of improving the robustness of an AI system. But I do think if it doesn't matter how tedious it is to write the code, you could probably layer on some techniques that are out of fashion, like formal verification, unit testing, other things. And if you layer all these on, I'm thinking about it as, I as a... It's like the guy in the matrix with the green letters coming down, how can I make something so I as a operator of the code generating machine can produce incredibly complex scale software, incredibly quickly and know that it works?

(00:42:26):

If you start with that as your design center, I think you'd probably changed the languages, you'd probably changed the systems, you'd probably change all these things and you're probably going to bring to bear a lot of things. And what's really fun about it is you can loosen a lot of constraints, like coding is free. Okay, so that's neat. With that in mind, what do you want to do? What would be best suited for the language, the compiler for testing, for self-reflection, for supervisor models, all these things. I think that's more of a programming system than a language, but I think when we create something like that, it can really enable creators, builders to create incredibly robust, incredibly complex systems.

(00:43:04):

And I'm super excited about VibeCoding, but I don't know generating a prototype has been the limiting factor in software ever. It's actually building increasingly complex systems and actually changing them with agility. If you look at the famous Netscape one to Netscape two rewrite, somewhat, a lot of people attribute that to part of their failure against Internet Explorer. It's like making these things is not hard. Maintaining them is hard and ensuring they're robust is hard. And I think we're in the very early phases of defining what this new system for developing software looks like and I'm very excited to see what emerges.

中文翻译:

我不知道它是否是一种语言，我更愿意称之为“编程系统”，因为“语言”可能太局限了。我对过去 40 多年计算机历史的简化理解是：我们先创造了硬件，然后创造了打孔卡（70 年代末告诉计算机该做什么的方式）。接着我们发明了早期的操作系统和分时系统，比如贝尔实验室和伯克利的 Unix，随后出现了 C 语言、Fortran 以及许多高级编程语言。

(00:38:15):

我们不断提升抽象层级。显然现在没人再用打孔卡了，很少有人写汇编，有些人写 C，有些人写 Rust，但更多的人在写 Python 和 TypeScript 等。随着抽象层级的增加，我们更容易做高杠杆的事情。比如，当年 Google 地图是真正的研发难题，但现在你让一个 React 程序员做一个可拖拽的地图，很多人都能做到。

(00:38:54):

1998 年 Salesforce 创立时，仅仅把数据库放在云端就很困难，那是当时的技术护城河。现在有了 AWS，这种技术护城河窄得可笑，但产品护城河却很大。如果写代码的行为从昂贵变得边际成本趋于零，那么我们构建的抽象层中有多少是基于“人类程序员的生产力”的？我认为是绝大部分。我经常笑称，Python 可能是 AI 生成最多的代码，因为训练数据里太多了，数据科学家也爱它。我也爱 Python，但对 AI 来说，生成 Python 简直是件糟糕透顶的事，因为它是史上效率最低的编程语言之一（比如全局解释器锁 GIL，速度很慢）。我写过很多大规模 Web 服务，Python 很难验证。

(00:40:00):

它不像 Perl 那么糟，但对于一个大型 Python 程序，你在运行时会发现多少错误？Python 的设计初衷是符合人体工程学，看起来像伪代码，让人写起来很愉悦。但在一个我们不再亲手写代码，而是操作代码生成机器的世界里，我们可能不再关心语言是否符合人体工程学。我们关心的是：当机器生成代码时，我们能否知道它是否按我们的意愿运行？如果不是，我们能否轻松修改？

(00:40:48):

我认为 Rust 很有趣。如果我让你看一个 C 程序并问“它是否内存泄漏？”，你很难做到，尤其是百万行级别的。但如果我让你验证 Rust 程序是否内存泄漏，你只需要编译它。编译成功就意味着内存安全。我们需要更多这样的东西。如果 AI 生成代码，而你必须逐行阅读，那阅读就会成为瓶颈；或者更糟，你根本不读，直接发布一堆未经核实的、不安全的代码。

(00:41:36):

所以问题是：如何让人类拥有尽可能大的杠杆？这意味着让计算机代劳。最简单的形式是 AI 监督 AI 进行代码审查，这很棒。自我反思是提高 AI 系统鲁棒性的有效方式。但如果写代码不再费力，我们可以叠加一些已经过时的技术，比如形式化验证（formal verification）、单元测试等。我把它想象成《黑客帝国》里那个看着绿色字符流下的家伙：我作为操作员，如何能极其快速地生产极其复杂的大规模软件，并确信它是有效的？

(00:42:26):

如果你以此为设计中心，你可能会改变语言、改变系统。最有趣的是你可以放开很多约束，比如“编码是免费的”。在这种前提下，什么样的语言、编译器、测试、自我反思、监督模型是最合适的？我认为这更像是一个

编程系统。当我们创造出这样的东西时，它能让构建者创造出极其鲁棒、极其复杂的系统。

(00:43:04):

我对“氛围编程”（VibeCoding）感到兴奋，但我认为生成原型从来不是软件开发的瓶颈。真正的挑战是构建日益复杂的系统并敏捷地修改它们。就像当年 Netscape 从 1.0 到 2.0 的重写，很多人认为那是他们输给 IE 的原因之一。做出东西不难，维护它们并确保鲁棒性才难。我认为我们正处于定义这种新软件开发系统的早期阶段，我很期待看到会产生什么。

[00:43:42] Lenny Rachitsky

English:

I feel like we're definitely living in the future when someone like you is suggesting that we build a matrix like experience and that's going to be potentially the future of coding and building. I can't wait for that. It feels like a great opportunity and a fun project.

(00:43:57):

This episode is brought to you by Vanta and I'm very excited to have Christina Cacioppo CEO and co-founder Vanta joining me for this very short conversation.

中文翻译:

我觉得我们绝对生活在未来，像你这样的人在建议我们构建一种类似《黑客帝国》的体验，而这可能是未来编程和构建的方式。我迫不及待想看到了。这听起来是一个巨大的机会，也是一个有趣的项目。

(00:43:57):

本期节目由 Vanta 赞助。我非常高兴邀请到 Vanta 的 CEO 兼联合创始人 Christina Cacioppo 加入这段简短的对话。

[00:44:06] Christina Cacioppo (Vanta)

English:

Great to be here. Big fan of the podcast and the newsletter.

中文翻译:

很高兴来到这里。我是播客和时事通讯的忠实粉丝。

[00:44:09] Lenny Rachitsky

English:

Vanta is a long time sponsor of the show, but for some of our newer listeners, what does Vanta do and who is it for?

中文翻译:

Vanta 是我们节目的长期赞助商，但对于一些新听众，Vanta 是做什么的，受众是谁?

[00:44:16] Christina Cacioppo

English:

Sure. So we started Vanta in 2018 focused on founders helping them start to build out their security programs and get credit for all of that hard security work with compliance certifications like SOC 2 or ISO 2701. Today, we currently help over 9,000 companies including some start-up household names like Atlassian Ramp and LangChain, start and scale their security programs and ultimately, build trust by automating compliance, centralizing GRC, and accelerating security reviews.

中文翻译:

好的。我们在 2018 年创办了 Vanta，专注于帮助创始人建立安全计划，并通过 SOC 2 或 ISO 27001 等合规认证，让他们辛苦的安全工作得到认可。今天，我们帮助超过 9000 家公司（包括 Atlassian、Ramp 和 LangChain 等知名初创公司）启动并扩展安全计划，最终通过自动化合规、集中化 GRC（治理、风险与合规）和加速安全审查来建立信任。

[00:44:46] Lenny Rachitsky

English:

That is awesome. I know from experience that these things take a lot of time and a lot of resources and nobody wants to spend time doing this.

中文翻译:

太棒了。我从经验中知道，这些事情耗时耗力，没人想把时间花在这上面。

[00:44:54] Christina Cacioppo

English:

That is very much our experience, but before the company and to some extent, during it. But the idea is with automation, with AI, with software, we are helping customers build trust with prospects and customers in an efficient way. And our joke, we started this compliance company so you don't have to.

中文翻译:

这正是我们的体会。我们的理念是利用自动化、AI 和软件，帮助客户高效地与潜在客户建立信任。我们常开玩笑说：我们创办了这家合规公司，这样你就不用自己操心合规了。

[00:45:10] Lenny Rachitsky

English:

We appreciate you for doing that. And you have a special discount for listeners. They can get $1,000 off Vanta at vanta.com/lenny. That's V-A-N-T-A .com/lenny for $1,000 off Vanta. Thanks for that, Christina.

中文翻译:

感谢你们所做的一切。你们还为听众提供了特别优惠：在 vanta.com/lenny 可以获得 1000 美元的优惠。谢谢你，Christina。

[00:45:25] Christina Cacioppo

English:

Thank you.

