FARHAN THAWAR

LENNY'S PODCAST

BILINGUAL TRANSCRIPT

ORIGINAL BY

Lenny Rachitsky

@lennysan • x.com/lennysan

ANALYSIS BY

@Penny777 • x.com/penny777

Generated: 2026-01-16

Farhan Thawar - 双语对照

This is the complete bilingual transcript for Lenny's Podcast featuring Farhan Thawar (VP and Head of Engineering at Shopify).

[00:00:00] Farhan Thawar

English:

If you do the hard path and it doesn't work, actually you still win because you've now done something hard. You've probably worked with smart people. You've learned something along the way that is valuable. I meet lots of job seekers. I go, what are you doing to try to find a job? Are you really learning anything from sending out 10 resumes a day? Why don't you look at the API Docs and build something? Even if you don't get a job at Shopify, you've learned something.

中文翻译:

如果你选择了那条困难的道路，即便最后没成功，实际上你还是赢了，因为你做了一件难事。你可能在这个过程中与聪明的人共事，学到了很有价值的东西。我见过很多求职者，我会问：“为了找工作你都在做什么？”每天投10份简历真的能让你学到东西吗？为什么不去看看 API 文档然后动手做点东西呢？即便你最后没能进 Shopify，你也学到了真本事。

[00:00:20] Lenny Rachitsky

English:

First, I want to talk about another theme, creating intensity in your organization.

中文翻译:

首先，我想谈谈另一个主题：如何在组织中创造“强度”（Intensity）。

[00:00:24] Farhan Thawar

English:

Everyone says, "Oh yeah, work hard and do more hours when you're young, whatever." I'm like, "What if you just did more per minute?"

中文翻译:

每个人都会说：“噢，趁年轻要努力工作，多加班”之类的。我的想法是：“如果你能提高每分钟的产出呢？”

[00:00:29] Lenny Rachitsky

English:

The more I dig into the Shopify way working, the more fun stuff I never expected emerges. There's been a drive to delete code and simplify.

中文翻译:

我越深入研究 Shopify 的工作方式，就发现越多意想不到的趣事。比如，他们有一种删除代码和简化系统的驱动力。

[00:00:37] Farhan Thawar

English:

We have a Delete Code Club. We can always almost find a million-plus lines of code to delete, which is insane.

中文翻译:

我们有一个“删代码俱乐部”。我们几乎总能找到超过一百万行可以删除的代码，这简直疯狂。

[00:00:42] Lenny Rachitsky

English:

I found this great quote from you, "Not everyone can look stupid in public over and over, but I believe it's my superpower."

中文翻译:

我发现你有一句名言：“不是每个人都能忍受一次又一次在公众面前显得愚蠢，但我认为这是我的超能力。”

[00:00:48] Farhan Thawar

English:

I have been in many situations with many sharp people who have said to me, that's the stupidest fucking question I've ever heard. My goal there is not to annoy the person, but it's to understand the content.

中文翻译:

我曾多次遇到过一些非常厉害的人，他们对我说：“这是我听过最他妈愚蠢的问题。”我的目的并不是要激怒对方，而是为了真正理解内容。

[00:00:59] Lenny Rachitsky

English:

I was looking at your LinkedIn and your career history, and I noticed that you've worked for a different billionaire every decade of your life.

中文翻译:

我看过你的 LinkedIn 和职业生涯，我注意到你人生中的每一个十年都在为不同的亿万富翁工作。

[00:01:05] Farhan Thawar

English:

They're mostly different people, but they're similar in one thing is that they haven't...

中文翻译:

他们大多是不同的人，但在一点上非常相似，那就是他们从未……

[00:01:12] Lenny Rachitsky

English:

Today my guest is Farhan Thawar. Farhan is Vice President and Head of Engineering at Shopify. Shopify is an incredibly interesting company because they have over 10,000 employees who are fully remote, and even though they were founded almost 20 years ago, they continue to operate with urgency, velocity and have very first principles, ways of thinking, which translates into them seeing record usage, blowing away their earnings calls just recently, and building a beloved product. A lot of this is thanks to Farhan, who in our conversation shares very specifically what he's done to maintain intensity and urgency within the engineering team. Including their meeting cadences, the counter-intuitive power of pair programming, how they run meetings, how they cancel meetings constantly and so much more. He also shares his experience with indexing towards choosing the harder option when you have multiple options to choose from and why that ends up making your life easier.

(00:02:08):

He also shares a bunch of great hiring advice and a bunch of hiring stories which are going to blow your mind. He also talks about their engineering intern program where they're going to hire over a thousand engineers just for their intern program in 2025. I've had a lot of people on this podcast from Shopify, but that is for a very good reason because this company and its leaders have a lot to teach us about how to run an incredible business and build an incredible product. If you enjoy this podcast, don't forget to subscribe and follow it in your favorite podcasting app or YouTube. It's the best way to avoid missing future episodes and it helps the podcast tremendously. With that, I bring you, Farhan Thawar. Farhan, thank you so much for being here, and welcome to the podcast.

中文翻译:

今天的嘉宾是 Farhan Thawar。Farhan 是 Shopify 的副总裁兼工程负责人。Shopify 是一家非常有趣的公司，他们拥有超过一万名员工，且完全远程办公。尽管公司成立已近 20 年，但他们依然保持着紧迫感、高速运转，并坚持“第一性原理”的思考方式。这转化为创纪录的用户使用量，在最近的财报会议中表现惊人，并打造出了备受喜爱的产品。这在很大程度上要归功于 Farhan。在我们的对话中，他详细分享了他是如何维持工程团队的强度和紧迫感的，包括他们的会议节奏、结对编程（Pair Programming）那违反直觉的力量、他们如何开会、如何不断取消会议等等。他还分享了在面对多个选项时，为何倾向于选择更难的那个，以及这为何最终会让生活变得更轻松。

(00:02:08):

他还分享了许多精彩的招聘建议和令人惊叹的招聘故事。他还谈到了他们的工程实习生计划——2025 年他们将仅为实习计划就招募超过一千名工程师。我邀请过很多来自 Shopify 的嘉宾，这是有原因的，因为这家公司及其领导者在经营卓越业务和打造卓越产品方面有很多值得我们学习的地方。如果你喜欢这个播客，请不要忘记在订阅并关注。这是避免错过后续节目的最好方式，也对我们帮助巨大。现在，让我们欢迎 Farhan Thawar。Farhan，非常感谢你能来，欢迎来到本播客。

[00:02:51] Farhan Thawar

English:

Thanks for having me.

中文翻译:

谢谢邀请。

[00:02:52] Lenny Rachitsky

English:

As I was preparing for our conversation, I talked to a bunch of people that you've worked with over the years, and there's basically three themes that kept coming up over and over and over. One is hiring, two is creating intensity in your organization, and three is choosing the hard path. First of all, does that resonate? Second of all, does it sound good to talk about these three themes in our time together?

中文翻译:

在准备这次对话时，我联系了一些多年来与你共事过的人，基本上有三个主题被反复提及：一是招聘，二是创造组织强度，三是选择困难的道路。首先，这让你产生共鸣吗？其次，在接下来的时间里讨论这三个主题你觉得可以吗？

[00:03:16] Farhan Thawar

English:

Yeah, I mean, I have ideas on where all three of those things came from, and I think that it is something that if you looked back on my career, I've hit points on each of those things, but I don't think at the onset I knew that that's what I was doing, but it turns out in retrospect, that's what I ended up doing.

中文翻译:

是的，我知道这三点是怎么来的。回顾我的职业生涯，我在每个阶段都触及了这些点。虽然在开始时我并没意识到自己在这么做，但回过头来看，这确实是我最终呈现出来的特质。

[00:03:32] Lenny Rachitsky

English:

Perfect, so this is the Steve Jobs. Everything looking backwards it all connects.

中文翻译:

太棒了，这就是史蒂夫·乔布斯说的那样：向后看时，一切都串联起来了。

[00:03:36] Farhan Thawar

English:

Yes.

中文翻译:

没错。

[00:03:37] Lenny Rachitsky

English:

[Sponsor Message for DX and Persona - Omitted for brevity as per standard transcript practice, but can be included if required]

... Okay, so let's start by talking about the hard path.

中文翻译:

[赞助商广告 - 略]

……好，那我们先从“困难的道路”聊起。

[00:05:42] Farhan Thawar

English:

Okay.

中文翻译:

好的。

[00:05:43] Lenny Rachitsky

English:

Advice that I've heard you share with people often is when they're trying to decide amongst a bunch of options is to choose the harder path because that makes life easier down the road. Share this advice why it's so important, where you learned, where you developed that this is the right approach.

中文翻译:

我经常听到你给别人的建议是：当在一堆选项中做抉择时，要选择那条更难的路，因为这会让以后的生活变得更轻松。请分享一下这个建议，为什么它如此重要？你是从哪里学到或总结出这是正确方法的？

[00:05:59] Farhan Thawar

English:

But the short version is that if you have a choice and you choose the easy thing and it works great. If you choose the hard thing and it works great, you did more work, but if it doesn't work and you chose the easy thing, you've actually not learned anything because you chose the... You haven't done a lot of work, you haven't probably worked with the smartest people because they're not usually around on the easy path, and what happens is you've gone through this exercise and now you're like, I've kind of lost. I lost the choice, or I was trying to do something, it didn't work out for me. But if you do the hard path and it doesn't work, actually you still win because you've now done something hard. You've probably worked with smart people, you've learned something along the way that is valuable, and I can give you a quick example.

(00:06:41):

So I meet lots of job seekers and they're like, I go, "What are you doing to try to find a job?" I'm like, "I'm sending out 10 resumes a day." I'm like, "Okay." That sounds kind of easy, and are you really learning

anything from sending out 10 resumes a day? Versus I would say to them, "Hey, you know all these companies you're interested in, Shopify might be one of them. Why don't you look at the API Docs and build something, build a Shopify app, build an admin extension, build something on top of Shopify." Even if you don't get a job at Shopify, which is maybe your goal, you've learned something, you've built something, you have things in your GitHub repo now and you can show people. You're learning about the product that might translate to a job you might get somewhere else. So I think that even though it's harder, right? Of course, you can't every day build an app on a different platform.

(00:07:23):

Maybe you can once a day. You will learn something in the hard path. And the same thing happened to me in taking hard courses. I would get worse marks, but I ended up meeting smarter people in those courses because they were there for the same reason I was because the content was hard. And so that's something that I've just realized in my life that if I do the hard thing and I just naturally tend towards doing that, I ended up doing... I went to Waterloo and I did a minor in electrical engineering on top of computer science, and when I did my MBA, I did a minor in financial engineering because the smarter people were in that path and they're still my friends today.

中文翻译:

简短的版本是：如果你面临选择，选了容易的事且成功了，那很好。如果你选了难的事且成功了，你付出了更多努力。但如果失败了，而你选的是容易的事，你其实什么都没学到，因为你没投入多少工作，也没能和最聪明的人共事（因为聪明人通常不在容易的道路上）。结果就是你经历了一番尝试，却感觉彻底输了。但如果你走的是困难的路却失败了，你实际上还是赢了，因为你挑战了高难度，可能结识了聪明人，学到了宝贵的经验。

(00:06:41):

举个例子：我遇到很多求职者，我问：“你为了找工作在做什么？”他们说：“我每天投10份简历。”我说：“好吧，这听起来挺容易的，但你真的能学到东西吗？”相反，我会对他们说：“嘿，既然你对这些公司感兴趣，比如 Shopify，为什么不去看看 API 文档，动手做一个 Shopify 应用、一个管理后台扩展，或者在 Shopify 之上开发点东西？”即便你最后没拿到 Shopify 的 Offer，你也学到了东西，亲手做出了成果，你的 GitHub 仓库里有了可以展示的内容。你对产品的理解可能会转化为在其他地方获得工作的机会。虽然这更难，你不可能每天都在不同平台上开发应用。

(00:07:23):

但哪怕你偶尔做一次，你也能从困难的路径中学到东西。我在选修高难度课程时也有同样的经历。虽然我的分数可能更低，但我遇到了更聪明的人，因为他们也是为了挑战硬核内容才聚在那里的。这就是我人生中的感悟：如果我选择做难事，并且自然而然地倾向于此，结果往往更好。我在滑铁卢大学读计算机科学时辅修了电子工程；读 MBA 时辅修了金融工程，就是因为聪明人都在那条路上，他们至今仍是我的朋友。

[00:07:54] Lenny Rachitsky

English:

So building on the last point you just made, people could hear this and think, okay, if it's harder, that's going to be the right path. Sometimes harder is still a bad idea. For example, joining a terrible company that's extremely frustrating to work at or building, I don't know, a house in a really dumb way, but it's just really hard. What else do you find is important to think about when you're thinking it's not just harder, but also XYZ should probably be true?

中文翻译:

基于你刚才说的，人们可能会想：只要更难，就是正确的路。但有时“更难”依然是个坏主意。比如，加入一家工作体验极差的烂公司，或者用一种极其愚蠢但非常辛苦的方式盖房子。除了“更难”之外，你认为还需要

满足哪些条件（XYZ）才值得去选？

[00:08:20] Farhan Thawar

English:

Yeah, one real one is of course the people, because I find that my path has always focused on trying to pick the best learning journey. Where can I learn the most? And for me, everyone's different. Some people might learn better from books or the domain they're in, but for me, I learn a lot from people, and so I try to put myself in those harder rooms on purpose. There was a time when I was doing my MBA in financial engineering, by the way, and I'm a tech guy and still a tech guy, and all these people were going into finance jobs.

(00:08:46):

There was a point where somebody said to me, "Why are you in this class?" Because they knew that I was doing it for fun, and so it was because it was the learning journey. And so I would say a big part of it for me is yes, there's the how do you win even if you lose, because if it goes poorly, you can still come out of it with skills, but if you actually take the hard path, you'll have these intense working relationships with smart people that again will continue on in your life.

(00:09:12):

And that also just forces you to be in this constant state of uncomfort by going into these rooms and saying, "I don't know anything," and it's harder. And I agree with you. You don't want to do dumb things like, oh, let's just do this thing in a dumb way. That's not what I mean. I mean, let's try to do the hard thing that we can learn from. And by the way, it happens to be on the path. It's just is it might take more manual work or it might not be the way most people do it, but we think we can learn more from that path.