中文翻译:

谢谢。

[00:45:26] Lenny Rachitsky

English:

Okay. One more question along these lines and then I want to zoom out on just where AI is heading and something I love to ask folks like you that are at the cutting edge of AI is what you're teaching your kids. I know you have kids, I feel like the world is going to be very different when they grow up. What are you encouraging them to learn that you think is different maybe from previous generations to help them be successful in a world of AI abundance?

中文翻译:

好，关于这个话题最后一个问题，然后我想放大视角聊聊 AI 的走向。我喜欢问像你这样处于 AI 前沿的人一个问题：你都在教孩子什么？我知道你有孩子，我觉得当他们长大时，世界会变得非常不同。你鼓励他们学习什么，你认为这与前几代人有什么不同，从而帮助他们在 AI 充裕的世界中取得成功?

[00:45:53] Bret Taylor

English:

I don't know if I'm teaching them differently, but I'm really trying to encourage them to make AI a part of their lives. I was reflecting actually when I took the AP calculus exams in '97, '98 AB and BC, I could use a graphing calculator. And I haven't done this research I was meaning to plug this into ChatGPT before our conversation, but I'll do it after. Did the calculus exam change before and after they allowed the calculator in the exam? I assume it did. But essentially, to when you allow the calculator in the exam, you need to make sure that none of the questions benefit people for having a calculator or not, which actually forces you to rethink the problems to test calculus knowledge that don't benefit from like road arithmetic or the other things you can do on a graphing calculator.

(00:46:47):

I think that a lot of education doesn't presume you have a super intelligence in your pocket. And so, if you ask someone to write an essay on a book that they read, you could probably hallucinate one pretty easily from one of the big providers like ChatGPT. And maybe if you are skilled enough that prompting, maybe even your teacher won't know it's written by an AI. So what do you do? How do you teach kids differently? It's really hard for teachers right now because I think we haven't gone through the transition of adding calculators to the exams. So, I think a lot of the mechanisms we have to evaluate students are broken by the existence of ChatGPT and the like. So I think we're in a very awkward phase, but I think we can still both teach kids how to think and teach kids how to learn. And I think our education system can catch up and I actually think these models can be one of the most effective educational tools in history.

(00:47:46):

I don't know if you're a visual learner or reading learner. I like to read. I didn't love going to lectures. I don't learn that well from them. I like to read the book and if you have a teacher who doesn't teach in your style, you can now go home and ask ChatGPT to teach you in another mechanism. My kids use ChatGPT to quiz them before a test. You can use audio mode or chat mode. It's better than cue cards. My daughter took home a Shakespeare book, she took a picture of a page she didn't understand and ChatGPT explained it to her way better than I would've as well. I think every child in this world has a personalized tutor that can teach them in the way that they best learn, visually, over audio, reading. We have a platform that can test you, that can quiz you. I think it's really an amplifier of agency.

(00:48:39):

I think the kids who have agency, who have aspirations to learn something, I think you have what is the best combination of every teacher you've ever had and these models and you can use it. So with my kids, my oldest daughter learned how to code and she was making a website and every time she had a question for me, I would just make her use ChatGPT. Not because I was trying to be an obnoxious father, but I'm like she needs to learn to use this tool because it's amazing.

(00:49:13):

So, I really am trying to have them learn how to use it constructively in their lives. But all that said, I just feel a ton of empathy for public school teachers right now. It's very hard because the technology is moving faster than our educational system. And I think particularly as it relates to evaluation, it's just really challenging for teachers right now. And I worry because these technologies amplify agency, the opposite can also be true. If you are a student trying to not learn something, I think these tools probably provide a lot of mechanisms to avoid it as well. And so, I think there's a challenge for parents and teachers and I think we're going to end up with a handful of years here.

(00:49:53):

But I brought up the calculus AP exam because obviously, a graphing calculator is not ChatGPT, don't get me wrong. But I think we've been able to figure out a way to conform homework and in-class learning and tests around the technologies available to us fairly successfully to date. And I'm fairly confident we'll figure it out and I think it's going to... And on the much more positive side, I don't know, I went to public schools, I don't know if you did too. You end up with some pretty bad teachers at times and now, you have an outlet. You don't need to be the rich kid who can afford a tutor anymore to get tutoring. If you are a kid who excels in math and your school doesn't have advanced statistics classes, well, now you do.

(00:50:40):

So I think this is just an incredibly democratizing force with kids who have agency and I think that's very exciting. I'm hopeful that there's a 11-year-old right now who's going to start a really amazing company 10 years from now whose ChatGPT is going to be their primary tutor that led to that outcome. And I think that's pretty cool.

中文翻译:

我不确定我教他们的方式是否不同，但我确实在努力鼓励他们让 AI 成为生活的一部分。我回想起 1997、98 年参加 AP 微积分考试时，是允许使用图形计算器的。我还没做过这项研究（本想在对话前问问 ChatGPT，待会儿去问）：微积分考试在允许使用计算器前后有变化吗？我猜是有变化的。当你允许使用计算器时，你需要确保题目测试的是微积分知识，而不是那些能通过计算器轻松完成的死算。

(00:46:47):

我认为现在的很多教育并没有假设你口袋里有一个“超人工智能”。如果你让学生写读后感，他们可以轻松用 ChatGPT 生成一篇。如果提示词（Prompt）写得好，老师甚至看不出是 AI 写的。那该怎么办？如何以不同的方式教孩子？现在的老师很难，因为我们还没经历过像“考试加入计算器”那样的转型。现有的很多学生评估机制都被 ChatGPT 破坏了。我们处于一个尴尬的阶段，但我认为我们仍然可以教孩子如何思考、如何学习。教育系统会跟上的，而且我认为这些模型会成为史上最有效的教育工具之一。

(00:47:46):

无论你是视觉型学习者还是阅读型学习者，AI 都能帮到你。我不喜欢听讲座，我更喜欢读书。如果你遇到一个教学风格不适合你的老师，你可以回家让 ChatGPT 用另一种方式教你。我的孩子用 ChatGPT 在考试前考考自

己，用语音模式或聊天模式，这比抽认卡好用多了。我女儿带回一本莎士比亚的书，她拍下看不懂的一页，ChatGPT 给她的解释比我解释得好得多。我觉得世界上的每个孩子现在都有了一个个性化的导师，可以用他们最擅长的方式（视觉、听觉、阅读）来教导他们。这真的是一种“能动性”（Agency）的放大器。

(00:48:39):

对于那些有能动性、渴望学习的孩子来说，AI 模型是你所有老师优点的集合体。我大女儿学习编程做网站时，每次她问我问题，我都会让她去问 ChatGPT。不是因为我想当个甩手掌柜，而是因为她需要学会使用这个神奇的工具。

(00:49:13):

我努力让他们在生活中建设性地使用它。但话虽如此，我非常同情现在的公立学校老师。技术发展比教育系统快得多，尤其是在学生评估方面，老师们面临巨大挑战。我担心的是，既然技术能放大能动性，反面也成立：如果一个学生不想学习，这些工具也提供了很多逃避学习的手段。所以家长和老师都面临挑战。

(00:49:53):

我提到 AP 微积分考试是因为，虽然图形计算器不是 ChatGPT，但到目前为止，我们已经成功地让作业和考试适应了现有技术。我确信我们会解决这个问题。从积极的一面看，我上的是公立学校，有时会遇到不太好的老师，但现在你有了出路。你不再需要是富家子弟才能请得起家教。如果你数学很好，但学校没有高级统计课，现在你可以自学了。

(00:50:40):

我认为这对于有能动性的孩子来说是一种极其民主化的力量。我希望现在某个 11 岁的孩子，因为把 ChatGPT 当作主要导师，能在 10 年后创办一家了不起的公司。这真的很酷。

[00:51:02] Lenny Rachitsky

English:

I have a two-year-old and it feels like there's a new milestone of, there's like when to give them a phone, when to give them, I don't know, Snapchat, whatever kids use these days and then it's like when to give them their first ChatGPT account. Well no, I wonder how soon that's supposed to happen.

中文翻译:

我有一个两岁的孩子，感觉现在有了新的里程碑：什么时候给他们手机，什么时候给他们 Snapchat（或者现在小孩用的什么东西），然后是——什么时候给他们第一个 ChatGPT 账号。我在想这应该多早发生。

[00:51:16] Bret Taylor

English:

I think my personal take is, it's different than the former two. I don't think mobile phones are great in school or great for kids and I personally advocate for waiting a long time. But I think that ChatGPT is more like Google search and it's one thing to have a device in your pocket that's addictive and has push notifications, but it's another thing to use AI to learn. And so, I think the two are different. And I really think of AI fundamentally as a utility. I don't think a lot of parents before ChatGPT said, "When should I let my kid use Google search?" That's a different type of tool. I think thinking of it like that is the way I think about these technologies.

中文翻译:

我个人的看法是，它与前两者不同。我不认为手机对学校里的孩子有好处，我个人主张等很久再给。但我觉得 ChatGPT 更像 Google 搜索。口袋里有一个让人上瘾、充满推送通知的设备是一回事，而使用 AI 来学习是另一回事。我认为两者性质不同。我从根本上把 AI 看作一种工具（Utility）。在 ChatGPT 出现之前，没多少家长会问“我该什么时候让孩子用 Google 搜索？”，因为那是不同类型的工具。这就是我看待这些技术的方式。

[00:51:55] Lenny Rachitsky

English:

Is the form factor for your kids like an iPad or a laptop or something?

中文翻译:

你孩子使用的设备形态是什么？iPad 还是笔记本电脑之类的？

[00:51:58] Bret Taylor

English:

Yeah. They use the computer on the desk.

中文翻译:

是的，他们用桌上的电脑。

[00:52:01] Lenny Rachitsky

English:

Got it. All right. Good tips. This is good for me to learn all these things as my kid ages. Okay, I'm going to zoom out and let's talk about business strategy AI. One of the biggest questions a lot of founders think about these days is just where should I build? What will foundational model companies not squash and do themselves? Being someone building a very successful AI business and also being on the board of OpenAI, feel like you have a really unique perspective on what is probably a good idea and it's probably not a good idea. How do you think the AI market is going to play out and where do you think founders should focus and also just try to avoid?

中文翻译:

明白了。好建议，随着我孩子长大，这些对我很有参考价值。好，我想放大视角聊聊 AI 的商业策略。现在很多创始人思考的最大问题之一是：我该在哪里构建？基础模型公司不会“碾压”并亲自去做什么？作为一名正在构建成功 AI 业务的人，同时又是 OpenAI 的董事会成员，我觉得你对于什么是好主意、什么不是，有着非常独特的视角。你认为 AI 市场会如何演变？创始人应该关注哪里，又该避开哪里？

[00:52:36] Bret Taylor

English:

I think there's three segments of the AI market that will end up fairly meaningful markets and then I'll end with how I think it's going to play out. So first is the frontier model market or foundation model market. I think this will end up the small handful of hyperscalers and really big labs just like the cloud infrastructure as a service market. And the reason for that is that creating a frontier model is entirely a function of CapEx. And you need a company with huge amounts of CapEx capacity to build one of these models. All of the companies that were startups that tried to do this have already been consolidated or almost all of them, inflection, adapt, character and others. And I think there doesn't appear to be a viable business model for a startup because of the amount of CapEx required and there's just not enough fundraising runway to get to escape velocity. And also, the models deteriorate in value fairly quickly as an asset class.

(00:53:33):

And so, you need just a lot of scale to make a return on the investment for a model that deteriorates in value so quickly. So, I think that's going to end up probably no entrepreneur should build a frontier model. That's my take.

中文翻译:

我认为 AI 市场最终会形成三个非常有意义的细分市场。首先是“前沿模型”（Frontier Model）或“基础模型”市场。我认为这最终会像云基础设施服务（IaaS）市场一样，由少数几家超大规模企业和大型实验室把持。原因在于，创建前沿模型完全取决于资本支出（CapEx）。你需要一家拥有巨额资本支出能力的公司。几乎所有尝试做这件事的初创公司（如 Inflection, Adept, Character 等）都已被整合。我认为初创公司在这方面没有可行的商业模式，因为资金需求太大，融资跑道不足以达到逃逸速度。而且，作为一种资产类别，模型的价值贬值非常快。

(00:53:33):

因此，你需要巨大的规模才能在价值迅速贬值的模型上获得投资回报。所以，我的看法是：创业者不应该去构建前沿模型。

[00:53:47] Lenny Rachitsky

English:

Unless you're Elon.

中文翻译:

除非你是埃隆（马斯克）。

[00:53:51] Bret Taylor

English:

Yeah. Oh, yeah. He's different. And he has the capacity to raise billions in capital and my guess is most of your other listeners don't, and then he is the greatest of all time for reason and he's different. You don't compare yourself to him. The other part of the market is the tooling, and I think there's a lot of folks selling pickaxes in the Gold Rush. This is data labeling, services. This is data platforms, it's eval tools. More specialized models like 11 Labs has a great set of voice models that a lot of companies use that are really high quality and it's like if you're trying to be successful in AI, what are the different tools and services that you need?

(00:54:31):

There is some risk to the tooling market because it's pretty close to the sun. So, if you look at the infrastructure as a service market and the cloud tooling market like the Confluent and Databricks and Snowflake, a lot of the Amazon and Azure and others have competing products in those areas because they're very adjacent to the infrastructure itself and every infrastructure provider is trying to differentiate by moving up the stack and you're right there.

(00:54:57):

So, there's some real meaningful companies as I mentioned, like Snowflake, Databricks, Confluent and others, but there's a lot of others that were obviated by technology from the infrastructure providers themselves. So those companies probably are the most at risk for a developer day from one of these big foundation model companies releasing exactly what they do. So there's probably a lot of people who need your tool, but the question will be if or when is probably the right way to think about it, one of these large infrastructure providers introduces a competitor, why will people continue to choose you? So it's a good market but it's a little bit close to the end as I said.

(00:55:38):

Then there's the applied AI market. I think this will play out for companies who build agents. I think agent is the new app. I think that's going to be the product form factor. There's companies like Sierra, we help companies build agents to answer the phone or answer the chat for customer experience and customer service. There's companies like Harvey that make agents for both a legal, paralegal profession, anti-trust reviews, reviewing contracts etc, etc. There's companies that do content marketing. There's companies that do supply chain analysis. I think this is like the software as a service market. They'll probably be higher margin companies because you're selling something that achieves a business outcome as opposed to being a byproduct of the models themselves. They will almost certainly pay taxes down to the model providers, which is why those model providers will end up extremely large scale but probably slightly lower margin and I think the market for them will be probably less technical.

(00:56:37):

If you think about the purest form of software as a service, it's not like you ask what database do you use? It's really about the feature and function. I think that's where agents will go. I think it's going to be more about product than it is about technology over time. Just going back to my metaphor, in 1998 when Mark and Parker started Salesforce, just getting that database running in the cloud was like a technical achievement. Nowadays, no one asks about that because you can just spin up a database in AWS or Azure and it's like no problem. I think today, orchestrating an agentic process on top of the models, sounds really fancy and it's really hard and all that stuff. I'm pretty sure that's going to be easy in three or four years. It's just like just as the technology improves. And so, over time you say, what is an agent company? Well, it looks a little bit like [inaudible 00:57:30] as a service.

(00:57:30):

You talk a little bit less about how you deal with the models in the same way modern SaaS, few people ask what database you use, but you'll probably ask a lot about the workflows and what business outcomes that you're driving. Are you generating leads for a sales team? Are you minimizing your procurement spend? Whatever value you're providing, it's going to slowly evolve towards that.

(00:57:53):

I'm very excited. I don't think startups should probably build foundation models. But you can shoot your shot if you have a vision for the future, go for it. But I think it's probably a challenging market that's already consolidated. I'm very excited about the other two markets. I'm particularly excited as building agents becomes easier, to see a lot of long tail agent companies come out. I was looking at a website for the top 50 software companies in the stock market and obviously, the top five are the big, big boom ones like Microsoft, Amazon, Google, all that, but the next 50 are all SaaS companies and some of them are very exciting, some of them are super boring, but this is how the software market has evolved and I think we're going to see something similar with agents.

(00:58:38):

It's not just going to be these huge markets like we're in customer service and software engineering. It's going to be a lot of things where people are spending a lot of time and resources that an agent can just solve, but it requires an entrepreneur who actually understands that business problem, and deeply, and I think that's where a lot of the value is going to be unlocked in the AI market.