中文翻译:

是的，一个核心因素当然是“人”。我发现我的职业路径始终专注于选择最佳的“学习旅程”。哪里能让我学到最多？每个人不一样，有人看书学得快，有人在特定领域钻研，而我从人身上学到最多。所以我会有意识地让自己进入那些“更难进的房间”。顺便说一下，我读金融工程 MBA 时，我本身是个技术人，现在也是，而班上其他人都是为了进金融行业。

(00:08:46):

当时有人问我：“你为什么在这门课上？”因为他们知道我纯粹是为了好玩。对我来说，这就是学习旅程。所以我觉得很大一部分在于：即便失败了你也能赢，因为你能带着技能离开；而且如果你走困难的路，你会与聪明人建立深刻的合作关系，这些关系会伴随你一生。

(00:09:12):

这也会强迫你处于一种持续的“不适状态”，进入那些房间并承认“我什么都不懂”，这很难。我同意你的观点，不要为了做而做，比如用愚蠢的方式做蠢事。我的意思是：尝试去做那些能让我们学到东西的难事。它恰好在你的路径上，只是可能需要更多体力活，或者不是大多数人走的路，但我们认为能从中学到更多。

[00:09:37] Lenny Rachitsky

English:

Speaking of that, I found this great quote from you, "Not everyone can look stupid in public over and over, but I believe it's my superpower and I try to make it my whole team's superpower too."

中文翻译:

说到这，我发现你有一句很棒的话：“不是每个人都能忍受一次又一次在公众面前显得愚蠢，但我认为这是我的超能力，我也试着让它成为我整个团队的超能力。”

[00:09:49] Farhan Thawar

English:

Yeah, I mean, it sounds funny, but again, I'm the one who's always trying super dumb things and sometimes they work, and even my wife hates that I try these things even at home, right? I'll just like, what's an example? It might be a new washing machine and I might try some weird mode with some clothes, and I'm like, "Oh, you ruin the clothes." I'm like, okay. But now I know that this mode should not be used, but maybe I would've uncovered that there's some super fast, quick wash that I can do in 20 minutes that now saves us 40 minutes of wash time every single time we use the washing machine.

(00:10:22):

So there's things like that, but I will ruin lots of clothes trying to do that, but the same at work. We'll try things and sometimes it can lead to disaster, hopefully not, but you can imagine people trying to like, oh, let me try this new configuration of GCP and maybe we'll get some benefit, but maybe we'll take Shopify down. We don't want to do that. So you want to have some sort of guardrails. But there is something around trying dumb things and saying dumb things. Half the time, by the way, when I say something dumb, people go, "I had the same question." They just were scared to say it.

中文翻译:

是的，这听起来很有趣。我总是那个尝试超级愚蠢事情的人，有时它们真的奏效。甚至我妻子都讨厌我在家里尝试这些，比如新买的洗衣机，我会尝试用某种奇怪的模式洗衣服，结果她说：“噢，你把衣服毁了。”我说：“好吧，现在我知道这个模式不能用了。”但也许我会发现某种超级快速的洗涤模式，只要 20 分钟，以后每次洗衣服都能省下 40 分钟。

(00:10:22):

就是这类事情。虽然我会毁掉很多衣服，但在工作中也是一样。我们会尝试新事物，有时会导致灾难（希望不会），比如有人想尝试 GCP（谷歌云）的新配置，也许会有收益，但也可能让 Shopify 宕机。我们不希望那样，所以需要一些护栏。但尝试蠢事和说蠢话是有价值的。顺便说一下，有一半的时间当我问出愚蠢的问题时，别人会说：“我也想问这个。”他们只是不敢说出来。

[00:10:55] Lenny Rachitsky

English:

So for folks that may want to, because I feel like this skill is so hard, but so important, being okay with failing, being okay with looking dumb, is there something you tell people to help them build this other than just like, I'm genetically good at this stuff? What helped you become comfortable with being wrong and failing before you were a big shot exec where it's like, oh, he's fine. He knows what he's doing.

中文翻译:

对于那些想要培养这种能力的人——因为我觉得这种“不介意失败、不介意显得愚蠢”的技能很难但极其重要——除了“我天生擅长”之外，你有什么建议吗？在你成为大权在握的高管（大家会觉得你做什么都有道理）之前，是什么让你能够坦然面对错误和失败的？

[00:11:17] Farhan Thawar

English:

I don't know. I mean, I kind of grew up working in retail and people come into the store, and then I would say, "Hey," and you're working on commission, and they're not always buying stuff, and if they don't buy it, you don't make any money. And so maybe just the fact of going up and forcing myself to talk to people and then trying to get them to, and maybe you spend an hour with a client and then they don't buy anything, but you're getting that reaction of a bunch of negatives, and all you have to do is say, okay, and just go to the next customer. You can't really well on it and be like, oh my God, my whole day is ruined. But instead, you have to learn from that and say, "Okay, let me try this. Let me try that." And it's not easy, but it was a way to build up some confidence and people say, this telemarketing, or there's a bunch of things you can do to get a lot of rejection. Cold calling is another one, and that can lead you to actually building up resistance. I don't know if I'm genetically better at it or not. I just think that I literally don't care if I look dumb. I've always said the dumb thing, I'm not doing things on purpose to get nos, which by the way, is part of some sales training, which is go and get 10 nos. But I haven't done that. But I have been in many situations with many sharp people, business people, successful people who have said to me... Turn around and said, "That's the stupidest fucking question I've ever heard." I've definitely had that happen to me. And I'm like, "All right, let's move on to the next question."

中文翻译:

我也不太确定。我是在零售业打工长大的，客人进店，我得打招呼，我是拿佣金的。他们不一定买东西，不买我就没钱。所以，可能仅仅是强迫自己走向陌生人、与他们交谈、尝试推销，即便花了一小时客户最后什么都没买，你面对的是一堆负面反馈，而你唯一能做的就是说声“好吧”，然后走向下一个客户。你不能沉溺其中，觉得“天呐，我这一天全毁了”。相反，你得从中学习：“好吧，下次我试试这个，试试那个。”这不容易，但这是建立自信的一种方式。人们常说电话推销或很多事会让你遭遇大量拒绝，冷启动电话（Cold calling）也是一种。这些经历能让你产生免疫力。我不知道我是否天生擅长，我只是真的不在乎自己看起来蠢不蠢。我总是说蠢话，但我不是为了被拒绝而故意这么做（虽然有些销售培训会让你去收集 10 个拒绝）。我经历过很多场合，很多精明的人、成功的生意人转过头对我说：“这是我听过最他妈愚蠢的问题。”我确实遇到过，而我的反应是：“行，那我们看下一个问题。”

[00:12:36] Lenny Rachitsky

English:

I love that attitude and I think that's key to it. It's just bounce off of it and not be crumble and I think it's empowering for folks to hear from someone like you that has done so well that people tell you that is the dumbest question I've ever heard-

中文翻译:

我喜欢这种态度，这是关键。就是被弹回来而不是被击垮。听到像你这样成功的人也会被别人说“这是我听过最蠢的问题”，这对大家来说非常有力量。

[00:12:48] Farhan Thawar

English:

Oh yeah, still.

中文翻译:

噢是的，现在依然如此。

[00:12:48] Lenny Rachitsky

English:

Still. Okay.

中文翻译:

依然如此。好吧。

[00:12:50] Farhan Thawar

English:

Yeah. So how about this? That I heard that's the dumbest fucking question. And then recently I heard, I've already explained this to you three times because I kept asking and I didn't understand, and literally I got this message back saying, "I've already explained this to you three times." I'm like, "Okay, still don't get it." So my goal there is not to annoy the person, but it's to understand the content. And actually, by the way, I say these were messages, I saved them. I literally screenshot it because I'm like, remember this? It doesn't matter. I'm trying to learn.

中文翻译:

是的。比如最近，我听到有人说：“我已经给你解释过三遍了。”因为我一直在问，我还是没听懂。对方回消息说：“我已经解释三遍了。”我的反应是：“好吧，但我还是没懂。”我的目的不是要烦他，而是为了理解内容。顺便说下，这些消息我都保存了，我截图了，因为我想提醒自己：这没关系，我是在学习。

[00:13:20] Lenny Rachitsky

English:

One more question along these lines. I was looking at your LinkedIn and your career history, and I noticed that you worked for a different billionaire every decade of your life. So there's a guy named Joe Lemond in your twenties and Chamath in your thirties and Toby this decade, maybe yourself next decade, if things go well. Other than what you've shared or maybe it is what you've shared, is there a thread across these three folks that have been really successful that you've learned from that maybe is consistent across them all or even just specific to each one?

中文翻译:

关于这一点还有一个问题。我看你的 LinkedIn 经历，发现你每个十年都在为不同的亿万富翁工作。20 多岁是 Joe Lemond，30 多岁是 Chamath，这十年是 Tobi（Shopify 创始人），如果顺利的话，下个十年可能是为你自己。除了你已经分享的，这些非常成功的人身上有没有什么共同的特质，是你学到的，或者是每个人特有的？

[00:13:49] Farhan Thawar

English:

It's interesting because again, this is looking back, you're like, wait a sec, I didn't plan it this way. There's no way you could plan it, right? I'm going to work for a different billionaire every decade. That's not how it works but they're similar. They're mostly different people, but they're similar in one thing is that they have an irrational view of what the world should look like over the next decade or so. They're very long-

term thinkers, very rational in that they'll look and say, "Hey, in 10, 15, 20, 25 years, this is what the world's going to look like." And I'm not good at seeing that vision, but I'm good at trying to move towards that vision 1% a week. And so the melding of the two, I know where I'm good and I'm good at pushing the ball forward. And if they're good at the long-term vision, we can both align to say, "You're good at this thing. I'm good at this thing. Why don't we merge forces?"

(00:14:35):

And so that is something that has resonated with me is like, how do I find these irrational all progress depends on the unreasonable man. How do I pair with these people because I'm altogether too reasonable and there's no way for me to become unreasonable, and so I have to merge with these people. And so that is again, something that I specifically have sought out. And even when I was starting my own company in 2015, I actually sat down and wrote a list of all the unreasonable people that I knew in Toronto, and I went down the list and met every single one of these entrepreneurs to figure out are we API compatible and could I work with them? And I ended up picking one of them and starting a company.

中文翻译:

这很有趣，因为这又是“向后看”总结出来的，我并没有计划过。你不可能计划“我每个十年都要为不同的亿万富翁工作”。但他们确实有相似之处。虽然性格迥异，但共同点是：他们对未来十年左右的世界应该是什么样子有着一种“非理性”的愿景。他们是非常长期的思考者，在某种程度上又非常理性，他们会说：“嘿，10、20、25 年后，世界会是这个样子。”我不擅长描绘那种愿景，但我擅长每周朝着那个愿景推进 1%。所以这是两者的结合：我知道自己的长处是“推着球往前走”，如果他们擅长长期愿景，我们就能达成一致：“你擅长这个，我擅长那个，我们联手吧。”

(00:14:35):

这让我产生共鸣：我该如何找到这些“非理性”的人？正如那句话所说，“所有的进步都取决于那些不讲理的人（Unreasonable man）”。我如何与这些人搭档？因为我太“讲理”了，我没法变得不讲理，所以我必须与他们融合。这是我有意识去寻找的。甚至在 2015 年我自己创业时，我坐下来写了一份多伦多所有我认识的“不讲理的人”的名单，我挨个去见这些创业者，看我们的“API 是否兼容”，我能不能和他们共事。最后我选了其中一个，一起创办了公司。

[00:15:15] Lenny Rachitsky

English:

That is an amazing story. So first of all, I just love this insight that being aware of this is not a superpower of mine and I'm not going to try to build it. I'm going to find someone to merge with, and connect your APIs together and be the person that builds it, not the person that envisions what to build. I think that's awesome because a lot of people... I need to get good at all these things. I need to become the best at vision and strategy and execution and collaboration and then all these things. And so I think alone, this isn't really interesting instead it's just recognize your strengths and weaknesses and double down on your strengths.

中文翻译:

这个故事太棒了。首先，我非常喜欢这个洞察：意识到某件事不是我的超能力，我也不打算去强行培养它，而是找一个能互补的人，把我们的“API”连接起来，做那个负责构建的人，而不是负责构思的人。这很了不起，因为很多人总觉得“我必须擅长所有事，我必须成为愿景、战略、执行、协作等所有方面的专家”。我认为承认自己的优缺点并加倍发挥长处才是关键。

[00:15:47] Farhan Thawar

English:

Yeah, it sounds funny, but me and you talked about it that a framework I wrote down, which I tweet out, me writing that down changed how I picked jobs forever, right? Because I had this lull after my first job in between where I was trying to figure out why nothing felt as exciting as my first job. And it turns out that it took me to sit down and be like, what do I actually care about? And people can get confused. I get confused all the time, by the way, by things that are not on my framework. So for example, a good one, title, company, money, all these things can confuse you because you could have somebody say recruiter messages you and says, "Hey, by the way, here's a new job and here's the compensation." You're like, "Oh my God, this is exciting." And if you don't have a written down framework of the things you actually care about, it's very hard to be distracted. So very hard not to be distracted. You get distracted by that. So instead, I look at the framework and go, does this align with my framework? Right?

(00:16:43):

Actually to the point of, I actually sent my framework to a recruiter one time and I said, "Hey, this thing," because they kept going back and forth to me and I go, "Hey, this doesn't align with my framework." So it really saves me time from not being distracted, but it also forced me to think about every year I can reevaluate what I'm doing and look at the framework and say, "Is this true to my values?" Now, my wife will say this, that I'm like a robot. When I realize that my framework is being violated, I will resign instantly and I've done that before. So without even having another job or anything, I just go like, oh, my framework is being violated and then resign. There's this thing where I know what I enjoy working on, and that framework helps me find it. And so I encourage everyone, anybody looking for a job I always say, "Write down a framework. You can use mine as inspiration, but figure out what you care about and make sure that what you're working on lines up with that."

中文翻译:

是的，听起来很有趣。我写过一个框架（我发过推特），它彻底改变了我选择工作的方式。在我第一份工作之后，我经历了一段低谷期，我试图弄清楚为什么之后的工作都没有第一份那么让我兴奋。结果发现，我需要坐下来思考：我到底在乎什么？人很容易被迷惑，我也经常被框架之外的东西迷惑。比如：头衔、公司名气、薪水，这些都会让你分心。猎头给你发消息说：“嘿，这有个新职位，薪水很高。”你会觉得：“天呐，太棒了。”如果你没有一个写下来的、关于你真正在乎什么的框架，你很难不被这些东西带跑。所以现在，我会看框架：这符合我的框架吗？

(00:16:43):

事实上，有一次我直接把我的框架发给了一个猎头，我说：“嘿，这个职位不符合我的框架。”这真的帮我节省了时间，避免了分心。它也迫使我每年重新评估：我正在做的事是否忠于我的价值观？我妻子说我像个机器人，一旦我发现我的框架被违背了，我会立刻辞职，我以前就这样做过。甚至在没找好下家的情况下，我直接说：“噢，我的框架被违背了”，然后辞职。我知道我喜欢做什么，框架帮我找到它。所以我鼓励每个找工作的人：“写下一个框架。你可以参考我的，但要弄清楚你在乎什么，并确保你的工作与之匹配。”

[00:17:37] Lenny Rachitsky

English:

And this framework is the questions to ask about where to go work. That's your framework. Okay, cool. We'll link to that so folks can check it out. The example of you resigning when it didn't meet your framework, is that a story you're up for sharing? Is there something to learn from that?

中文翻译:

这个框架就是关于“去哪里工作”要问的问题。好的，我们会附上链接供大家参考。关于你因为框架被违背而辞职的例子，你愿意分享一下吗？有什么值得学习的教训吗？

[00:17:50] Farhan Thawar

English:

Yeah, sure. So this happened when I was at my previous, a few companies ago, we were running a mobile company called Xtreme Labs. That was the one that Chamath was the major investor in and so we worked with him directly. And what I realized was... And so that company was amazing. We worked on it for many years. It was a mobile app development company. We got to work on mobile apps for the biggest brands in the world, Facebook, Twitter, Instagram, Vine, the NFL, NBA, Bloomberg, Slack, you name it we worked on those mobile apps, and this is right when the iPhone and Android were really gaining steam in '09 to 2013 era. And then we eventually got acquired by Pivotal. And over time, my role at Pivotal, Pivotal, and Pivotal Labs changed from, hey, I was running the biggest office in the world. I was running the biggest pair programming office.

(00:18:34):

I'm a big fan of pair programming to one in which we were really trying to attach consulting to the product. And I ended up being a field CTO, which really, I mean it was fun to learn about that world, but it was different than what I was doing. And so if I looked across my framework, it was violating all the things that I was trying to work. I wasn't working with the smartest people anymore. I was on IC. I wasn't learning as much as I could be learning. And so I wasn't on this and I wasn't having a lot of impact. I was like, oh, wait a sec, this is completely not aligned. And then I just told the team, "Hey, I plan on resigning." And that, by the way, led to great other things because I'm an investor in new companies that have spun out from there and it was a great experience. I'm just saying it at the time, lended me to say, "Hey, you're not actually focused on the right things."

中文翻译:

当然可以。这发生在我几家公司之前，当时我们经营一家名为 Xtreme Labs 的移动开发公司。Chamath 是主要投资者，我们直接和他合作。那家公司非常棒，我们做了很多年，是一家移动应用开发公司。我们为全球最大的品牌开发应用：Facebook、Twitter、Instagram、Vine、NFL、NBA、Bloomberg、Slack 等等，几乎你能想到的都有。那是 2009 到 2013 年，iPhone 和安卓正迅猛发展的时期。后来我们被 Pivotal 收购了。随着时间推移，我在 Pivotal 的角色发生了变化。原本我管理着全球最大的办公室，也是最大的结对编程（Pair Programming）办公室。

(00:18:34):

我是结对编程的忠实粉丝。但后来角色变成了试图将咨询业务挂钩到产品上。我最终成了一名 Field CTO（现场首席技术官），虽然了解那个领域很有趣，但和我原本做的事情完全不同。对照我的框架，它违背了我所有的初衷：我不再和最聪明的人一起工作，我成了一个独立贡献者（IC），我学不到我想学的东西，也没有太大的影响力。我意识到：“等一下，这完全不匹配。”于是我告诉团队：“嘿，我打算辞职。”顺便说一下，这后来带来了很多好事，因为我投资了从那里拆分出来的新公司，那是一段很棒的经历。我只是想说，在那一刻，框架让我意识到：“嘿，你现在的精力没放在正确的事情上。”

[00:19:20] Lenny Rachitsky

English:

I want to come back to Xtreme Labs. I know there's other stories there that are interesting, but first I want to talk about another theme of things that people often raised when I asked them about you. And this

one is intensity, and it's specifically creating intensity in your organization and the value of that, the power of that. I've seen that the way you describe this and I love is how do you expend more kilojoules per hour versus spend more hours on work. So talk about just why intensity, first of all, is so important to an organization.

中文翻译:

我想晚点再聊 Xtreme Labs，那里肯定有很多有趣的故事。但首先我想聊聊另一个主题，这也是我向别人打听你时经常被提到的：强度（Intensity）。具体来说，就是在组织中创造强度及其价值和力量。我非常喜欢你描述这一点的方式：如何“每小时消耗更多的千焦能量”，而不是“在工作上花费更多的小时”。请先谈谈，为什么强度对一个组织如此重要？

[00:19:53] Farhan Thawar

English:

Yeah. So I think there's a few things. One, I have this fundamental belief that one hour is one hour. It's the same hour. If you spend an hour or I spend an hour, it's the same time that goes by. And if I just expend more calories in that hour. Let's say we both work nine to five. If I can just get more done in the nine-to-five, we have both... The time has elapsed the same for us, but I just got more done. And that allows me, of course, then I'd be like, "Hey, I'm going to take my kid to soccer and do other stuff." We can still do the same things out of work, but during work, I just want to try to get as much done as possible during the time versus expanding the time and I can give you an example. I used to work at a company where it was like I worked 12 hours a day, but I was playing foosball in the middle of the day.

(00:20:37):

And then we'd go for a coffee break and you do these things. And of course, the time expanded to 12 hours versus trying to compress into that eight-hour day. And pair programming is a great example because, so it's such an intense activity. Two people on one machine, you can get so much done when two people are working together, not being distracted by the internet and distractions, and just focus on writing the solution to the problem at hand. And it's so tiring that usually when people switch on to pair programming, they sleep 10 to 12 hours a night the first few nights because it's so intense. You're working so hard. But for me, that intensity actually leads to extraordinary outcomes even if you don't have to put in more hours. I think most people, you probably hear this all the time, everyone says, "Oh yeah, work hard and do more hours when you're young, whatever."

(00:21:26):

I'm like, what if you just did more per minute? Like quickly get through things. I think there's another unintuitive fact is that people who are really good can actually output high-quality collateral quickly. So take a person who is good and extremely good, the extremely good person can actually get a lot of output in a short amount of time, and the person who's good might take longer. I think there's a time variance there that people don't think about. So you can not drop the quality too much, but get the time down by 2X, 3X, right? Parkinson's Law of scale instead of, if I give you an hour to do something, a really good person can get a high-quality output in one hour.

中文翻译:

是的，我觉得有几点。第一，我有一个基本信念：一小时就是一小时。无论你花一小时还是我花一小时，流逝的时间是一样的。如果我能在那一小时里消耗更多的热量——假设我们都从九点工作到五点，如果我能在九五期间完成更多工作，时间对我们来说流逝得一样快，但我产出更多。这让我之后能有时间带孩子去踢足球或做别的事。工作之外我们做的事一样，但在工作期间，我想尽可能多地完成任务，而不是拉长工作时间。举个例子：我以前在一家公司，每天工作 12 小时，但中午在玩桌上足球。

(00:20:37):

然后去喝咖啡休息，做这些杂事。结果时间被拉长到了 12 小时，而不是压缩进 8 小时里。结对编程（Pair Programming）就是一个绝佳的例子，因为它是一项极其高强度的活动。两个人共用一台机器，当两个人一起工作、不被互联网分心、只专注于解决眼前的问题时，效率极高。这非常累，通常刚开始尝试结对编程的人，头几天晚上都要睡 10 到 12 小时，因为太高强度了。你工作得非常努力。但对我来说，这种强度能带来非凡的结果，即便你不需要投入更多的小时数。我想大多数人常听到这种说法：“趁年轻多努力，多加班。”

(00:21:26):

我的想法是：如果你每分钟产出更多呢？快速处理事情。还有一个违反直觉的事实：真正优秀的人其实能快速产出高质量的成果。拿一个“优秀”的人和一个“极其优秀”的人相比，极其优秀的人能在短时间内完成大量产出，而优秀的人可能需要更久。人们往往忽略了这种时间上的差异。你可以在不降低太多质量的前提下，将时间缩短 2 倍、3 倍。这就是帕金森定律（Parkinson's Law）的反向应用：如果我给你一小时做某事，一个真正厉害的人能在一小时内交出高质量的答卷。

[00:22:30] Farhan Thawar

English:

It is the most underutilized management tool in engineering, bar none. It is just not used as much as it could be. So pair programming, for those who don't know, it's two people on one computer. So two keyboards, two mics, two monitors, but one computer, they work together and if it's remote, they use it. You can use a tool like tuple, which we use, and you can just remotely be on one computer, and you're totally right. The famous tweet about pair programming is, wait a sec, we have two engineers on one computer, won't they write half as much code? And the answer is, oh, no, no, they'll write even less than that because it's not about lines of code. The throughput limiter is not hands-on keyboard. It's not like we're both sitting there and the limiter is like us trying to get through the keystrokes onto the screen.

(00:23:12):

The limiter is where is the good elegant solution? How do we think through the problem and build the right solution for the problem at hand? Tobi famously built a lot of Shopify paired programming, and what he would do is he would actually set a timer and him and the CTO Cody would pair program for one hour. And if they did not finish the problem in one hour, they would delete all the code and they would keep the tests and they would start over. And then what their thinking was, if we were not able to articulate and write the code for this feature in one hour, we must be on the wrong design. We must be building the wrong thing and so they delete all the code, kept the tests, and then wrote it again. And sometimes he'd be over by one minute and he would still delete the code and start over because his thinking was the right elegant solution should be able to be written in one hour.

(00:24:03):

And so pair programming, I mean, that's an extreme version of it, but even at Pivotal Labs, if your pair was sick that day and you wrote a bunch of code, the strong version is your pair would come in the next day, delete all the code that you wrote, and then you'd write it again the next day. And again, what better time to rewrite code than right after you've written it because you now know the problem domain? And by the way, it sounds like a waste of time. It sounds like I'm just deleting code but the reason is that code lives a long time. Code is a liability and the right solution, the usually shorter lines, more elegant solution tends to appear after you've done a bunch of pathfinding. And the only way to do that pathfinding is start and then delete and then start and be like, oh, no.

(00:24:47):

Now I know. Delete. And it's super hard to delete, by the way, because we're humans and we have this sunk cost fallacy, so it's hard to delete. But if you can do that, you will actually land upon a much, much better solution. And of course, pair programming has high, high rates of learning because you're just sitting beside... You're not only whether it's tuple or remote or directly, you learn keystrokes and you learn how somebody thinks about a problem. You go back and forth on the talking, and yes, you will write probably less code, but you will move faster along the path of delivering value for your customers than you would if you did it on your own. And there's all these studies that show happiness is higher, knowledge transfer is higher, less silos, intensity is higher, all the things. And at a price of 20% or something of what you would normally do.

(00:25:33):

The analogy I have is the underhanded free throw in basketball, statistically known to sink more baskets, but looks dumb and nobody does it. Literally, Shaquille O'Neal, I'm not that big a basketball fan, but I read this about Shaquille O'Neal, who's a Hall of Famer, and they said, "Why don't you throw underhand?" Because he was notoriously bad at free throws and he goes, "It looks dumb." Even though he's paid millions of dollars a year to do this thing. It looks dumb, doesn't want to do it.

中文翻译:

它是工程管理中最被低估的工具，没有之一。它的使用频率远低于它应有的水平。对于不知道的人，结对编程就是两个人共用一台电脑：两个键盘、两个麦克风、两个显示器，但只有一台主机。他们一起工作，如果是远程，可以使用像 Tuple 这样的工具（我们就在用）。你完全说对了，关于结对编程最著名的吐槽是：“等一下，两个工程师用一台电脑，产出的代码量不是只有一半吗？”答案是：“噢不，产出的代码量甚至更少。”因为重点不在于代码行数。吞吐量的瓶颈不在于“手在键盘上敲的速度”，不是我们坐在那儿拼命敲字。

(00:23:12):

瓶颈在于“哪里才是优雅的解决方案？”我们如何思考问题并为当前问题构建正确的方案？Tobi（Shopify 创始人）当年开发 Shopify 时大量使用了结对编程。他会设一个一小时的计时器，和他当时的 CTO Cody 一起结对。如果一小时内没解决问题，他们会删掉所有代码，只保留测试用例，然后重头再来。他们的逻辑是：如果我们不能在一小时内清晰地表达并写出这个功能的代码，那说明我们的设计肯定是错的，我们做偏了。所以他们删掉代码，保留测试，重写。有时只超时了一分钟，他还是会删掉重来，因为他认为正确的优雅方案应该能在一小时内写完。

(00:24:03):

这是极端版本。但在 Pivotal Labs，如果你的搭档那天病了，你一个人写了一堆代码，最硬核的做法是：第二天搭档回来，把你写的代码全删了，你们俩再重新写一遍。其实，重写代码的最佳时机就是刚写完的时候，因为此时你对问题领域最了解。顺便说下，这听起来像在浪费时间，像是在瞎折腾。但原因是：代码会存在很久，代码其实是一种“负债”。而正确的、通常更短、更优雅的方案，往往是在你经历了一番探索（Pathfinding）后才会出现的。而探索的唯一方法就是开始、删除、再开始。

(00:24:47):

现在我懂了，删掉。顺便说下，删除代码超级难，因为我们是人类，有“沉没成本谬误”，很难割舍。但如果你能做到，你最终会得到一个好得多的方案。而且，结对编程的学习效率极高，因为你就坐在旁边——无论是通过 Tuple 还是面对面，你能学到对方的快捷键，学到对方思考问题的方式。你们不断交流，是的，代码行数变少了，但在为客户交付价值的道路上，你们比单打独斗跑得更快。所有研究都表明：幸福感更高、知识传递更快、打破信息孤岛、强度更高。代价仅仅是比平时多出约 20% 的成本。

(00:25:33):

我的类比是篮球里的“端尿盆”式罚球（Underhanded free throw）。统计学证明这种姿势命中率更高，但看起来很蠢，所以没人做。沙奎尔·奥尼尔（Shaquille O'Neal）是名人堂球员，但他罚球极烂。别人问他：“你

为什么不尝试端尿盆罚球？”他说：“那看起来太蠢了。”即便他每年拿着几千万美金去做这件事，他还是因为怕看起来蠢而不愿意做。

[00:26:15] Lenny Rachitsky

English:

On that note, so what percentage of Shopify do this? Is this how y'all operate?

中文翻译:

说到这，Shopify 有百分之多少的人在这么做？这是你们的标准运作方式吗？

[00:26:21] Farhan Thawar

English:

Yeah. So, Shopify, I mentioned that Toby and Cody did this at the beginning of Shopify. And the cool thing about pair programming is, and in my old world at XtremeLabs is that we knew exactly what to build, because we were building mobile apps that were almost like contract manufacturing. We're like, "We have an iOS version. Can we build an Android version?" So, we quickly were able to say, "Here's the spec. Go quick." Shopify is such a different company, right? We are a pathfinding company. We are trying to find the right thing to build. And so, pair programming may or may not make sense all the time. Like Pivotal and Xtreme, we were doing 40 hours a week.

(00:26:54):

Shopify is much more of a four to eight hour a week pair programming culture, where you're gathering together on a problem and saying, "Hey, let's pair for half a day, or let's pair every Wednesday." And we use that tool in our arsenal to move quickly down a path. But a lot of other time is spent pathfinding and trying to figure out what to build, and trying to convince field be like, "Hey, we're going to go down this path. Oh, now I know exactly."