中文翻译:

是的，埃隆是个例外。他有能力筹集数十亿美元，而我猜大多数听众没有。他是史上最伟大的创业者之一，你不能拿自己跟他比。

市场的第二部分是“工具”（Tooling）。淘金热中有很多卖铲子的人：数据标注、数据平台、评估工具。还有更专业的模型，比如 11 Labs 的高质量语音模型。如果你想在 AI 领域成功，你需要各种工具和服务。但工具市场有一定的风险，因为它“离太阳太近了”。如果你看 IaaS 市场和云工具市场（如 Confluent, Databricks, Snowflake），亚马逊和微软等巨头在这些领域都有竞争产品，因为它们与基础设施非常接近。每个基础设施提供商都想通过向上层移动来实现差异化。

虽然有 Snowflake 这样的大公司，但也有很多公司被基础设施提供商自带的技术所取代。这些公司最担心的就是基础模型公司在“开发者大会”上发布一个功能，正好覆盖了他们的业务。所以，虽然很多人需要你的工具，但问题是：当大型基础设施提供商推出竞争产品时，人们为什么还会继续选择你？所以这是一个好市场，但正如我所说，离终点有点近。

第三部分是“应用 AI”（Applied AI）市场。我认为这将是构建“智能体”（Agents）的公司的天下。我认为“智能体是新的 App”，这将是产品的形态。比如 Sierra，我们帮公司构建处理客服电话或聊天的智能体；比如 Harvey，为法律行业做智能体；还有做内容营销、供应链分析的公司。这就像 SaaS 市场，它们可能是利润率更高的公司，因为你卖的是业务结果，而不仅仅是模型的副产品。它们肯定要向模型提供商交税，这也是为什么模型提供商规模巨大但利润率略低的原因。

在最纯粹的 SaaS 形式中，没人问你用什么数据库，大家只关心功能。智能体也会如此。随着时间推移，重点会从技术转向产品。回到我的比喻：1998 年 Salesforce 刚开始时，把数据库跑在云端是技术成就；现在没人问这个，因为 AWS 随手就能开一个。今天，在模型之上编排智能体流程听起来很高级、很难，但我敢肯定三四年后这会变得很简单。随着技术进步，你会发现智能体公司越来越像“业务流程即服务”。

你会越来越少谈论如何处理模型，就像现代 SaaS 很少谈论数据库一样。你会更多地谈论工作流和业务结果：你是否为销售团队生成了线索？是否降低了采购支出？无论你提供什么价值，都会慢慢向此演化。

我非常兴奋。我不认为初创公司应该构建基础模型，但如果你有愿景，也可以尝试。我更看好另外两个市场。随着构建智能体变得容易，会出现大量“长尾”智能体公司。我看过股市市值前 50 的软件公司，前五名是巨头，接下来的 50 名全是 SaaS 公司。有些很刺激，有些很枯燥，但这就是软件市场的演变方式。智能体也会经历类似的过程。不仅会有客服、软件工程这样的大市场，还会有很多细分领域，只要那里有人力和资源消耗，智能体就能解决。这需要真正深入理解业务问题的创业者，我认为那是 AI 市场释放巨大价值的地方。

[00:59:00] Lenny Rachitsky

English:

That is incredibly helpful. This makes me think about, I had Marc Benioff on the podcast, you guys were co-CEOs and he was extremely agent-pilled. All he wanted to talk about was Agentforce. Clearly you're also very agent-pilled. What is it that-

中文翻译:

这非常有帮助。这让我想起 Marc Benioff 来播客时，他简直成了“智能体信徒”。他满脑子想聊的都是 Agentforce。显然你也是个“智能体信徒”。到底是什么——

[00:59:31] Bret Taylor

English:

If you talk to an economist like Larry Summers who, on the OpenAI board with me, they'll talk about what is the value of technology? Will it help strive productivity in the economy. And if you look at one of the big jumps in productivity in the economy was in the '90s, and I think a lot of folks I talked to think it was actually that very first wave of computing where people made ERP systems and just put accounting into computers and databases, even mainframes, we're talking like the PC era. Because it was such a huge step-up, just imagine the ledgers of numbers that you'd have for a large multinational company before and it truly just transformed departments.

(01:00:14):

I'll give you a little toy example. My dad just retired. He was a mechanical engineer and he was talking about when he first started his career in the late '70s and he went into a mechanical engineering firm, the majority of the firm were drafts people. So basically, you take an engineering design but you needed to do all the different vantage points and for all the different floors and to give to the contractor to do the thing. Now, there are zero drafts people at his company. You just make the design in first AutoCAD and now Revit and it's a 3D model and the drafting has actually been eliminated. It's just not a thing one needs to do anymore. The actual design and drafting, drafting is not a thing that exists. It's just a design. That's true productivity gains, right? It's like the job of the mechanical engineering firm was to do a design. The drafting was this necessary output for the contractor, but it wasn't really adding value. It was just like the supply chain change.

(01:01:12):

If you look at the history of the software industry from the PC on, there's been meaningful productivity gains but just not nearly as meaningful as that first huge jump. And I'm not smart enough to know exactly why, but it is interesting, the promise of productivity gains from technology hasn't been as realized I think as some people thought. I think agents will truly start to bend the curve again like we did in the very early days of computing because software is going from helping an individual be slightly more productive to actually accomplishing a job autonomously. And as a consequence, just like you don't need drafts people in a mechanical engineering firm, you just won't need someone doing that thing anymore. It means they can do something else that's higher leverage and more productive and you can actually... A smaller group of people can accomplish more and truly drive productivity gains in the economy.

(01:02:15):

And I think if you've ever sold enterprise software, you end up in these discussions as a vendor with the customer where you'll have a value discussion and you'll do these somewhat convoluted things like okay, it's like you're selling a sales thing. Okay, well, if every salesperson sells 5% more... And you should pay us a million dollars. And it's roughly that conversation and it's so unattributable especially... And it's why it's so hard to sell productivity software, which I learned the hard way, it's just hard to know what's the value of making everyone 10% more productive? Did you actually make them 10% more productive or did something else change? You don't really know all these things. But now with an agent actually accomplishing a job, not only is it actually truly driving productivity in a very real way, but it's measurable as well.

(01:03:10):

So all those things combined means I think this is actually a step change in how we think about software because it does a job autonomously, which is more self-evident, a productivity driver. It's measurable so people value it differently as well, which is why I also believe in outcomes-based pricing for software.

(01:03:32):

And all of that combined to me, it feels like as significant as the cloud or I think more technologically, but just in terms of how it transforms the business model of the software industry where there's going to be a before and after. I don't know how many people still sell perpetually licensed on premises software, but it's de minimis at this point. I think we're going to go through a similar transition. The whole market is going to go towards agents. I think the whole market is going to go towards outcomes-based pricing, not because it's the only way, but the market is going to pull everyone there because it's just so obviously the correct way to build and sell software.

中文翻译:

如果你和像 Larry Summers（和我一起在 OpenAI 董事会）这样的经济学家聊天，他们会谈论技术的价值：它是否能提高经济生产力？如果你看 90 年代生产力的巨大飞跃，很多人认为那是第一波计算浪潮带来的，人们建立了 ERP 系统，把会计账目放进计算机和数据库。因为那是一个巨大的进步——想象一下以前跨国公司的纸质账本，技术真正改变了部门运作。

举个小例子：我父亲刚退休，他是一名机械工程师。他说 70 年代末刚开始职业生涯时，机械工程公司大部分人是绘图员。你做一个工程设计，需要有人画出所有不同的视角、楼层，交给承包商。现在，他公司里一个绘图员都没有了。你直接在 AutoCAD 或 Revit 里做 3D 模型，绘图这个环节被消灭了。这就是真正的生产力提升：公司的任务是设计，绘图只是交给承包商的必要输出，它本身不产生价值。

纵观从 PC 时代以来的软件史，虽然有生产力提升，但都不如第一次飞跃那么显著。有趣的是，技术带来的生产力承诺并没有像人们想象的那样完全实现。我认为智能体将真正开始再次改变曲线，就像计算的早期时代一样。因为软件正在从“帮助个人提高一点效率”转变为“自主完成一项工作”。结果就是，就像机械工程公司不再需要绘图员一样，你不再需要有人专门做那件事了。这意味着人们可以去做更高杠杆、更有产出的事情，更少的人可以完成更多的工作，从而真正推动经济生产力。

如果你卖过企业软件，你就会知道，作为供应商，你和客户讨论价值时会非常纠结。比如你卖一个销售工具，你说：“如果每个销售多卖 5%，你就该付我们 100 万美元。”这种价值很难归因。这就是为什么生产力软件很难卖（我深有体会）：让每个人效率提高 10% 的价值到底是多少？你真的提高了 10% 吗？还是因为别的因素？你无法确定。但现在，智能体直接完成一项工作，这不仅是真实的生产力驱动，而且是可衡量的。

所有这些结合在一起，意味着这是我们思考软件方式的一个阶跃式变化。因为它自主完成工作，生产力驱动更显而易见；它是可衡量的，所以人们对它的估值也不同。这也是为什么我坚信“基于结果的定价”。

对我来说，这感觉和云计算一样重要，甚至在技术上更重要。它将改变软件行业的商业模式，会有一个明显的“之前”和“之后”。现在几乎没人再卖永久许可的本地软件了。我认为我们将经历类似的转型：整个市场将转向智能体，转向基于结果的定价。不一定是因为这是唯一的方式，而是因为市场会把每个人都推向那里，因为这显然是构建和销售软件的正确方式。

[01:04:08] Lenny Rachitsky

English:

Let me pull on that last thread. So we had Madhavan on the podcast recently, pricing expert, legend, monetizing innovation author and he talked about pricing strategy for AI companies and he was very much in your camp of, if you can, you need to price your product as an outcome-based product and the access uses exactly what you shared, which is, you can do that if you can attribute the impact and it's autonomous, it's running on its own. And he actually used Sierra as one of the shining examples of this being successful. Can you just briefly just explain a little bit what is outcome-based pricing for people that haven't heard this term before and then just how does it work for Sierra to give an example?

中文翻译:

我想顺着最后一点聊聊。我们最近邀请了定价专家 Madhavan（《货币化创新》作者），他也非常支持你的观点：如果可以，你应该按结果定价。前提是影响可归因且是自主运行的。他甚至把 Sierra 作为成功的典范。你能简要解释一下什么是“基于结果的定价”，并以 Sierra 为例说明它是如何运作的吗？

[01:04:45] Bret Taylor

English:

Yeah, I'll start with the example and then I'll broaden it. So at Sierra, we help companies make customer facing AI agents primarily for customer service, but more broadly, for customer experience. So if you have a problem with your [inaudible 01:04:58], you'll call or chat with Harmony, who's their AI agent. If you have ADT home security and your alarm doesn't work, you can chat with their AI agent. Sonos speakers, a lot of different consumer brands. And if you think about running a call center, there's a cost for every phone call that you take. Most of it is labor costs, but if you have, let's just say a typical phone call is anywhere between 10 and $20 USD. Some of it's software, some of it's telephony, but a lot of it is just like the hourly wage of the person answering the phone.

(01:05:32):

So if an AI agent can take that call and solve it, that is in the industry often called a call deflection or a containment. And that essentially means you saved, call it $15 because you didn't have to have someone pick up the phone. So in our industry, basically we say, "Hey, if the AI agent solves the customer's problem, they're happy with it and you didn't have to pick up the phone," there's a pre-negotiated rate for that and we call it resolution based. There are other outcomes as well. We have some sales agents being paid a sales commission, believe it or not. We do. We really think of our agents as truly customer experience like the concierge for your brand and we want to make sure that our business model is aligned with our customer's business model.

(01:06:22):

As you said, these agents need to be autonomous and the outcome has to be measurable. That's not always possible, but I think it's broadly possible. And what's really neat about it is if you talk to any CFO or head of procurement with their big vendors, they look at the bill of materials and it's overwhelming and it's impossible to know if you're getting the value that you hoped from that contract. I think consumption based, which was popular particularly in the infrastructure space is closer to it. But I'm not sure a token is actually a good measure of value from AI either. I always use the analogy, like right now, most of the coding agents are priced per token or per utilization, but there's this famous story of an Apple engineer who had a bad manager who's like had you report how many lines of code you wrote every day? Which every engineer in the world knows is an idiotic way to measure productivity.

(01:07:16):

He famously went in with a report that had a negative number because I think he did a big refactoring, deleted a bunch and it was his way of saying like, fuck you to the man. I think tokens are similar. Yeah, you used a lot of tokens, like good for you, did it produce a pull request that was good? And I think that's the whole point of all this. I think there's a huge difference between outcomes-based pricing and usage based pricing because especially in AI, they're not necessarily even correlated and you could have a long phone call and not solve the customer's problem and they give you a negative review online and call the call center again, all that effort was for nothing. In fact, you might've added negative value. And so, I am a huge believer in this.

(01:07:58):

And what's fun about it is it really just aligns... I think every technology company aspires to be a partner, not a vendor. And I think at Sierra, we are truly a partner to every single one of our customers because we're all aligned on what we want to achieve. And I think that is really where the software industry should go. It requires a lot of different shape of a company. You have to be able to help your customers achieve those outcomes. You can't just throw software at the wall because you'll never get paid if it doesn't. Your orientation becomes so extremely customer-centric when you do this the right way. I think it's just a better version of the software industry. So I think it's right from first principles, it's right for procurement partners and I think it's right for the world.

中文翻译:

好，我先举例再展开。在 Sierra，我们帮公司制作面向客户的 AI 智能体，主要用于客服，更广泛地说是用于客户体验。比如你用的产品出了问题，你会和他们的 AI 智能体沟通。如果你用 ADT 安防，报警器坏了，你可以和 AI 聊。还有 Sonos 音箱等很多消费品牌。

经营呼叫中心时，接听每个电话都有成本。大部分是人力成本，一个典型的电话成本在 10 到 20 美元之间。如果 AI 智能体能接听并解决问题，这在行业内被称为“呼叫转移”或“拦截”。这意味着你省下了 15 美元，因为不需要人工接听。所以我们的模式是：如果 AI 解决了客户问题，客户满意，且不需要人工介入，我们就按预先商定的费率收费，这叫“基于解决（resolution-based）”的定价。还有其他结果，比如我们的销售智能体甚至按销售佣金收费。我们希望商业模式与客户的业务目标保持一致。

正如你所说，智能体必须是自主的，结果必须是可衡量的。虽然不总是可行，但大部分情况下是可以的。有趣的是，如果你和任何 CFO 或采购主管聊，他们看大供应商的账单时会觉得头大，很难知道合同是否真的产生了预期的价值。按量计费（Consumption-based）在基础设施领域很流行，它更接近真相，但我认为“Token”并不是衡量 AI 价值的好标准。

我常打个比方：现在很多编程智能体按 Token 或使用量收费。但有个著名的故事，苹果公司一个工程师有个糟糕的经理，要求每天汇报写了多少行代码。全世界的工程师都知道这是衡量生产力的愚蠢方式。那个工程师后来交了一份代码行数为负数的报告，因为他做了一个大重构，删了很多代码。我觉得 Token 也是一样：你用了大量 Token，那又怎样？它产生了一个好的 PR（拉取请求）吗？

我认为“基于结果的定价”和“基于使用量的定价”有巨大区别。在 AI 领域，两者甚至不一定正相关。你可能打了一个很长的电话但没解决问题，客户还给了差评，那所有的努力都白费了，甚至产生了负价值。所以我坚信这种模式。

它的迷人之处在于它让供应商和客户真正对齐。每家科技公司都想成为“合作伙伴”而非“供应商”。在 Sierra，我们是客户真正的伙伴，因为我们的目标是一致的。这需要公司具备不同的形态：你必须能帮客户实现结果，否则你拿不到钱。这会让你变得极其以客户为中心。我认为这是软件行业更好的版本，符合第一性原理，对采购方友好，对世界也有好处。

[01:08:43] Lenny Rachitsky

English:

I've been chatting a little bit about productivity gains. There's a lot of skepticism in the headlines these days of just like what is AI actually doing? Is it actually helping people be more productive? There was a recent study actually, I don't know if you saw, where they showed engineers were less productive with AI because it was just putting them in different directions. They had to research all what's going wrong here? So I think CX is a really good example where you clearly are seeing gains. Are you seeing actual gains at your company or any other company you work with outside of CX in terms of productivity that is like clearly yes, this is working and a huge deal?

中文翻译:

我们聊到了生产力提升。现在媒体上有很多怀疑论：AI 到底在做什么？它真的帮人提高效率了吗？最近有一项研究显示，工程师用 AI 后生产力反而下降了，因为 AI 把他们引向了不同的方向，他们不得不花时间研究哪里出了错。客服（CX）是一个能看到明显提升的例子。在客服之外，你在自己公司或其他公司看到过那种“显然有效且意义重大”的生产力提升吗?

[01:09:15] Bret Taylor

English:

I'm extremely bullish on the productivity gains from AI, but I do think the tools and products right now are somewhat immature and it's quite counterintuitive. So, for example, almost every software engineering firm I know uses something like Cursor to help their software engineers. Most people use Cursor right now as a coding autocomplete, though they have a lot of agentic solutions and there's a lot of... OpenAI has Codex and Cloud has... I can't remember the Anthropic products. So there's lots of agentic agents coming as well. One of the interesting things because the technology is immature, the code it produces often has problems. There's a lot of people approaching this to actually realize those productivity gains because as any engineer who's written a lot of code will tell you, it's pretty easy to look at and edit and fix code you wrote.