(00:27:16):

And sometimes, by the way, 18 months later, we've now figured out all the things and that's the time we should delete everything and start over. And, that's something that we will do at that point. And so, you don't want to be pair programming for 18 months. You want to be wayfinding and pathfinding. And then go, "I see the matrix. Let me just delete everything and now build it." Because the learning is what you're going for. We have all the learning, now let's write the code.

中文翻译:

Shopify 在初创期 Tobi 和 Cody 确实是这么做的。结对编程在 Xtreme Labs 时期非常有效，因为那时我们很清楚要建什么，就像代工厂一样：“我们有 iOS 版了，能做个安卓版吗？”所以我们可以快速说：“这是规格说明，快写。”但 Shopify 完全不同，我们是一家“寻路”公司（Pathfinding company），我们在寻找该建什么。所以结对编程并不总是适用。在 Pivotal 和 Xtreme，我们每周做 40 小时。

(00:26:54):

在 Shopify，更多是每周 4 到 8 小时的结对编程文化。大家针对一个问题聚在一起说：“嘿，我们结对半天吧，或者每周三结对。”我们把它当作武器库里的一个工具，用来快速推进。但更多时间花在寻路、思考该建什么、说服大家走这条路。

(00:27:16):

顺便说下，有时 18 个月后，我们终于搞清楚了所有事情，那才是该删掉一切重来的时候。到那时我们会这么做。所以你不需要结对编程 18 个月。你需要先寻路、探索，然后说：“我看清本质了，现在删掉一切，重新构建。”因为学习才是目的。既然学到了，现在可以写代码了。

[00:27:39] Lenny Rachitsky

English:

Got it. So it's basically, when the code, you're pretty sure this is correct and it's really important segments of the code base pair program.

中文翻译:

明白了。所以基本上是当你确定代码逻辑正确，且是代码库中非常重要的部分时，才进行结对编程。

[00:27:49] Farhan Thawar

English:

Yeah. And then also we do a lot of pairing during an incident or a way to figure out together, work with somebody and say, "Hey, I'm not really sure and let's jump on a call together or jump on a tuple and go down the path." And say, "Let's figure out together what's going on."

中文翻译:

是的。此外，在处理线上事故（Incident）或需要共同排查问题时，我们也会大量结对。找个人说：“嘿，我不确定这里怎么回事，我们连个线或者上 Tuple 看看。”一起弄清楚到底发生了什么。

[00:28:01] Lenny Rachitsky

English:

I can't help but ask AI, how does that impact this way of working?

中文翻译:

我忍不住想问，AI 对这种工作方式有什么影响?

[00:28:06] Farhan Thawar

English:

So, AI is super interesting. What's happening right now with an AI copilot like GitHub Copilot is it is your pair programmer. So, you now can feel like you're pairing actually without another human. You can pair with the AI. And so, what's happening too is that you're seeing people use Whisper, like they're talking to Cursor, and they're talking through Whisper to say, "Okay, let's build a new React component that does this." And they're talking and then it's building. "Oh no, that's not what I meant. I meant doing this." So you can actually not even have to type, just using voice, go back and forth with your pair programmer.

(00:28:35):

I would say, that's amazing. I would still contend take that experience and add two humans together. So you've got an AI copilot and humans, because what's happening is generating code. And the two humans can look and say, "Oh, I know what it's trying to do." And, either delete the code, because you have

inspiration and write it yourself, or just take the suggestion and move it forward. But, I love today's world of AI copilots, because you never have to code loan on your own right? You never have to code alone. You can try a different language now, because the API and the syntax is much easier to pull forward. And so, all of those things are a win for engineering and a win for everybody who wants to build any software.

中文翻译:

AI 非常有趣。像 GitHub Copilot 这样的 AI 助手现在就是你的结对编程伙伴。你现在可以感觉到在没有另一个人的情况下进行结对。此外，人们开始使用 Whisper（语音识别），比如对着 Cursor（AI 编辑器）说话，通过 Whisper 说：“好，建一个能实现这个功能的 React 组件。”你一边说，它一边写。你说：“噢不，我不是那个意思，我是说这样做。”你甚至不需要打字，只需语音就能和你的“结对伙伴”来回沟通。

(00:28:35):

这很神奇。但我依然认为，最好的体验是“两个人类 + AI 助手”。因为 AI 生成代码后，两个人类可以一起看：“噢，我知道它想干嘛。”然后要么根据灵感删掉重写，要么接受建议继续推进。我喜欢现在的 AI 助手时代，因为你再也不用孤独地写代码了。你现在可以尝试不同的编程语言，因为 API 和语法更容易掌握。这对工程界和所有想开发软件的人来说都是巨大的胜利。

[00:29:14] Lenny Rachitsky

English:

That makes a lot of sense. Basically, everyone's going to be a pair programmer in the future. Okay, I want to come back to what else you have done at Shopify to create intensity. And I think, again, it's important to highlight the intensity is meant to, "How do we get more done in the time we have and then go home?" Not just work all day every day, weekends kind of intensity.

中文翻译:

很有道理。基本上未来每个人都会是结对程序员。好，我想回到你在 Shopify 还做了哪些事来创造强度。我想再次强调，这种强度是为了“如何在有限时间内完成更多工作然后回家”，而不是那种没完没了加班、牺牲周末的强度。

[00:29:35] Farhan Thawar

English:

Yeah. So we have a few things that we have going for us, right? So one, we have this tool called GSD, which stands for get shit done, which you've probably heard from maybe talking to other Shopify folk, which is this notion of weekly updates to the whole company on what's happening. Again, Parkinson's law at scale. If you ask people every week, they want to show progress every single week. So that's one way I talked about pair programming as well. The other thing we do as a company is, we used to have twice a year was our cadence, Black Friday, Cyber Monday, or we had an event in the summer. And now, we do six-week reviews. So, teams have this notion of every six weeks actually coming together and walking through the roadmap, the resourcing, and what they're working on with their immediate leadership, but then also, with Toby.

(00:30:18):

And so, what's cool about that, and by the way, it's a huge time investment, right? We all get into a room, it's happening tomorrow. So Tuesday, Wednesday, Thursday, a bunch of us will be in the office together and we're going to go through every project in the company, and we're going to talk about the project, the resourcing, how it's going, and we're going to make changes. And again, that creates intensity,

because you want to show what has been done, what have you learned since the last six-week review. And, we find six weeks is a very good cadence, because it's short enough that you can remember the context and it's long enough... Six weeks is long enough, especially if you have, let's say, a team of a dozen engineers, you can do a lot. And not only that, you can do a lot in a day, but this is a check-in point.

(00:30:59):

And what I've noticed too with intensity is, let's say, we get a review and there's some feedback we get in that review. We don't wait until the next six-week review, right? The next day we are building things, we are iterating, we are tagging people. And then, by the next week's we're like, "Here's the trajectory." Right? Actually, I want to get that Elon shirt made, "What have you done this week?" Because, Parkinson's law is real. It sounds funny, but I keep bringing this up, but whatever time you allot to something will be the time it takes, right?

(00:31:26):

So, if you're doing something monumental, I don't know, you're doing a reorg or something, right? You can do it the slow way. "Let's sit down, and plan, and roll it out." And it's probably six months in most companies. Shopify, this is a week or two. You sit down, you're like, "Hey, this is the bones of it. Let's bring some more people and think." But then, it's going to start leaking. So we just launch it. Right? And, we do the same thing with lots of things. We just try to move more quickly and get out of the... We don't do change management, we just land it, and then go, "Hey everyone, it's a volatile company. This is what's happening. But this is how we get things done quickly." And then, move on with our lives.

中文翻译:

是的。我们有几大法宝。第一，我们有一个叫 GSD 的工具，意思是“把事搞定”（Get Shit Done），你可能听其他 Shopify 员工提过。它的核心是每周向全公司更新进展。这又是帕金森定律的大规模应用：如果你要求每周更新，大家就会想每周都展示进度。这是除了结对编程外的另一种方式。第二，我们改变了节奏。以前是一年两次大节点（黑五网一和夏季活动），现在我们做“六周回顾”。每隔六周，团队会聚在一起，与直接领导以及 Tobi 一起梳理路线图、资源分配和工作进展。

(00:30:18):

这很酷，顺便说下，这投入了巨大的时间成本。我们明天就要开这个会，周二、周三、周四，我们一群人会待在办公室里，过一遍公司的每一个项目，讨论项目情况、资源、进度并做出调整。这创造了强度，因为你想展示自上次回顾以来你做了什么、学到了什么。我们发现六周是一个非常好的节奏：它足够短，让你能记住上下文；又足够长，特别是对于一个十来人的工程师团队，六周能做很多事。

(00:30:59):

关于强度我还注意到一点：如果在回顾中得到了反馈，我们不会等到下一个六周才行动。第二天我们就开始构建、迭代、标记相关人员。到下周，我们就能展示新的轨迹。其实我想印一件埃隆·马斯克那样的 T 恤，上面写着：“这周你干了什么？”因为帕金森定律是真实存在的。听起来好笑，但我反复强调：你给一件事分配多少时间，它就会花掉多少时间。

(00:31:26):

如果你要做一件大事，比如组织架构调整（Reorg），你可以慢慢来：坐下来、计划、逐步推行。在大多数公司这得花半年。在 Shopify，这只要一两周。坐下来，定好骨架，找人商量，然后直接发布。我们很多事都这么干。我们不搞复杂的“变革管理”（Change management），我们直接落地，然后告诉大家：“嘿各位，这是一家多变的公司，情况就是这样，但这是我们快速搞定事情的方式。”然后继续前进。

[00:32:00] Lenny Rachitsky

English:

Wow, there's so much there. I have been through the six-month reorgs. And, I think that context you just shared of, "We're at a volatile company, we're changing things. It's not going to be smooth, but we think this is for the best." It's just the culture of Shopify. It sounds like, "We want to keep moving fast. We know this isn't going to be the smoothest thing, but we just know this better to make the change at this point versus wait."

中文翻译:

哇，信息量很大。我经历过那种长达六个月的架构调整。你刚才分享的背景——“我们是一家多变的公司，我们在改变。过程不会一帆风顺，但我们认为这是最好的选择”——这听起来就是 Shopify 的文化。听起来像是：“我们要保持高速。我们知道这不会很完美，但我们知道现在就改比等着要好。”

[00:32:25] Farhan Thawar

English:

It's how Toby increased the resiliency in the company. He would walk around in the old days when we had a data center and just unplugged machines, right?

中文翻译:

这就是 Tobi 提高公司韧性的方式。早年我们还有数据中心时，他会走过去直接拔掉机器电源，对吧?

[00:32:32] Lenny Rachitsky

English:

Chaos monkey.

中文翻译:

混乱猴子（Chaos Monkey，一种测试系统稳定性的工具）。

[00:32:33] Farhan Thawar

English:

Yeah, chaos monkey. You're right. But that actually works, because it just says, "Hey, by the way, shit's going to break. And so, let's be resilient to that." And so, same thing here. "Hey, by the way, someone's going to move your cheese. It's fine. We are here to create more entrepreneurs in the world. We're not here to have a six-month change management roadmap. And, that will just actually hurt the speed at which we can deliver value to merchants."

中文翻译:

没错，混乱猴子。这确实有效，因为它在传达：“嘿，顺便说下，东西总会坏的，所以我们要有韧性。”这里也一样：“嘿，有人要动你的奶酪了，没关系。我们的存在是为了在世界上创造更多企业家，而不是为了搞一个六个月的变革管理路线图。那只会损害我们为商家交付价值的速度。”

[00:32:52] Lenny Rachitsky

English:

So, on all the things you shared, so there's weekly updates. So the weekly updates are each person shares what they're working on for the week. Is that the idea?

中文翻译:

关于你分享的这些，每周更新是每个人分享他们这周在做什么吗？是这个意思吗？

[00:32:59] Farhan Thawar

English:

Each project.

中文翻译:

是每个项目。

[00:33:01] Lenny Rachitsky

English:

Each project.

中文翻译:

每个项目。

[00:33:02] Farhan Thawar

English:

So, each project, it has an update, it might have a video of, "Here's the experience." It'll have a obviously a bunch of writing on what's changed since last time. We have a process called OK1 and OK2, which is like, OK1 is typically at the director level where they're like, "Okay, I'm aligned with the direction that this is going at." Or, "I'm not aligned." And they can make changes. Then, when it goes to OK2, it's typically the VP level of the area, who's now looking to say, "Okay. What you're working on actually aligns with the overall architecture. But by the way, have you looked at this context? Maybe you haven't seen this, this is happening, or the industry." And so, you're trying to align at that level. And then, again, like I mentioned, every six weeks we go through with Toby, and he's an intense guy himself.

(00:33:40):

And so, a lot of it is like, "Hey, why is this taking so long? Are we overthinking it? Are we not trying to move forward on this thing, because we're blocked on something? Is there some piece of infra?" Actually, I'll give you a good example. In one of the reviews from last time, there was an interesting AI problem we were trying to solve with LLMs that required us to have a very large output context window. And, most of the LLMs today have a very small output context window. But in the review, right, we have shared Slack channels with all the LLM folks, right? I messaged in the open AI channel, I messaged in the Gemini channel, whatever. And, within an hour we had increased the context on a bunch of major models and we were able to move forward through the thing, just because I asked.

(00:34:29):

And so, that's an example. It didn't take another six week review, but it increased the intensity, because the team was like, "Oh, we were blocked because we thought we had to now chunk up this data and do

this thing because we had smaller output context and we thought we could do a big input context, but we'd have to do this caching." And, it was like this whole thing. I'm like, "Well, did we just ask them if we get bigger?" And then, they were like, "Oh, we don't have this as undocumented, but we'll just enable it right now for Shopify." And so, that created the intensity of the team to be like, "Oh, we can now quickly get unblocked." So that's the example of just moving quick and trying to just, again, ask a dumb question. I'm like, "This is probably not possible, but..." And then, they came back and said, "Oh yeah, we can do that."