(01:10:10):

Reviewing other people's code or particularly finding a subtle logical error in someone else's code is actually really hard. It's actually much harder than editing code that you wrote yourself. So if the code produced by a coding agent is often incorrect, it actually can take a lot of cognitive load and time to fix it. And in fact, if you end up producing lots of issues with your customers, you could end up producing a lot of features, but actually, is like mucking up the machine a little bit and having something that's not ideal. There's a couple of techniques that I think are interesting. First, I think there's a lot of AI starts now working on things like code reviews. I think this idea of self-reflection in agents is really important. Having AI supervise the AI is actually very effective. Just think about it this way, if you produce an AI agent that's right 90% of the time, that's not that great, but how hard would it be to make another AI agent to find the errors the other 10% of the time? That might be a tractable problem.

(01:11:10):

And if that thing's right 90% of the time, just for argument's sake, you can wire those things together and have something that's right 99% of the time. So it's just a math problem and it turns out that you can make something to generate code, you can make something to review code and you're essentially using compute for cognitive capacity and you can layer on more layers of cognition and thinking and reasoning and produce things increasingly robust. So I'm very excited about that. The other thing though is root cause analysis. So we have an engineer at Sierra who exclusively focuses on the model context protocol server serving our cursor instance. And our whole philosophy is, if cursor generated something incorrect, rather than just fixing it, try to root cause it. Try to get it so the next time Cursor will produce the correct code and essentially, is context engineering.

(01:12:05):

What context did Cursor not have that would've been necessary to produce the right outcome? So I think people who are trying to get productivity gains in departments like software engineering need to stop waiting for the models to magically work if they want to see the gains now. And you really have to create root cause analysis and systems and say, how do we go root cause every bad line of code and actually give the right context and produce the right system so the models can do it today? Over time that'll probably be less necessary and you'll have less context engineering necessary to do it, but you really have to think of this as a system and I think people are waiting for the models to just magically get better.

And I'm like, well that will happen eventually, but if you want the gains now you got to put in the work. That's essentially why applied AI companies exist.

(01:12:53):

And the work is non-trivial, but you can do it. And so, for customers using platforms like Sierra, yeah, AI agents aren't perfect, but we're creating a system that lets customers create a virtuous cycle of improvement. If you want to go from a 65% automated resolution rate to 75%, we have a billion tools to let AI help you do that, identify opportunities for improvement, figure out why people are frustrated, what new capabilities can we add to our agent to improve the resolution rate? And you let AI put the needles at the top of the haystack on your behalf and I think that's really the way to optimize these systems.

中文翻译:

我对 AI 带来的生产力提升非常乐观，但我认为现在的工具和产品还不够成熟，而且这其实很反直觉。例如，几乎我认识的每家软件公司都在用 Cursor。大多数人目前把 Cursor 当作代码自动补全工具，尽管它有很多智能体方案。因为技术还不成熟，它生成的代码经常有问题。

任何写过大量代码的工程师都会告诉你：修改自己写的代码很容易，但审查别人的代码，尤其是寻找细微的逻辑错误，其实非常难，比改自己的代码难得多。如果编程智能体生成的代码经常出错，修复它会消耗大量的认知负荷和时间。如果你因此给客户带来一堆 Bug，虽然你产出了很多功能，但实际上是在把事情搞砸。

有几种技术我觉得很有趣。首先，很多 AI 初创公司在做代码审查。智能体的“自我反思”非常重要，让 AI 监督 AI 非常有效。想象一下：如果一个 AI 智能体 90% 的时间是正确的，这不算太好；但如果再做一个 AI 智能体去专门找那 10% 的错误呢？这可能是一个可以解决的问题。如果第二个 AI 也有 90% 的准确率，把它们连在一起，你就能得到 99% 的准确率。这本质上是用计算力换取认知能力，通过叠加认知、思考和推理层，让系统变得越来越鲁棒。

另一件事是“根本原因分析”（Root Cause Analysis）。在 Sierra，我们有一名工程师专门负责为我们的 Cursor 实例提供模型上下文协议（MCP）服务。我们的哲学是：如果 Cursor 生成了错误的代码，不要只是修复它，要找到根本原因。尝试让 Cursor 下次能生成正确的代码，这本质上是“上下文工程”（Context Engineering）。

Cursor 缺少了哪些必要的上下文才导致了错误结果？想要在软件工程等部门获得生产力提升，就不能干等着模型奇迹般地变完美。你必须建立根本原因分析系统，去分析每一行烂代码，提供正确的上下文，构建正确的系统，让模型在今天就能胜任。随着时间推移，这种上下文工程的需求可能会减少，但你必须把它看作一个系统。很多人在等模型变强，我说：“那终究会发生，但如果你现在就想要收益，你就得投入工作。”这就是应用 AI 公司存在的意义。

这项工作并不简单，但你可以做到。对于使用 Sierra 平台的客户，AI 智能体并不完美，但我们提供了一个系统，让客户建立改进的良性循环。如果你想把自动化解决率从 65% 提高到 75%，我们有大量工具让 AI 帮你识别改进机会、分析用户挫败感、增加新功能。让 AI 帮你从大海里捞针，这才是优化这些系统的方法。

[01:13:28] Lenny Rachitsky

English:

I've never heard of this technique of improving Cursor by adding additional context. What's the actual way of doing that? You build an MCP server that everything runs through or is it like you add Cursor rules? What's the actual approach there?

中文翻译:

我从未听说过通过增加额外上下文来改进 Cursor 的这种技术。具体是怎么做的？是构建一个 MCP 服务器让所有东西都经过它，还是添加 Cursor 规则？实际的方法是什么？

[01:13:41] Bret Taylor

English:

I'm probably out of my depth here, but it's essentially MCP because that's how you provide context to Cursor. And I think that almost always when you have a model making a poor decision, if it's a good model, it's lack of context. And so, you really want to find the intersection of your particular product and code base with the context available to these coding agents and systems and fix it at the root is the principle here.

中文翻译:

这可能超出了我的技术细节掌握范围，但本质上是利用 MCP（模型上下文协议），因为那是你向 Cursor 提供上下文的方式。我认为，当一个优秀的模型做出糟糕的决定时，几乎总是因为缺乏上下文。所以，原则是找到你特定产品、代码库与编码智能体可用上下文之间的交集，并从根源上解决问题。

[01:14:06] Lenny Rachitsky

English:

Got it. That is very cool. I hadn't heard of people doing that, model context protocol, makes sense. We've talked about productivity gains outside TX. Just to give you a chance to share how amazing what you've built is, what are some of the gains you see from people using Sierra?

中文翻译:

明白了，这很酷。我还没听说过有人这么做，MCP 协议，有道理。我们聊了客服之外的生产力提升。给你个机会展示一下你构建的产品有多棒：使用 Sierra 的客户看到了哪些收益？

[01:14:18] Bret Taylor

English:

Yeah, our customers see anywhere between 50 and 90% of their customer service interactions completely automated, which I think is really exciting. And we serve just a really, really broad range of customers. We serve the health insurance industry, the healthcare provider space, banks. You can actually refinance your home using an agent. One of our customers built on our platform to the telecommunications industry, DIRECTV, SiriusXM to a lot of retailers as well, which is really fun. Everyone from Wayfair to clothing retailers like OluKai and Chubbies Shorts. What's really neat about it is it's a pretty diverse range of use cases and it's everything from helping you sign up for... We have an agent that helps with customer support in one of the big dating applications to helping you upgrade or downgrade your SiriusXM plan. Actually, it's really funny, we do technical support from everything from home alarm systems to sonar speakers to more recently, CAT scan machines, which I think is amazing.

(01:15:28):

So technicians going in and fixing the CAT scan machine can chat with an AI agent to help them guide them through that process. We're the leader in the space, we're trying to enable every company in the world to create their agent with their brand at the top that I think will become as meaningful of a digital touch point as their website or their mobile app. In the short term, it can really transform the costs of running a customer service team. And what's remarkable is do so with really high customer satisfaction scores. That Weight Watchers agent, I believe has a customer satisfaction score of 4.6 out of five, which is pretty amazing. And what's interesting about service too, it's often people having a problem. And so, when you have a clear, I don't know if you use them in the airport, I think that agent has a CSAT score of 4.7 out of five people are coming in with a problem and [inaudible 01:16:19] delighted. And I think that's really the opportunity here.

(01:16:22):

Our whole vision is that we're going to move towards a world where every single one of the interactions with your customers can be instant. It can be multilingual, it can be over audio, it can be over chat, it can be digital, it can be over the phone and it can be very personalized. And I think that's really, really exciting. And if you think about all the best moments you've had with a brand, it's like that store associate who you know, and it's like for me, it's like the butcher at the grocery store. I love to cook, he knows me. We talk. Can you actually produce that at scale for a company with 100 million customers and can you do it in a really personal way? And I think we're really on the cusp of enabling that.

中文翻译:

我们的客户看到 50% 到 90% 的客服互动实现了完全自动化，这非常令人兴奋。我们的客户群非常广泛：医疗保险、医疗服务提供商、银行。你甚至可以通过智能体办理房屋再贷款。我们的客户还包括电信行业的 DIRECTV、SiriusXM，以及 Wayfair、OluKai、Chubbies Shorts 等零售商。

最酷的是用例的多样性：从帮助你注册大型约会应用，到升级或降级你的广播套餐。有趣的是，我们还提供各种技术支持，从家庭报警系统、音箱，到最近的 CT 扫描机。维修 CT 机的技术人员可以与 AI 智能体聊天，引导他们完成维修过程。

我们是这个领域的领导者，目标是让世界上每家公司都能创建带有自己品牌标识的智能体。我认为智能体将成为与网站、App 同样重要的数字触点。短期内，它能彻底改变客服团队的成本结构。更了不起的是，它还能保持极高的客户满意度（CSAT）。Weight Watchers 的智能体评分是 4.6（满分 5 分）；机场的一个智能体评分高达 4.7。人们带着问题来，最后带着喜悦离开。

我们的愿景是：未来与客户的每一次互动都是即时的、多语言的、跨渠道的（语音、聊天、电话），并且是非常个性化的。想想你与品牌最美好的互动时刻，通常是那个认识你的店员。对我来说，是超市里那个认识我的屠夫，我们经常聊天。你能为拥有 1 亿客户的公司大规模实现这种个性化体验吗？我认为我们正处于实现这一目标的边缘。

[01:17:03] Lenny Rachitsky

English:

Let me ask you one more question before we get to a very exciting lighting round. There's a lot of founders struggling with go-to-market in AI with their AI apps. There's so many apps these days, so many products, so many things coming at buyers, at large B2B companies. Clearly you guys have figured something out. I imagine your name helps, investors help, but what have you learned about just how to successfully do go-to-market with an AI product, say an agent-specific product that you think would be helpful for folks trying to do this better?

中文翻译:

在进入闪电式问答之前，我再问一个问题。很多 AI 应用的创始人在“进入市场”（GTM）方面感到挣扎。现在有太多的 App、太多的产品涌向大公司的买家。显然你们已经摸索出了一些门道。我猜你的名声和投资者都有帮助，但关于如何成功推广 AI 产品（尤其是智能体产品），你学到了哪些对其他人有帮助的经验？

[01:17:35] Bret Taylor

English:

I think there's a small handful of go-to-market models that have been proven to work, and I think it's important to choose the right one for the product category you're going after. One category I would say is developer-led. This is somewhere famously Stripe and Twilio where probably two of the original that did this exceptionally. And essentially, the go-to-market motion there is to appeal to an individual engineer often within the department of the CTO who have accountability and a fair amount of latitude to choose a solution. This works if your product is a platform product. It doesn't work, for example, if your product is trying to help a line of business because lines of business typically don't have dedicated engineering teams or let alone, the latitude to just go download a new library or start using a web service like that. It particularly works well if you sell to startups just because startups tend to have engineering teams with quite a bit of latitude to choose services to help them solve the problem given by the founder.

(01:18:44):

Then there's product-led growth. It's a broad term, obviously every company's product matters, but product-led growth more specifically means users can sign up from the website, often get put on a trial. Often you can buy a couple of seats with a credit card and those work where your user and your buyer are the same person. So it works for small business software almost always because sole proprietors do everything. And so you're selling small business software like Shopify in the early days and there's a lot of other products like that where you're trying to sell to small merchants. That's great. It doesn't work well when your buyer and the user of the software are different. So I'll use the example of something like expense reporting software. The user of that software is an individual employee, but the buyer is often a finance department. And so having sign up and buy with your credit card doesn't make sense because the person using is not the person with the credit card and it just doesn't work.

(01:19:39):

And then there's direct sales. And direct sales had gone, I don't want to say out of fashion, but if I think of the best direct sales companies, probably there's a lot of lineage from Oracle, but you think SAP, Oracle, ServiceNow, Salesforce, Adobe perhaps, and there's others as well. And these were companies that sold into large lines of business in a relatively traditional sales motion. I think because product-led growth became very popular. I think a lot companies use that, which is great, that motion produces great products, but if PLG means that you aren't actually engaging with the buyer of your software, you're not going to grow. And so, I've actually seen more recently, with a lot of AI companies, direct sales come a little bit more back into fashion because I think so many of the opportunities in AI actually meet that qualification where the buyer and the user are not necessarily the same person and it really requires that go-to-market motion.

(01:20:40):

Where I see entrepreneurs stumble is they'll choose a go-to-market motion without thinking through what is the process of purchasing this software? What is the process of evaluating the value of this software? And I think people just need to be much more first principles about it and much more thoughtful about it. And candidly, I think a lot of companies should leverage direct sales more than they do. And even though because of the sometimes justified reputation of the quality of products of some of these direct sales companies, it had gotten a bad name. And I think I'm thankful to see it coming back in a lot of the AI market.

中文翻译:

我认为有几种被证明有效的 GTM 模型，关键是根据你的产品类别选择正确的一种。

第一种是“开发者驱动”（Developer-led）。Stripe 和 Twilio 是这方面的典范。这种模式是吸引 CTO 部门下的个体工程师，他们有权选择解决方案。这适用于平台型产品。但如果你的产品是服务于业务部门（Line of

Business）的，这就不管用，因为业务部门通常没有专门的工程团队，也没有权限随便下载个库或使用某个 Web 服务。这种模式在卖给初创公司时效果很好，因为初创公司的工程师有很大的自主权。

第二种是“产品驱动增长”（PLG）。这意味着用户可以从网站注册、试用，并用信用卡购买。这在“用户即买家”的情况下非常有效，比如面向小微企业（SMB）的软件（如早期的 Shopify）。但如果买家和用户不是同一个人，PLG 就不太灵。比如报销软件，用户是普通员工，但买家是财务部。员工没法用自己的信用卡买公司用的报销系统。

第三种是“直接销售”（Direct Sales）。直销曾一度被认为“过时”了，但像 Oracle, SAP, ServiceNow, Salesforce 都是靠这种传统销售模式成功的。因为 PLG 太流行了，很多公司都在用，这能产生好产品，但如果你不接触真正的买家，你就无法增长。最近我看到很多 AI 公司让直销重新流行起来，因为 AI 的很多机会都属于“买家与用户分离”的情况，必须靠直销。

我看到创业者犯的错误是：在选择 GTM 模式时，没有想清楚软件的购买流程是什么？评估价值的流程是什么？大家需要回归第一性原理，更深思熟虑一些。坦白说，我认为很多公司应该比现在更多地利用直销。虽然直销公司有时因为产品质量口碑不好而名声受损，但我很高兴看到它在 AI 市场回归。

[01:21:20] Lenny Rachitsky

English:

I feel like this message is something a lot of founders need to hear, especially founders that aren't from a business background that sales turns them off, they don't think they're going to be great at sales. Just this push of this might be what you have to get really good at and this is how you win and you can't just rely on product like growth.

中文翻译:

我觉得很多创始人需要听到这个信息，尤其是那些没有商业背景、对销售感到排斥、觉得自己不擅长销售的创始人。有时候你必须擅长销售才能赢，不能只依赖产品驱动增长。

[01:21:36] Bret Taylor

English:

Yeah.

中文翻译:

是的。

[01:21:38] Lenny Rachitsky

English:

Bret, is there anything else that you wanted to share? Any last nugget of wisdom? Anything you want to double click on before we get to our very exciting lightning round?

中文翻译:

Bret，还有什么想分享的吗？最后的智慧锦囊？在进入闪电式问答之前，有什么想深入探讨的吗？

[01:21:47] Bret Taylor

English:

No, go ahead.

中文翻译:

没有了，开始吧。

[01:21:48] Lenny Rachitsky

English:

Okay, let's do it. Here we go. Welcome to our very exciting lightning round. I've got five questions for you. Are you ready?

中文翻译:

好，开始。欢迎来到激动人心的闪电式问答。我有五个问题，准备好了吗？

[01:21:53] Bret Taylor

English:

Yeah, go ahead.

中文翻译:

准备好了，请讲。

[01:21:54] Lenny Rachitsky

English:

What are two or three books that you find yourself recommending most to other people?

中文翻译:

你最常向别人推荐的两三本书是什么？

[01:21:59] Bret Taylor

English:

I don't read a lot of nonfiction, but probably if I had to pick one in the area of the topics we talked about, Competing Against Luck, which was the book that produced Jobs to be Done, which is a framework I really believe in. My only critique is I think most of these business books should be like an article. So maybe buy the book and punch it into ChatGPT and get the summary. But buy the book it's Clayton Christensen talked about it, but it's a really good framework for thinking about delivering value with your products. And I think it definitely influenced me.