中文翻译:

每个项目都有更新，可能包含一段视频展示“这是目前的体验”，显然还有文字说明自上次以来的变化。我们有一个叫 OK1 和 OK2 的流程。OK1 通常在总监（Director）级别，他们会看：“好，我认同这个方向”或者“我不认同”，然后做出调整。到了 OK2，通常是该领域的副总裁（VP）级别，他们会看：“好，你做的东西符合整体架构。但顺便问下，你考虑过这个背景吗？也许你没看到这个情况，或者行业动态。”这是在那个层面达成一致。然后，正如我提到的，每六周和 Tobi 过一遍，他本人也是个非常有强度的人。

(00:33:40):

所以很多对话是：“嘿，为什么这花了这么久？我们是不是想多了？我们是不是因为被什么东西卡住了而没法推进？是不是缺什么基础设施？”举个好例子：上次回顾中，有一个有趣的 AI 问题，我们需要大语言模型（LLM）有非常大的输出上下文窗口。而目前大多数 LLM 的输出窗口很小。但在回顾现场，我们有和所有 LLM 厂商共享的 Slack 频道，对吧？我直接在 OpenAI 频道、Gemini 频道发消息。不到一小时，他们就为我们调大了几个主要模型的上下文限制，我们直接就能推进了，仅仅是因为我问了一句。

(00:34:29):

这就是一个例子。它不需要等到下一个六周回顾，但它增加了强度。因为团队原本觉得：“噢，我们被卡住了，因为输出窗口小，我们得把数据切片，还得做缓存。”搞得非常复杂。我说：“那我们问问他们能不能调大点？”对方回：“噢，虽然文档没写，但我们可以现在就为 Shopify 开启这个功能。”这让团队瞬间感受到了强度：“噢，我们可以这么快就解除阻塞。”这就是快速行动、敢于问“蠢问题”的例子。我说：“这可能不太可能，但是……”然后他们回：“噢可以，我们能办到。”

[00:35:03] Lenny Rachitsky

English:

That's a great example. And, as you're describing the ways that you create intensity and velocity within Shopify, it's interesting that what you're listing is a bunch of meetings and check ins, which to most people would feel like, "Why do we need so many..." There's all this like, "Less meetings." And I know you guys famously cancel all your meetings and that's a whole thing we can talk about. But, it's interesting that more check ins and regular check ins allow you to move faster. I imagine it's partly because it just makes sure you're not working on things that are unnecessary, and dumb, and not going to be used. And it's just continue to refine, these are actually the most important.

中文翻译:

这是个很棒的例子。当你描述在 Shopify 创造强度和速度的方法时，有趣的是你列举了一堆会议和检查点。对大多数人来说，这听起来像是：“为什么需要这么多会……”现在到处都在提倡“减少会议”。我知道你们曾大张旗鼓地取消了所有会议，这我们可以待会儿聊。但有趣的是，更多的、定期的检查反而让你们跑得更快。我猜部分原因是因为这能确保你们没在做那些没用的、愚蠢的、最后不会被使用的东西，而是不断精炼出最重要的任务。

[00:35:38] Farhan Thawar

English:

I mean, it's a combination of trust but verify, right? Because don't forget, the goal of the check-in is not for you to be like, "Ha, ha, I caught you not doing your work." It's not like the Dilbert boss, "Hey, did you do your thing?" Right? Even when I look at the Elon text, which is like, "Hey, what did you get done this week?" It wasn't to try to catch Parag in a, "You didn't do anything or you did a bunch of useless stuff." It was hopefully to pair on the problem, meaning, when I ask somebody, "Hey, did we move forward on this LLM project, because we now have this larger context window?" And then, they came back and said, "Oh, here's what we learned." So then, I can then look at the answer and say, "Oh, so now, have you thought of this? Have you tried..." It's a way to pair on the problem.

(00:36:18):

So, we have this word, everybody talks about micromanagement as a word, and we don't actually think it's a dirty word at Shopify, but the reason we don't think it's a dirty word is because it's not just, again, Dilbert boss saying, "Where did you do the thing?" It's like, "Hey, can I work on this problem with you? And if I work on this problem with you, I got to see where you are pretty often, and then give you advice, or you're going to share context with me, because I'm not in the work every day." To then come back and say, "Oh, based on what I know and what you know, can we move this in this direction? Maybe that's better for merchants."

(00:36:50):

I don't want to overuse the pairing paradigm, but it is really much like, "Can I pair with you?" And I learned this actually very early in my Shopify tenure, because Toby would have these one-on-ones with me and I'd be like, "Toby, you don't have to waste your time, man. You hired me. I got this." Well, he goes, "Oh, you misunderstand why you're here. We are here to work on problems together." And I was like, "Oh, I didn't even think..." I thought he hired me to take problems away. He hired me to work on problems with me. That's completely different than what I thought.

中文翻译:

这其实是“信任但要核实”的结合。别忘了，检查的目的不是为了说：“哈哈，抓到你没干活了。”不是像《呆伯特》（Dilbert）里的老板问：“嘿，你那事做了吗？”即便我看埃隆·马斯克发的短信问：“这周你搞定了什么？”也不是为了抓 Parag（推特前 CEO）的辫子说你没干活。而是为了“针对问题进行结对”。意思是，当我问某人：“嘿，既然有了更大的上下文窗口，LLM 项目推进了吗？”他们回答后，我可以接着说：“噢，那你想过这个吗？试过那个吗……”这是一种共同解决问题的方式。

(00:36:18):

所以，大家常把“微观管理”（Micromanagement）当成贬义词，但在 Shopify 我们不觉得它是脏话。原因在于，它不是那种老板问“东西在哪”的微观管理，而是：“嘿，我能和你一起解决这个问题吗？如果我要和你一起解决，我就得经常看到你的进度，然后给你建议，或者你跟我同步背景，因为我不是每天都在一线。”然后我们结合彼此的信息，看能不能把方向往商家更有利的地方带。

(00:36:50):

我不想过度使用“结对”这个词，但这真的很像“我能和你结对吗？”我在刚进 Shopify 时就学到了这一点。Tobi 跟我做 1 对 1 面谈，我说：“Tobi，你不用在我身上浪费时间，你雇了我，我能搞定。”他回道：“噢，你误解了你为什么在这。我们在这儿是为了‘一起’解决问题。”我当时心想：“噢，我完全没想到……”我以为他雇我是为了让我把问题拿走，结果他雇我是为了和我一起解决问题。这和我之前的想法完全不同。

[00:37:18] Lenny Rachitsky

English:

I love that. Okay. One thing you mentioned is meeting thing. For people that don't know what you all did with meetings, I think it might be worth just sharing that briefly, because it's awesome and something a lot of companies can learn from.

中文翻译:

我喜欢这个观点。好，你提到了会议的事。对于那些不知道你们对会议做了什么的人，我觉得值得简要分享一下，因为那太酷了，很多公司都可以借鉴。

[00:37:28] Farhan Thawar

English:

Yeah, sure. Actually, the funny story about the meeting Armageddon, is that, I was messaging Toby prior to me starting at Shopify about meeting Armageddon. And so, I actually think I had a little hand in him doing this before I got to Shopify. I was like, "Hey, have you seen..." I think it was Dropbox, "Have you seen a Dropbox is doing and Meetingageddon? And so, he was like, "This is super interesting." This is years before I started. So, I think it's funny that it ended up being a real thing. But here's what we do. Once a year at a random time, we will delete all recurring meetings that have more than two people, so not one-on- ones, and are internal people only, so not interviews or external partner meetings. And then, we have a two-week moratorium where you're not allowed to add a reoccurring meeting. You can have a regular meeting, but not a reoccurring meeting.

(00:38:11):

And the idea is that there's a lot of inertia behind a recurring meeting. It just always is there, and you know it's coming up, and it's hard to delete, because you're like, "Oh yeah, we talk about this thing every week." And so, what we do is we just do a meeting reset. And, I think, yeah, it's just called chaos monkey and the admins go in and just delete everything. Now what's cool about it is, it forces you to rethink, "Do we need a recurring meeting? Or do we just need one meeting? Or do we need a different cadence?" That's one thing. The other thing is it frees up so much crafter time, right?

(00:38:42):

One of the stats I track across engineering is how many hours are individual contributors in meetings per week? And we noticed that after we did... We did two things by the way. And I have a spicy second one for this, but the first one was we deleted meetings. And the second thing we did was we moved a lot of our Slack into Facebook workplace, which I'll talk about. Those are the only two things we did. And we saw a huge decrease in the amount of time crafters were in meetings. And then, we saw all kinds of other productivity enhancements, because they were able to have that flow time and work on things.

(00:39:12):

So, we're at something like three hours of meetings per week for an individual contributor at Shopify, which is phenomenal. Three hours a week is amazing. I think managers is not that bad. I think I tweeted this. I think it's six or seven hours per week. That's not bad at all, in order to get aligned. And then, all the rest of the time is work time.

中文翻译:

好。关于“会议大清洗”（Meeting Armageddon）有个趣事：在我加入 Shopify 之前，我就在给 Tobi 发消息聊这个。我觉得在他实施这件事之前，我可能推了一把。我当时问：“嘿，你看到 Dropbox 做的‘会议大清洗’了吗？”他说：“这非常有意思。”那是好几年前的事了。所以最后它成了现实，我觉得挺好玩的。我们的做法是：每年随机找个时间，我们会删除所有超过两人的“周期性会议”（Recurring meetings）。不包括 1 对 1

面谈，且仅限内部人员（不包括面试或外部合作伙伴会议）。然后有两周的“禁令期”，期间不允许添加任何周期性会议。你可以开临时会，但不能开周期性的。

(00:38:11):

核心逻辑是周期性会议有巨大的惯性。它总在那儿，你知道它要来，而且很难删掉，因为你会觉得：“噢，我们每周都要聊这个。”所以我们做一次“会议重置”。管理员直接进去把它们全删了。酷的地方在于，它强迫你重新思考：“我们真的需要每周开会吗？还是开一次就行？或者换个节奏？”这不仅释放了大量“匠人”（Crafter，指开发者）的时间。

(00:38:42):

我在工程团队追踪的一个指标是：个人贡献者（IC）每周开会多少小时？我们发现做了两件事后（第二件事比较劲爆），会议时间大幅下降。第一件是删会议，第二件是把大量的 Slack 沟通转移到了 Facebook Workplace（我会细说）。就这两件事，我们看到匠人们开会的时间骤减。随后生产力大幅提升，因为他们有了“心流时间”来专注工作。

(00:39:12):

现在，Shopify 的个人贡献者每周大约只有 3 小时会议，这简直不可思议。每周 3 小时太棒了。经理们的情况也不错，我发过推特，大约是每周 6 到 7 小时。为了达成一致，这个时长完全可以接受。剩下的所有时间都是工作时间。

[00:39:29] Lenny Rachitsky

English:

And how many hours was it before Meetingageddon and Workplace?

中文翻译:

在“会议大清洗”和启用 Workplace 之前是多少小时？

[00:39:33] Farhan Thawar

English:

Yeah, you're asking me a good question. I have to go look and see. But, it came down by something like 50 or 60%. It was something like five or six hours for individual contributors and came down to three. And then, the managers, I think it was 10 that came down to 6, something like that. But it was a huge difference. And the only two things, like I mentioned, were one was the Meetingageddon, the other one was like this, and I can talk about this. This is a-

中文翻译:

问得好，我得去查查数据。但大约下降了 50% 到 60%。个人贡献者以前大约是 5 到 6 小时，降到了 3 小时。经理以前大约是 10 小时，降到了 6 小时左右。差别非常大。正如我提到的，只有两件事：一是会议大清洗，二是这个……我可以聊聊这个。

[00:39:53] Lenny Rachitsky

English:

Yes, let's talk about this.

中文翻译:

好，聊聊这个。

[00:39:54] Farhan Thawar

English:

... Yeah. So, I mean, I love Slack. It's the IM tool. Everybody uses it. But it can for sure cause distraction. And so, what we did was we moved all announcement information. So, anytime you're sending a status update or large group announcement, we moved all of that to Facebook Meta Workplace, like Facebook for work basically, which is now being deprecated. So, we'll have to figure that out. But, it just moved all this stuff to a ML feed that you can consume differently than you would Slack, because Slack is like I message you and you see an alert and all this stuff, versus Workplace is like, "Oh, I want to go and consume content from the company, and get updates, and share updates." And so, that reduced a lot of distraction as well. And so, I'd love to figure out what the next tool for us is, but it is probably something more like a river of information that I can dip my toe into, versus IM and chat everywhere.

中文翻译:

我喜欢 Slack，它是每个人都在用的即时通讯工具。但它确实会造成分心。所以我们的做法是：把所有的“通知类信息”移走。任何时候你要发状态更新或大群组通知，我们都把它移到 Facebook (Meta) Workplace（基本上就是企业版 Facebook，虽然现在它要停运了，我们得另想办法）。它把这些内容变成了一种基于机器学习的“信息流”（Feed），你的消费方式和 Slack 完全不同。Slack 是我发消息给你，你看到提醒，然后被打断；而 Workplace 是：“噢，我现在想去看看公司的内容、获取更新、分享动态。”这减少了大量干扰。我很想知道我们的下一个工具会是什么，但它可能更像是一条“信息之河”，我可以偶尔把脚伸进去蘸一下，而不是到处都是即时消息和聊天。

[00:40:45] Lenny Rachitsky

English:

That is super interesting. So, specifically, things that are just updates where you don't need a discussion, you almost want to discourage a discussion.

中文翻译:

这非常有趣。所以，特别是那些纯更新、不需要讨论的事情，你甚至想通过这种方式减少讨论。

[00:40:50] Farhan Thawar

English:

Yeah, I mean, it has the commenting, but it's not the same tool. And, by the way, Slack is amazing, we use it. It's just that for this thing, it wasn't working for us. And so, we wanted to move that somewhere else.

中文翻译:

是的，它虽然有评论功能，但性质不同。顺便说下，Slack 很棒，我们也在用。只是针对“通知”这件事，它不适合我们，所以我们想把它移到别处。

[00:41:03] Lenny Rachitsky

English:

I feel like, the more I dig into the Shopify way of working, the more fun stuff I never expected emerges. I'm curious what else is going to emerge. So we've been talking about ways that y'all, and you specifically, have created intensity, especially in the engineering organization. And then, you've also shared just broadly Shopify. What else is on that list? What else have you found helps create more kilojoules per hour?

中文翻译:

我觉得我越挖掘 Shopify 的工作方式，意想不到的趣事就越多。我很好奇还有什么。我们聊了你如何创造强度，特别是在工程团队，你也分享了 Shopify 整体的情况。名单上还有什么？还有什么能帮助创造“每小时更多千焦产出”？

[00:41:24] Farhan Thawar

English:

Yeah, so again, I think there's nothing, I would say again, start from the beginning, there's nothing more than pair programming, because literally you can't do anything else but be on your computer. So pair programming is the number one. I will say, the weekly cadence helps a lot, right, which you mentioned. Again, which is part of GSD, which is sharing the updates, and then the six-week reviews, that does a lot. On the other side, we also have a lot of metrics and alerts that help us see when potentially things might be happening in the system that can allow us to be like, "Hey, wait a sec, there's too many things going on of this type. We probably have to sit back and reset and figure out what's going on."

(00:41:56):

So one thing that happened, for example, was we started seeing that it was taking a lot of time to develop in our admin, like engineers at Shopify developing in the admin. So we called, what's called, a code yellow, which is before code red. But code yellow is this idea of like, "Hey, we're going to call a code yellow on the admin." We want to make sure that the developer experience inside Shopify is really good. It should be easy to start up the repo. It should be easy to make changes, it should be easy to see the changes. And so, those are the things that, again, we can create intensity, because this code yellow allows the champion to tap anyone on the shoulder and say, "Stop what you're doing and come help this thing." Which is an infrastructure layer thing. And by building out this infrastructure, it allows you to go fast. It takes longer to build infrastructure, but it makes you go fast forever afterwards, right?

(00:42:41):

I'll actually give you an example of one of these. So, we in 2020, 2021, the heyday of pandemic, obviously, there was a crypto summer again and crypto was going nuts. And we were sitting back and saying, "Wow, a lot of our merchants are now asking for NFT gating." NFT gating, which is, "If you have the token, you can now go into the storefront and see my products. You can see my prices and you can check it out, but only if you have the token." And, we were getting a lot of demand from merchants to be like, "We want to do this. We want to sell an NFT. And we want our buyers to have the NFT to have this great experience." And we're like, "We agree. We want you to be able to do whatever you want. And so, we want to build this for you too."

(00:43:18):

And, sitting with Toby, he's like, "You guys are thinking about it wrong." He goes, "How long would it take to build NFT gating?" I'm like, "I don't know. Two, three weeks." He goes, "Now, how long would it take to build a platform layer, which exposes APIs so anyone could build NFT gating in one hour?" I'm like, "I don't know. Two, three months." He goes, "Do that." He goes, "Because you don't know what they're going to build on top of the platform." NFT gating is one thing, one use case, but if you spend the time to

build out the infrastructure layer, he calls it putting gas in the tank, if you put the gas in the tank, people could drive on that gas for a long time going forward. And so, he goes, "I always want you to think about..." And the key part was, "What can you build so anyone could build this in one hour?" Right?

(00:44:00):

So, he does this thing to us all the time where he goes, "Oh, this should only be..." He'll say it and people get the wrong thing. He'll say, "Oh, you could write this in a day." And what he means is, "What has to be true so that you could write this in a day? What infrastructure do you need?" And, he actually develops this way. He will write code against an API that doesn't exist, because he's like, "You know what should exist here? This API." He'll write the code, he'll go back and forth and refine the client on the server, and then he'll go, "This is correct, the correct client code. Now, let me go implement the server code."

(00:44:31):

And this is notion of building things as infrastructure that sound slower today, because it's going to take... It's two, three months, instead of two or three weeks. But after that, the things that people built on top, right, were so easy to build, there were so many more scenarios that were emerged. It's just a different way of thinking about software. And, it's intensity in a different way, it's intensity around building this infrastructure layer, which we want to build quickly, but takes two or three months, in this case, but then can get everyone building on top of our infra in a much, much quicker way. And of course, who knows what can flourish from there?

中文翻译:

是的。首先，我还是要强调，没有什么比结对编程更能创造强度了，因为你除了盯着电脑干活什么也干不了。所以结对编程是第一位的。其次，你提到的每周节奏非常有帮助，包括 GSD 更新和六周回顾。另一方面，我们有很多指标和警报，能帮我们发现系统中潜在的问题。比如：“嘿，等一下，这类事情发生得太频繁了，我们得停下来重置一下，搞清楚怎么回事。”

(00:41:56):

举个例子：我们发现工程师在 Shopify 管理后台（Admin）进行开发的效率变慢了。于是我们启动了所谓的“黄色代码”（Code Yellow），它仅次于“红色代码”。黄色代码的意思是：“嘿，我们要对管理后台启动黄色代码。”我们要确保 Shopify 内部的开发者体验非常棒：启动仓库要快、修改要容易、预览修改要方便。这也能创造强度，因为黄色代码允许负责人拍拍任何人的肩膀说：“放下你手头的事，来帮我搞这个。”这属于基础设施层面的工作。虽然构建基础设施花的时间更长，但它能让你以后永远跑得更快。

(00:42:41):

再举个例子。2020、2021 年疫情期间，加密货币非常火。很多商家要求“NFT 门控”（NFT gating）功能，即：“如果你有这个代币，你才能进店看我的产品和价格。”商家需求很旺盛，我们也想为他们实现。

(00:43:18):

我和 Tobi 坐在一起，他说：“你们想错了。做一个 NFT 门控要多久？”我说：“不知道，两三周吧。”他问：“那如果做一个平台层，暴露 API，让‘任何人’都能在一小时内做出 NFT 门控，要多久？”我说：“大概两三个月。”他说：“那就做这个。因为你不知道别人会在平台上建什么。”NFT 门控只是一个用例，但如果你花时间建好基础设施层（他称之为“往油箱里加油”），人们以后就能靠这些油跑很久。他的核心思想是：“你能建出什么东西，让别人能在一小时内搞定这个功能？”

(00:44:00):

他经常对我们说：“噢，这应该只要……”大家有时会误解。他说：“这你应该一天就能写完。”他的真实意思是：“要满足什么条件，你才能在一天内写完？你需要什么基础设施？”他自己就是这么开发的：他会针对一个尚不存在的 API 写代码，因为他觉得：“这里‘应该’存在这样一个 API。”他先写客户端代码，反复推敲，然后说：“好，客户端代码这样写是对的，现在我去实现服务端代码。”

(00:44:31):

这就是把事情当作“基础设施”来构建的理念。今天听起来更慢（两三个月 vs 两三周），但之后别人在上面构建东西会变得极其容易，会涌现出更多场景。这是一种不同的软件思考方式。这也是一种强度，是关于快速构建基础设施层的强度，虽然它本身耗时，但它能让所有人之后在我们的基座上以快得多的速度构建。谁知道以后会开出什么样的花朵呢？

[00:45:06] Lenny Rachitsky

English:

It's interesting and it makes sense that so much of the way you all think is about building the best possible platform, versus the, necessarily, best possible point solution for someone. And it also explains why you spend so much time in crafting really great code and pair programming, because again, it's a platform for other people to build stuff on. So, I think, a lot of this is very useful, especially for platform businesses.

中文翻译:

这很有趣。你们的思考方式更多是关于构建最好的平台，而不是仅仅为某人提供最好的点对点解决方案。这也解释了为什么你们花这么多时间打磨代码和进行结对编程，因为这是一个供他人构建的平台。这对平台型业务非常有用。

[00:45:29] Farhan Thawar

English:

No, exactly. And actually, you're making me think of a stat. So, last year, maybe a tangent, but I tweeted out that GitHub Copilot has written over 1 million lines of code for Shopify. And people are like, "Oh my god." And it got picked up. And everyone's talking about it. And I go, "I don't know why is everybody getting so crazy, because what I want to see is GitHub Copilot deleting 1 million lines of code." That's when you know we're actually at this point where this is close to an engineer. Right? And so, we famously have deleted millions of lines of code this year, because we were trying to focus on, again, the sunk cost and rebuilding things elegantly, or you don't need this anymore and rebuilding. And I even tweeted out, I think someone said, "Oh, Shopify is in the top 10 Ruby code bases in the world." And I said, "I never want to see us on this list again." We shouldn't be gunning for number one, we should be gunning for number 100, right? We want to be not on this list. Right? Someone else can take the crown.

中文翻译:

没错。你让我想起一个数据。去年我发过一条推特（可能有点跑题），说 GitHub Copilot 已经为 Shopify 写了超过 100 万行代码。大家都很震惊，到处都在传。我说：“我不明白大家为什么这么激动，我真正想看到的是 GitHub Copilot ‘删除’ 100 万行代码。”那才说明它接近一个真正的工程师了。今年我们删除了数百万行代码，因为我们想专注于摆脱沉没成本、优雅地重构，或者删掉不再需要的东西。有人说：“噢，Shopify 是全球前十大 Ruby 代码库之一。”我回道：“我再也不想在这个榜单上看到我们了。”我们不该争第一，我们该争第 100 名。我们想从榜单上消失，让别人去拿那个冠军吧。

[00:46:22] Lenny Rachitsky

English:

[Sponsor Message for Vanta - Omitted]

... Wait, talk about more about this. So, there's been a drive to delete code and simplify. What's behind that? What's going on there?

中文翻译:

[赞助商广告 - 略]

……等等，再多聊聊这个。你们有一种删除代码和简化的驱动力，背后的原因是什么？到底是怎么运作的？

[00:47:22] Farhan Thawar

English:

Yeah, so there's a few things, right? One is, the more context you can fit in your head around a code base. And you can never really fit all of Shopify in your head, because it's a big complicated set of tools we give to merchants. But, the more you can simplify, the much easier everything becomes, resiliency, performance, reliability, maintainability, all the illities become much, much, much easier when the code base is simple. Now, all you need really is the mandate of like, "Oh, well, let's look at this code. And, if I could start this today, would I really build this thing? Or do I now have enough domain expertise to say, 'Oh no, this is the right solution?' So can I delete start over and build this more easily?" And now, everything else becomes easy to build on top of.

(00:48:04):

And so, routinely, we have a delete code club, we have hack days, which happens two or three times a year, where there's always one team that is focused on deleting code. They even have a manual, "Here's how to find things to delete." And, it's amazing. We almost always delete. I don't know if this is good or bad. We can always almost find a million plus lines of code delete, which is insane. But, at the same time, I applaud the teams for going after the cruft and the code base. And everything gets easier, right? Codelets loads faster. It's easier to understand.

(00:48:34):

This is why when I look at GitHub stats, you shouldn't really look at... I think Google put out and said, "Oh, 25% of all code is now written by AI." I'm like, "Where's the delete? Where is the 25% of all code is deleted by AI?" Right? This is where we have to start thinking about it. Because, the right code is never the volumous lines of code metric. It's always something else. It's always elegance. And that's where we have to think about. So, it is something that, as part of us being long-term infrastructure thinking, we really do care about that.

中文翻译:

有几点原因。第一，你能在大脑里装下的代码库上下文越多越好。你永远没法把整个 Shopify 装进脑子里，因为它太复杂了。但你越简化，一切都会变得越容易：韧性、性能、可靠性、可维护性，当代码库简单时，所有这些“性”都会变得容易得多。现在你需要的只是一个指令：“好，让我们看看这段代码。如果我今天重新开始，我还会这么写吗？或者我现在是否有足够的领域知识说：‘不，这才是正确的方案’？那我能不能删掉重来，用更简单的方式构建？”这样，在上面构建其他东西也会变容易。

(00:48:04):

所以，我们定期有“删代码俱乐部”，每年两三次的黑客日（Hack Days）也总有一个团队专注于删代码。他们甚至有一本手册：“如何寻找可删除的内容”。这很神奇，我们几乎总能找到超过 100 万行可以删除的代码，这很疯狂。但同时，我为那些清理代码库糟粕的团队鼓掌。一切都变快了：加载更快、更容易理解。

(00:48:34):

这就是为什么我看 GitHub 统计数据时，不觉得……谷歌曾说“25% 的代码现在由 AI 编写”，我的反应是：“删掉的代码呢？什么时候 AI 能删掉 25% 的代码？”这才是我们需要思考的。因为正确的代码永远不是靠“行数”来衡量的，而是靠优雅。作为长期基础设施思考者，我们非常看重这一点。

[00:49:06] Lenny Rachitsky

English:

I love this, in part, because it connects to the topic we're talking about, which is speed, velocity, intensity, smaller code base, cleaner, better code base allows you to move faster. I used to be an engineer actually. But, both my engineering brain and my PM brain, I love the idea of killing stuff that's useless, fixing, making code cleaner, and better, and more durable. In reality, very hard for companies to prioritize this thing. Is there anything you found that helps you do that? You mentioned hack days and weeks are one part of that. Probably helps that Toby's an engineer and he understands the value of this stuff. But I guess, for folks that want to do this more, any advice?

中文翻译:

我喜欢这个观点，因为它和我们聊的主题——速度、强度——紧密相连。更小、更干净、更好的代码库能让你跑得更快。我以前也是工程师，无论是我的工程师大脑还是 PM（产品经理）大脑，都非常喜欢删掉没用的东西、让代码更简洁耐用。但在现实中，公司很难把这件事排在优先级。除了黑客日，你还发现有什么能帮助推动这件事吗？Tobi 本身是工程师肯定有帮助，但对其他人有什么建议吗？

[00:49:42] Farhan Thawar

English:

Yeah. So actually, when we're building something, we think of it in one of three buckets. We're like, "Are you building an experiment, a feature, or infrastructure?" And, once you bucket things, you can say, "Oh, it's an experiment." You're like, "Cool. This is not infra. This is like, we're trying something to learn." And, by the way, that might turn into an experiment or infra, but it starts off as an experiment. Now, if you're building a feature, a feature is basically you're taking advantage of an existing piece of infra, right? So, token gating is the example I gave. If you could now build that in one hour, you would probably say, "Oh, we have the right infra below it."

(00:50:14):

But if you did what Toby does, which is, he's like, "Here's the infra I wish existed. Here's the feature. The feature might be quick to build, but now I have to go and build the infra." You're now slotting yourself into infra land, which is like, that could take longer, but you're now enabling it for a bunch of use cases. You don't have to think about it at once, because you may have people using your API in a different way.

(00:50:32):

So, I think you have to slot yourself. Now, how do people get into this mode? It is super, super hard. And, I would say, Toby is the secret sauce here, because he pushes us to think about things as infra almost all the time. I mean, one of the things that annoys me the most probably is that I'll always come to him and say, "Hey, we can do A or B." And he looks at me and he goes, "You know what I'm going to say, right?" I'm like, "You're going to say go back and generate more options." Because he doesn't like those. "I don't like A or B. Come back when you have something else." Right? Actually, maybe I'll tell you a little anecdote. He has a story where he says, "There are unlimited amount of wrong options for any problem. There's probably 10,000 right options, but everybody stops at the first right one, instead of what you should be spending all your time on, because the options that don't work, you're not going to spend time on. But

you have to figure out which of the 10,000 options is the right one and spend time in what are all these right options? Don't just stop at the first one."