(01:22:37):

Actually one book I do recommend is Endurance, which is the story of Shackleton's trip to go to the South Pole. Half the book is him starving to death and eating seal meat with his crew of people frozen in their boat. I've never seen a better story of grit in my entire life. It's remarkable that it's a true story and if you're an entrepreneur going through a hard time, read that, you'd be like, okay, it could be worse. It's a great book too. It's just remarkable that it's a true story.

中文翻译:

我不怎么读非虚构类书籍，但如果要选一本与今天话题相关的，我会选《与运气竞争》（Competing Against Luck）。它是“待办任务”（Jobs to be Done）框架的来源，我非常认同这个框架。我唯一的批评是，大多数商业书其实写成一篇文章就够了，所以你可以买书然后让 ChatGPT 总结一下。它是克莱顿・克里斯坦森的作品，是思考产品价值的极佳框架，对我影响很大。

(01:22:37):

另一本我推荐的是《坚毅》（Endurance），讲的是沙克尔顿南极探险的故事。书的一半内容都在讲他和船员们如何在冰封的船上挨饿、吃海豹肉生存。这是我这辈子读过关于“坚韧”最好的故事。最了不起的是它是真人真事。如果你是正处于困境的创业者，读读这本书，你会觉得：“好吧，情况还没那么糟。”

[01:23:20] Lenny Rachitsky

English:

Do you have a favorite recent movie or TV show that you've really enjoyed?

中文翻译:

最近有什么非常喜欢的电影或电视剧吗?

[01:23:27] Bret Taylor

English:

I haven't gone to any new TV shows recently? We just watched Inception with the kids and they loved it and made me appreciate Christopher Nolan and what a cool movie. It's the type of movie when you watch the film and you can a conversations for two days afterwards about it. So, just a great film.

中文翻译:

最近没怎么看新剧。我们刚和孩子一起看了《盗梦空间》（Inception），他们非常喜欢。这让我再次感叹克里斯托弗・诺兰的伟大，这电影太酷了。它是那种看完之后能让你讨论两天的电影。

[01:23:59] Lenny Rachitsky

English:

Do you have a favorite product that you have recently discovered that you love or one you've loved for a long time?

中文翻译:

你最近发现并爱上的产品，或者一直以来非常喜欢的产品是什么?

[01:23:59] Bret Taylor

English:

I'm really a big fan of Cursor. I think it's changed. I love creating software and I'm excited though for agents. I've been really excited. I was very excited to see Codex from OpenAI and others. So I think Cursor will be in its current form, is a transition product. And I know they're working on agents as well, but I really enjoyed taking something I love and it's been my life's passion and really diving into this AI tool and seeing how it transforms how I create software. So I've just been spending a lot of time with the product just because it's so core to what I love to do and it's a really well crafted product.

中文翻译:

我是 Cursor 的超级粉丝。它改变了开发。我热爱编写软件，也对智能体感到兴奋。我认为 Cursor 目前的形态是一个过渡产品，我知道他们也在做智能体。我非常享受用这个 AI 工具来从事我毕生热爱的软件创作，看它如何改变我的工作方式。它是一个打磨得非常精良的产品。

[01:25:05] Lenny Rachitsky

English:

Do you have a favorite life motto that you often come back to and find useful in work or in life?

中文翻译:

你有没有什么座右铭，在工作或生活中觉得非常受用?

[01:25:05] Bret Taylor

English:

The best way to predict the future is to invent it, which I think I attribute to Alan Kay of Xerox PARC. He invented a lot of the core abstractions that we use in computing today. I'm an entrepreneur, it's why I love to build things and it's definitely a life motto for me.

中文翻译:

“预测未来最好的方式就是去创造它。”这句话出自施乐帕洛阿尔托研究中心（Xerox PARC）的 Alan Kay。他发明了我们今天使用的很多计算核心抽象。我是一名企业家，这就是我热爱构建东西的原因，这绝对是我的座右铭。

[01:25:29] Lenny Rachitsky

English:

I feel like many people say this, I feel like you've actually done this so many times. You're really living this motto. Final question. We talked about you inventing the like button at FriendFeed. Were other thoughts of what they would call it other than like? Was it just obviously like? Or was there other thinking there?

中文翻译:

很多人都说过这句话，但我觉得你真的多次践行了它。最后一个问题：我们聊到你在 FriendFeed 发明了“点赞”（Like）按钮。当时有没有考虑过叫别的名字？是显而易见就叫“Like”吗，还是有别的考量?

[01:25:48] Bret Taylor

English:

This was before emoji. So, if you read the comments on FriendFeed posts, at least 70% of them are cool or wow or yeah or neat. And one of the principle uses of FriendFeed was to have discussions about things. So you'd have a post and then a pretty fulsome discussion underneath. And compared to Twitter and others, it was a great place to have those discussions. And so, the product problem we were trying to solve is get all the one word answers out so that the discussion was actually actual comments as opposed to acknowledgements that you read the thing.

(01:26:30):

So, the original framing was one click comment. That was how we thought about it. And so, the first version that I made had a heart, and she denies remembering this, but Anna Yang, now Anna Muller who has worked at the company, she hated it. She said, "If I look at hearts on every post, I'm going to vomit. It's too much." And it also was interesting, we were simulating, it was like an article about a tragedy or something. A heart was just not the right thing. Like which actually turned out to be really hard to translate was just a much more neutral sentiment, and that's why it was hard to translate because it was subtle. So that's how we ended up with this.

(01:27:17):

We started with a heart, and I don't know if we ever heard the word love, but we definitely started off with the iconography and then like, which just felt like this positive yet as neutral as possible within the realm of positive so that it could work for a more complex story. But it was all because we needed a one-click comment. That's where the concept came from.

中文翻译:

那是表情符号（Emoji）流行之前的时代。如果你看 FriendFeed 帖子下的评论，至少 70% 都是“酷”、“哇”、“耶”或“赞”。FriendFeed 的主要用途是讨论，帖子下面会有很充实的讨论。我们想解决的产品问题是：把这些“一个词的回答”过滤掉，让讨论区充满真正的评论，而不是仅仅表达“我读过了”。

(01:26:30):

所以最初的构思是“一键评论”。我做的第一个版本用的是“爱心”图标。当时在公司工作的 Anna Yang（现在叫 Anna Muller，她不承认记得这事）非常讨厌它。她说：“如果每个帖子上都看到爱心，我会吐的，太过了。”而且如果是一篇关于悲剧的文章，爱心显然不合适。“Like”这个词其实很难翻译，因为它是一种非常中性的情感表达。

(01:27:17):

我们从爱心图标开始，然后选择了“Like”这个词，因为它既积极又尽可能保持中性，可以适用于复杂的故事。这一切都是因为我们需要一个“一键评论”的功能。

[01:27:48] Lenny Rachitsky

English:

Bret, this was incredible. This was an honor. I so appreciate you coming on this podcast. Two final questions, where can folks find you online if they want to reach out, maybe go see if they want to work at Sierra and how can listeners be useful to you?

中文翻译:

Bret，这太精彩了。这是我的荣幸，非常感谢你参加播客。最后两个问题：如果大家想联系你，或者想去 Sierra 工作，可以在哪里找到你？听众能为你做些什么？

[01:28:00] Bret Taylor

English:

If you want an AI agent to help with customer service, go to sierra.ai. If you want to apply here, sierra.ai/careers where we have offices in San Francisco and New York, Atlanta and London and are hiring pretty aggressively in every department. So please reach out if you're interested.

中文翻译:

如果你需要 AI 智能体来协助客服，请访问 sierra.ai。如果你想申请工作，请访问 sierra.ai/careers。我们在旧金山、纽约、亚特兰大和伦敦都有办公室，各部门都在积极招聘。感兴趣请联系我们。

[01:28:19] Lenny Rachitsky

English:

And how can listeners be useful to you? Is it tryout Sierra, anything else there?

中文翻译:

听众能帮你什么？试用 Sierra，还有别的吗？

[01:28:21] Bret Taylor

English:

Yeah, tryout Sierra. I'm a single issue voter.

中文翻译:

是的，试用 Sierra。我是个“单议题投票者”（意指目前只关注这一件事）。

[01:28:27] Lenny Rachitsky

English:

Bret, thank you so much for being here.

中文翻译:

Bret，非常感谢你能来。

[01:28:29] Bret Taylor

English:

Yeah, thanks for having me.

中文翻译:

谢谢邀请。

[01:28:30] Lenny Rachitsky

English:

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

大家再见。非常感谢收听。如果你觉得有价值，可以在 Apple Podcasts、Spotify 或你喜欢的播客应用中订阅。另外，请考虑给我们评分或留下评论，这能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。