(00:51:24):

And so, when I come to and say, "A or B." I'm picking two of the 10,000. And he's like, "That's not what I... Go back and generate more options, because those are not the optimal ones." So, he is quite the philosopher on these things. And it does really change the way the company works, because he'll push you on these things. And then, we over time learn to spot the same patterns. And I learn to push my team on infra, and deleting code, and making things simple, because by the way, who doesn't want to get free stuff? Right? Free performance, free, easy to navigate code base, free maintainability, free resiliency,, because now, we went and did the hard work of deleting. It is hard. But that goes back to the beginning, right? Choose the hard thing. Don't just build the feature-

中文翻译:

是的。实际上，当我们构建东西时，会把它放进三个桶之一：“实验、功能、还是基础设施？”一旦分类，你就知道：“噢，这是个实验。”那它就不是基础设施，我们只是为了学习。它以后可能变成功能或基础设施，但开始只是实验。如果你在做一个“功能”，你其实是在利用现有的基础设施。比如我说的代币门控，如果你能一小时做完，说明底层基础设施是对的。

(00:50:14):

但如果你像 Tobi 那样做，他会说：“这是我希望存在的基础设施，这是功能。功能可能很快，但我得先去建基础设施。”这时你就进入了“基础设施领域”，这可能更久，但你为未来的一堆用例铺了路。你不需要一次性想全，因为别人会以不同的方式使用你的 API。

(00:50:32):

所以你得给自己定位。如何进入这种模式？这超级难。Tobi 是这里的“秘密酱汁”，因为他几乎总是逼我们从基础设施的角度思考。最让我“苦恼”的一点是，我去找他说：“嘿，我们可以选 A 或 B。”他看着我说：“你知道我要说什么，对吧？”我说：“你会让我回去想出更多选项。”因为他不满足于 A 或 B。他有个理论：“任何问题都有无限个错误的选项。可能也有 1 万个正确的选项，但大家往往在看到第一个正确选项时就停下了。你应该把时间花在寻找那 1 万个正确选项中最好的那个，而不是停在第一个。”

(00:51:24):

所以当我给出 A 或 B 时，我只是从 1 万个里挑了两个。他说：“这不是我想要的，回去想更多，因为这两个不是最优的。”他在这些事上很有哲学思想。这确实改变了公司的运作方式，因为他会不断推你。久而久之，我们也学会了识别这些模式。我也开始推我的团队去做基础设施、删代码、简化。因为谁不想要“免费”的好处呢？免费的性能、免费的易读性、免费的可维护性，这都是因为我们做了删除代码这种“难事”。这又回到了开头：选择难的事，不要只做功能，要让功能变得容易构建。

[00:52:06] Lenny Rachitsky

English:

I feel like there's just more and more good stuff. What else is there that might be helpful to folks while you're thinking about it? And interesting, I was reading your tweet where you shared a lot of this advice, and you mentioned this briefly, but I think it's important with pair programming, one of the benefits is there's no multitasking. You're not checking Twitter, Slack while you're working because you're there being watched. And I could see why that is more productive just innately.

中文翻译:

我觉得好东西越来越多。还有什么对大家有帮助的吗？我读过你分享这些建议的推特，你简要提到了结对编程的一个好处：没有多任务处理。你不会在工作时刷推特或 Slack，因为有人在旁边看着你。我能理解为什么这天生就更高产。

[00:52:31] Farhan Thawar

English:

Oh no, for sure. Again, like underhand or free throws not only looks dumb, it feels dumb. People don't... They feel like they're wasting time sitting beside somebody and being like, "Well, I could be on my computer doing this thing." But together they are building a machine. Do you ever read the Undoing Project, which is about Amos Tversky and Daniel Kahneman, the famous philosopher.

(00:52:54):

Lenny Rachitsky: By Michael Lewis, I think.

Farhan Thawar: Behavioral economist. Michael Lewis. Exactly. And he said the famous line was "Alone, we're okay, but together, we're a genius." Right? That's a pair programmer. That's like two people. You're like, "Ah, we're okay. But together, we're a genius." And that's exactly what pair programming is. And hopefully me plus an LLM is a genius as well, but that's the genesis of that thinking. So I would say another thing that helps us create intensity is demo culture. So as part of the GSD updates, hopefully we encourage people to share high fidelity updates, which is not just imagery, but actually a demo. One of the things that can go wrong with just their screenshots is you don't really get the full experience. You can't tell how slow things are or whatever, but with a demo, so you can put a link. We have this tool called Spin, which is an internal development environment like a cloud dev environment where you can say, "Hey, click here to try this on Spin, and then you can try it and you can see how it works."

(00:53:52):

Or they say, "Turn on a beta flag in your own store and then try it." And so this short circuits a lot of misunderstandings, because you're like "I'm going to try it." And you're not waiting until the end, especially with a beta flag. You're like, "Hey, it's in my store. I just realized that I went in and now this page takes 20 seconds to load. Is that what you expect?" You're like, "Oh, we didn't find this use case." You're going learn that much more quickly.

(00:54:13):

And that creates, again, intensity on the fidelity of the feedback you're getting because famously, some of our PM team will create a friction log. They'll be like, "I am walking.." They'll just create a screen share, create a video and go, "I'm walking through. I turn on the beta flag, I'm walking through this experience. Here's my feedback on the experience." And you're getting this high fidelity throughput coming back to you that you're like, "Okay, let's fix these things for next week's iteration and then share another beta flag and say, 'Okay, try it now. Try it now. Try it now.'" And so you're not debating about status reports, you're kind of debating about the experience.

中文翻译:

噢当然。再说一次，就像端尿盆罚球，不仅看起来蠢，感觉也蠢。人们会觉得坐在别人旁边是在浪费时间：“我自己在那台电脑上也能做。”但两个人在一起是在构建一台“机器”。你读过《倒带生命》（The Undoing Project）吗？讲的是阿莫斯·特沃斯基和丹尼尔·卡尼曼。

(00:52:54):

Lenny Rachitsky: 迈克尔·刘易斯写的。

Farhan Thawar: 没错，行为经济学家。书里有一句名言：“单独一个人，我们还行；但在一起，我们就是天才。”这就是结对编程。两个人在一起就是天才。希望“我 + LLM”也是天才，这就是这种思维的起源。另一件创造强度的事是“演示文化”（Demo culture）。在 GSD 更新中，我们鼓励大家分享高保真的更新，不只是截图，而是真正的演示。截图的问题是你无法获得完整体验，看不出有多慢。但通过演示，你可以放一个链接。我们有一个叫 Spin 的工具，是一个云端开发环境，你可以说：“嘿，点这里在 Spin 上试试。”

(00:53:52):

或者说：“在你的商店里开启这个 Beta 标志（Beta flag）然后试试。”这能消除很多误解，因为你可以直接上手。你不需要等到最后，尤其是有了 Beta 标志，你可以说：“嘿，在我的店里试了，我发现进这个页面要 20 秒，这是预期的吗？”你会说：“噢，我们没考虑到这个场景。”你会学得快得多。

(00:54:13):

这又为反馈的保真度创造了强度。我们有些 PM 团队会制作“摩擦日志”（Friction log）：他们录屏并配音说：“我正在操作，开启了 Beta 标志，这是我的体验反馈。”这种高保真的信息流传回给你，你会说：“好，下周迭代把这些修了，然后再发个 Beta 标志说：‘现在再试试’。”这样你们讨论的不是进度报告，而是真实的产品体验。

[00:55:42] Farhan Thawar

English:

And having the leaders pair with those people on those problems, not just checking in, but actually pairing with them on the problems that they're facing. So you get both the crafters who are working on the stuff and the leaders who may have broader context working together to kind of unblock.

中文翻译:

让领导者与员工一起针对这些问题进行“结对”，不只是检查进度，而是真正一起面对他们遇到的困难。这样，一线工作的匠人和拥有更广背景的领导者就能共同协作，扫除障碍。

[00:56:46] Lenny Rachitsky

English:

I love it. I just wrote a note down of just how you set up your teams for success. Oh, just avoid distractions. So I think the pair programming helps the workspace, workplace shift from Slack helps. I think you're also very firm on their working hours. You basically don't let...

中文翻译:

太棒了。我记下了你们如何让团队获得成功：避免分心。结对编程有帮助，从 Slack 转移到 Workplace 也有帮助。我觉得你在工作时间上也很严格，你基本上不让……

[00:57:04] Farhan Thawar

English:

No, not really. Yeah. We're not super firm on working hours from that perspective, but we do have a lot of people on East Coast time zones. A lot of stuff happens then, but we do have people all over the world. But I did mention we do have the check-ins and the six-week reviews on the cadence. So that six week cycle does give you a little bit of the, what did you get done and are you blocked mentality. And you can expect coming in a couple of times and being like, "Hey, we didn't get a lot done being unblocked." For

you to change your approach to go, "Okay, I don't want to go to another review where we didn't get a lot done." So what am I doing this time to make sure we unlock a lot of progress? And that check-in can give you that ammo to be like, "Let's do this this time."

中文翻译:

不，其实不是。我们对工作时间没那么死板，但我们确实有很多员工在东部时间，所以很多事在那时发生。不过我们员工遍布全球。我提到过检查点和六周回顾的节奏。这种六周循环会给你一种“你完成了什么”和“你是否被卡住”的心理暗示。如果你连续几次在回顾时说“我们没进展”，你就会想改变方法：“好，我不想下次开会还是没进展。”所以这次我要怎么做才能确保突破？这种检查点能给你动力。

[00:57:44] Lenny Rachitsky

English:

And you're also remote first as a company. Which I think is especially cool. Now a lot of companies are going back to not remote. You guys are staying remote. Why do you guys decide to do that? What have you seen as a big benefit of that?

中文翻译:

你们也是一家“远程优先”的公司，这特别酷。现在很多公司都在要求回办公室，你们却坚持远程。为什么这么决定？你看到的巨大好处是什么？

[00:57:55] Farhan Thawar

English:

Yeah, so we have this remote, so I like to call it 90% remote or 95% remote because we have these intentional IRL experiences. So every summer, we just started last year. Sorry, this year. We're doing this thing called Shopify Summit where we bring the whole company together, get together and it's a combination of talks plus hack days and it's a come together experience like food and parties and bands. And it's a super fun way to re-energize and rebuild your trust battery over time. And then we have this thing called bursts, which is, "Hey, you want to work on a problem? You need to prototype, you need to hack." People can just say, "Hey, I'm starting a burst. We're going to have five people. We're all going to go to Ottawa or Toronto or Montreal or somewhere else and we're going to talk about this problem and get together."

(00:58:38):

And so a combination of those two things mean throughout the year, you can recharge. We have the trust battery notion, which is how much trust can you have between people and it can deplete over time if you're remote. So then we have offices which are like, come in if you want. Like I mentioned, I come in once a week and now Toby moved to Toronto, so now I'm in three days a week. But it's like if you want to come in, you don't have to, right? Today, I work from home, but tomorrow, I'm going in. And that allows you, again, to have those random interactions and allow you to feel like, yeah, we're 90 plus percent remote. But I would say the main reason is we want to hire the best people in the world and those people can be anywhere and just happens to be that not all of them are near an office.

中文翻译:

是的，我喜欢称之为 90% 或 95% 远程，因为我们有刻意的“线下体验”（IRL）。每年夏天（从今年开始），我们会举办 Shopify Summit，把全公司聚在一起，有演讲、黑客日、美食、派对和乐队。这是重新注入活力、重

建“信任电池”的绝佳方式。此外，我们还有一种叫“突击”（Bursts）的形式：如果你想解决一个问题、做原型或黑客开发，你可以发起一个突击，五个人一起去渥太华、多伦多或蒙特利尔聚几天。

(00:58:38):

这两者的结合意味着你全年都可以“充电”。我们有“信任电池”的概念：人与人之间的信任在远程状态下会随时间损耗。我们也有办公室，你想来就来。我以前每周去一次，现在 Tobi 搬到了多伦多，我每周去三天。但这不是强制的。这让你能有随机的互动，同时保持 90% 以上的远程。核心原因是我们想雇佣全球最优秀的人，而这些人可能在任何地方，不一定都在办公室附近。

[01:00:06] Farhan Thawar

English:

And it can be strategic by the way. I've seen people use it as a... I'm pretty sure Toby says he starts everyone at 50% and then he gets to know you. And then I've seen people use it as the opposite, saying, "Hey, look, this team is hard charging. I'm going to start you at a hundred. Assume that you already have high trust, do the things, and only if you are doing things that are off alignment, does your trust battery deplete." So I've seen people use it, the terminology as a shortcut way to figure out how to work with somebody.

中文翻译:

顺便说下，这也可以是战略性的。Tobi 说他给每个人的初始信任值是 50%，然后慢慢了解你。我也见过相反的做法：给新员工 100% 的信任，假设你值得高度信任，只有当你做的事情偏离目标时，信任电池才会损耗。这成了一种快速确定如何与人共事的术语。

[01:03:11] Lenny Rachitsky

English:

Okay, let's talk about hiring. You have some really interesting takes on hiring. One that I've heard about is that you don't like the interview process. You kind of like to prefer not to interview and do something instead. Talk about that.

中文翻译:

好，聊聊招聘。你对招聘有一些非常有趣的见解。我听说你不喜欢传统的面试流程，你更倾向于不面试，而是做点别的。谈谈这个。

[01:03:28] Farhan Thawar

English:

Yeah. So throughout my career, what I've noticed is that, and I'm sure everybody, this is a dirty little secret, right? Interviews are not a good predictor of performance. We know this. We know this from studies. We know this. Everybody at their company knows this, where somebody interviewed well, wasn't as good in the job or the opposite, didn't interview well and then came in and was phenomenal. One example I have from my startup right before we came to Shopify was I hired two people for machine learning. One was a PhD, taught at the university, was like, oh my god, no brainer, was also recommended by an employee. We're like, "Oh my god, this person's going to be great." The other person was a dude I met at the coffee shop who had never had a software job but was just so interested in machine learning and person A, we let go within a few weeks because not a fit for our culture.

(01:04:12):

And person B was at our startup and is still at Shopify today and is a phenomenal machine learning engineer who literally at our Christmas party was like, "This is my first software job." We're like, "How?" It was just so cute. And we gave them both the shot. The key was I didn't use the resume in either way to bias. We brought them both in. We said, "Here's the environment." It was all pair programming at my startup. And so they pair program, and actually as an aside, I really believe in pair programming when I made people work in pair programming with my own money. I paid for two people to be on one computer. So that's how you know...

(01:04:51):

Lenny Rachitsky: Less than half the code.

Farhan Thawar: Yeah, exactly. Right. Less than half the code. But anyway, so pairing. And it was pretty clear after just a few weeks, I would say let's say up to three months is the amount of time I give people, that person A wasn't going to work out than person B was. So what I really like to do is use this race car analogy. If I told you, "Hey, I want to go hire the best race car driver," there's not really that many questions you could ask them except for put them in the car. And so the same thing happens with us is that of course, we have to do interviews, but we do really spend time in the 30, 60, 90 days to make sure that the thing that they are bringing actually lines up with what we need at Shopify. And you should also be transparent with people because if it's not a fit, it's actually good for them and you to figure that out as quickly as possible because they could be amazing somewhere else.

(01:06:21):

We mentioned the chaotic environment and fast moving environment Shopify is, that's not for everyone, but that's okay, right? We're not looking for... We've talked about that we want to be as the best 10,000 person company in the world. We're not looking for millions of people. We just need the best 10,000. And so if it's not a fit, it's in your interest and our interest to figure that out quickly so that you can go somewhere where you will be amazing and for us to have the people who will be amazing at Shopify. And so job trials, I'm a huge fan of, which leads me to intern programs, what a great interview process because you now have real work product from somebody for four months. They get to see what it's like to be at Shopify for four months. We get to see what it's like for them to be at Shopify for four months, and that can turn into a full-time gig.

中文翻译:

是的。在我的职业生涯中，我注意到一个公开的秘密：面试并不能很好地预测工作表现。研究证明了这一点，每家公司也都知道这一点：有人面试表现极佳，入职后却不行；或者反过来，面试一般，入职后却惊为天人。我在加入 Shopify 之前的创业公司有个例子：我招了两个做机器学习的人。一个是博士，在大学教书，还是员工推荐的，我们觉得“天呐，这肯定没问题”。另一个是我在咖啡馆遇到的哥们，从未做过软件工作，但对机器学习极其痴迷。结果，A 几周后就被辞退了，因为不符合文化。

(01:04:12):

而 B 留了下来，至今还在 Shopify，是一名极其出色的机器学习工程师。他在圣诞派对上说：“这是我的第一份软件工作。”我们都惊呆了。我们给了他们同样的机会。关键是我没有被简历误导。我把他们都招进来，让他们在全结对编程的环境下工作。顺便说下，我当时是自掏腰包付两个人的工资让他们共用一台电脑，可见我多相信结对编程。

(01:04:51):

Lenny Rachitsky: 产出不到一半的代码。

Farhan Thawar: 没错，不到一半。但重点是结对。几周后（我通常给三个月时间）就很清楚了，A 不行，B 很棒。所以我喜欢用“赛车手”类比：如果你想雇最好的赛车手，问再多问题也没用，直接让他上车开两圈。我们也一样，虽然要面试，但我们更看重入职后的 30、60、90 天，看他的能力是否真的匹配 Shopify。如果不匹配，尽早发现对双方都好，因为他可能在别处发光发热。

(01:06:21):

Shopify 的环境多变且高速，不适合所有人，这没关系。我们只想做全球最好的“一万人规模”的公司，我们不需要几百万人，只需要最好的一万人。所以“试岗”（Job trials）非常重要。这也引出了实习生计划：实习是最好的面试，因为你有四个月的真实工作产出。他们了解 Shopify，我们也了解他们，最后可以直接转正。

[01:09:46] Farhan Thawar

English:

Yeah, I would love to do it. It's hard to do with the volume resumes we get, but we do do it at scale with the internship program. So like 2025, we're going to hire a thousand interns, and that is going to give me a thousand job trials to pick the top X percent of those to come to Shopify full time.

中文翻译:

是的，我很想全面推行试岗，但面对海量的简历很难做到。不过我们在实习生计划中大规模实施了这一点。2025 年我们将招募 1000 名实习生，这相当于 1000 次试岗，让我们能从中挑选最顶尖的人才加入 Shopify。

[01:15:31] Lenny Rachitsky

English:

I want to talk about just one other topic real quick and then we'll wrap this up. And it's around XtremeLabs. So there's a bunch of stuff here. It feels like it's just like this tech mafia of Canada that a lot of incredible people emerged out of, and there's a whole bunch of stuff we can talk about. One fact I heard while you were there is you had a hundred reports, direct reports that reported to you.

(01:15:56):

Farhan Thawar: A 120.

Lenny Rachitsky: 120 direct reports feels like a complete nightmare to me. Tell me why you decided to do that, if you'd recommend that for other people.

中文翻译:

在结束前我想聊聊 Xtreme Labs。那里就像是加拿大的“技术黑手党”，走出了很多厉害的人物。我听说你在那里时有 100 个直属下属（Direct reports）。

(01:15:56):

Farhan Thawar: 是 120 个。

Lenny Rachitsky: 120 个直属下属对我来说简直是噩梦。告诉我你为什么这么决定，你会推荐别人这么做吗？

[01:16:05] Farhan Thawar

English:

Yeah, so what ended up happening there was we started off small. It was 10 people. When I got to XtremeLabs, I wasn't the founder, but I was very early on and I just had everyone report to me. And then as we grew, I just kept having those people report to me and I was trying to figure out, we talked about crafters and crafters paradise, this idea of people don't really... Their managers are useful, but I was trying to figure out could I solve the problems that they needed their manager for in other ways? So for example, what should I be working on? I was like, "Okay, well, we have product backlogs" or "I'm blocked on something," or "I need feedback on the product I'm building," or "I'm stuck on this technical problem." I tried to figure out ways to not have managers be the answers to those questions.

(01:16:48):

I'm like, "There should be another answer." And so pair programming really helps you get unblocked quickly because you have another person that you can bounce technical ideas off of. Having a product backlog can tell you what to work on. We had demos every week, demos internally, and then we had demos with the clients every week because we were a contract manufacturing for mobile. That gave them feedback on whether they were going in the right direction. We had set working hours, which made things super intense in the office. This is again like 2009 to 2013. So all these things didn't really need a manager. And what I realized was what the unblocking thing needed a manager. I'm blocked on this. Well, I said, okay, if... I had all these directors. I did two things famously. One, I had a lot of direct reports, and two, I did not do any scheduled one-on-ones because you can't have 120 direct reports and do a scheduled one-on-ones because you're never doing anything but one-on-ones.

(01:17:31):

So I said, "I'm going to be around a lot because I don't have a lot of one-on-ones. So we can do unscheduled one-on-ones." And what that means is if you are blocked, actually there's a famous picture where I had this weird cube desk where it was like a circle almost in the middle of the whole floor of engineers. And I was always there because I wasn't in a lot of meetings and people that were blocked, they could just come up to me. Actually, one cool thing about pair programming IRL is you can look across and just see if it's working or not. Because if two people are intensely on the computer, you know it's working. If one person is laying back or you're like, "It's not working," so you can just walk up to-

(01:18:00):

... them and be like, "What's happening?" But they would come and ask me questions and I can unlock them. Like, hey, we're blocked on this. We don't have this API. We need money for this machine, whatever. And so the unscheduled one-on-ones ended up being a real clarifying thing for me. Because I did scheduled one-on-ones my whole career and I realized after leaving Microsoft for three years, I'm like, where are all those one-on-ones useful once a week for three years, right? A hundred and fifty one-on-ones.

(01:18:26):

So the unscheduled ones were, though. I was like, when I knocked on my manager's door and said, "I have this problem, those were important." So that's what I created at Xtreme. And the 120 directors, it just grew over time. I just didn't think I needed managers. So I was like, let me unblock these people on another way. And we came up with other things to systems to unblock them that didn't require a manager. And I just also had a good memory. I knew exactly everyone's skills and compensation. I knew all that off the top of my head, so that helps.

(01:18:54):

The thing that I broke was actually, this is Chamath. He came in and became our biggest investors. He's obviously is a smart guy, but he said the right thing, which is not, this can never work, because then I would go into defense mode and explain to him why it would work. He just said to me, I'm not going to

debate with you whether this works or not, but will it work at 400 people? And I said, probably not. He said, okay, so let's change it.

中文翻译:

是的。当时的情况是，我们刚开始只有 10 个人。我不是创始人，但我是早期加入的，所有人直接向我汇报。随着公司壮大，我一直在思考：既然我们要打造“匠人天堂”，经理确实有用，但我能不能用其他方式解决原本需要经理解决的问题？比如：“我该做什么？”——看产品待办列表（Backlog）；“我被卡住了”或“我需要反馈”——通过结对编程和每周演示。我们每周都有内部演示和客户演示，这能告诉大家方向对不对。我们还有固定的工作时间，让办公室氛围非常紧凑。在 2009 到 2013 年间，这些事其实不需要经理。我意识到，唯一需要经理的是“解除阻塞”（Unblocking）。

(01:17:31):

所以我做了两件出名的事：一是拥有大量直属下属，二是不做任何“预约好的” 1 对 1 面谈。因为 120 个人，如果预约面谈，我这辈子就只能开会了。我说：“我不开预约会，所以我会有很多空闲时间，我们可以做‘非预约’的 1 对 1。”这意味着如果你被卡住了，你可以直接来找我。有一张著名的照片，我的办公桌在工程师楼层的正中央。因为我不怎么开会，我总在那儿。线下结对编程有个好处：我扫一眼就知道进展顺不顺，如果两个人都在盯着屏幕，那是顺的；如果有人瘫在椅子上，那肯定出问题了。

(01:18:00):

我会走过去问：“怎么了？”他们也会来问我：“我们缺这个 API，我们需要钱买这台机器。”这种非预约面谈对我来说非常高效。我在微软待了三年，每周都做预约面谈，后来我反思：那 150 次面谈真的都有用吗？其实只有当我敲开经理门说“我有问题”时，那次谈话才是最重要的。所以我在 Xtreme 创造了这种模式。120 个下属是随时间增长的，我当时觉得不需要经理层，而是通过系统来解除阻塞。而且我记性好，每个人的技能和薪水我都能背下来。

(01:18:54):

后来打破这个模式的是 Chamath。他成了我们最大的投资者，他非常聪明。他没说“这行不通”（那样我会反驳），他问：“这在 400 人的规模下还能行得通吗？”我说：“可能不行。”他说：“好，那我们改吧。”于是我们增加了一点结构，但我让几个总监每人带 40 个下属，依然保持非常扁平。

[01:20:39] Lenny Rachitsky

English:

Final question before we get to our very exciting lightning round. We have this recurring segments that I call Fail Corner where generally people come on these podcasts, they share all their successes. Here's all the things that I've done, right. Here's all the big wins. And everyone feels like, oh man, I wish I was always successful people. When in reality, everyone that comes on has failed many times. Is there a story of a failure in your career that you could share that helps people see that even folks like you fail, and maybe what you learned from that experience?

中文翻译:

在进入精彩的闪电轮之前，最后一个问题。我们有一个固定环节叫“失败角落”（Fail Corner）。通常嘉宾会分享成功经验，让听众觉得他们一直顺风顺水。但现实中每个人都失败过很多次。你能分享一个职业生涯中的失败故事吗？

[01:21:10] Farhan Thawar

English:

I have a few. I'll say one thing by the way, because I think I read this in Tim Ferriss's book or in the podcast where he said, create a failure resume and write everything down. And I would not recommend doing this because I did this and I got super... I'm like a high energy happy guy. I wrote down, I have a note on my phone called Failure Resume, and I wrote down all the times I failed and it is depressing. So I would not encourage people to do that, but I'm happy to tell you about a few instances.

(01:21:33):

So well, one is actually I've been laid off twice and people would not expect like, oh, I'm doing this thing and I've been laid off twice. And I think in both times it was the right thing, it was the right thing for the company, the right thing for me. And I kind of used that experience as a way to reevaluate and eventually came out with my framework of how I want to spend time. But that's maybe a different story.

(01:21:54):

I'll tell you about one at Shopify, the first week that I started, 2019, we were rebuilding our point of sale system, which now does billions of dollars of GMB. But back then we were like, well, let's rebuild it with a new UX and a new technology platform. And it was my first week and I'm the mobile guy, coming from Extreme Labs. They're like, should we build this in React Native, or should we build this natively on the mobile platforms? And so I went through this evaluation, spent a lot of time, blah, blah, blah, and I came back with a hedged solution, which is kind of dumb. I said, let's do iOS and Swift and let's do Android in React Native. And the reason I said that was I said, I want to learn about React Native and I think Android's the harder platform, so let's build that in React Native, but iOS on Swift because that guarantees us a product in market. And we didn't have any React Native apps in market at the time. A year later we launched the iOS version and it was a huge success. And we then spent another six months building the React Native version for Android and everything else. And we realized pretty quickly that React Native was the platform for the future. We were like, oh my God, this will allow us to have one platform. You could run it on the web as well, and we could use the React engineers from the web to work on it. It was a clear winner. By then, we had also launched a shop app, which is React Native.

(01:23:11):

And so we learned a lot about this. And I went back and I said, hey everybody, I made a huge mistake. We just spent a year building this thing. It's in market, but we're going to have to rewrite. We're going to have to rebase back onto this iOS version. And I think I burned 18 months of time with a hundred engineers, literally from the decision I made in the first week of joining Shopify, and Toby, when I went to Toby and I told him, I go, hey man, I think I made this mistake and we have to do this and it's going to cost us a hundred engineers, another six, whatever. And he looked at me and he goes, you should tell everyone this story. That's all he said. Not like, hey, bad, good. He goes, did you learn something? It was an epiphany for me, but he was like, this is a learning org, and I totally failed and I told everyone I failed and my mistake and everything else, but he goes, just tell everyone because he goes, do you know what mistake you made?

(01:23:57):

And first I was like, I don't think, like, what mistake. He's like, you didn't take. He goes, I will always come down harshly on people who do not take risks, and you did not take a risk in this case. But if you take a risk and it doesn't work out, you'll never get in trouble, because you took the risk and it was the right risk to take, but he goes, but you didn't take the risk.

中文翻译:

我有好几个。顺便说下，我读过蒂姆·费里斯的书，他说要写一份“失败简历”。我不建议这么做，因为我试过，我本来是个精力充沛、乐观的人，但我写完手机里那份“失败简历”后，感觉太压抑了。但我很乐意分享几个例子。

(01:21:33):

第一，我其实被裁员过两次。大家可能想不到，像我这样的人也被裁过。但我觉得那两次对公司和我个人都是正确的决定。那些经历让我重新评估，并最终总结出了我的“时间分配框架”。

(01:21:54):

讲一个在 Shopify 的。2019 年我刚入职第一周，我们要重构 POS（销售点）系统。当时我们要用新的 UX 和技术平台。我是做移动端出身的，大家问：“我们该用 React Native 还是原生开发？”我做了一番评估，最后给出了一个“对冲”方案（现在看来很蠢）：iOS 用 Swift 原生开发，安卓用 React Native。理由是我想学习 React Native，且我觉得安卓平台更难，所以用它试水；而 iOS 用原生是为了确保产品能按时上市。当时我们还没有任何 React Native 应用上线。一年后，iOS 版上线并大获成功。接着我们又花了半年做安卓版。很快我们意识到 React Native 才是未来：它能实现全平台统一，甚至能跑在 Web 上，还能复用 Web 端的 React 工程师。它是完胜者。

(01:23:11):

于是我回去跟大家说：“嘿各位，我犯了个大错。我们花了一年做的东西虽然上线了，但我们得重写，把 iOS 版也迁到 React Native 上。”我入职第一周的一个决定，浪费了 100 名工程师 18 个月的时间。我去找 Tobi 坦白，说我犯了错，得花大代价重来。他看着我说：“你应该把这个故事告诉所有人。”他没说好坏，只问我学到了什么。这对我来说是一次顿悟。他说这是一个学习型组织，我失败了，我承认了。他问：“你知道你错在哪吗？”

(01:23:57):

我当时还没反应过来。他说：“你没有冒险。我总是会严厉批评那些‘不冒险’的人。如果你冒了险但失败了，你不会有麻烦，因为那是正确的风险；但在这个案例中，你选择了保守，你没有冒险。”

[01:27:45] Lenny Rachitsky

English:

[Discussion about Farhan using Lenny's old performance review framework - Omitted for brevity]

... With that, Farhan, we have reached our very exciting lightning round. Are you ready?

中文翻译:

[关于 Farhan 使用 Lenny 以前写的绩效评估框架的讨论 - 略]

……那么，Farhan，我们进入了令人兴奋的闪电轮。准备好了吗？

[01:30:17] Farhan Thawar

English:

I'm ready.

中文翻译:

准备好了。

[01:30:18] Lenny Rachitsky

English:

What are two or three books that you have recommended most to other people?

中文翻译:

你向别人推荐最多的两三本书是什么?

[01:30:23] Farhan Thawar

English:

There's one. So Toby has an annoyingly long set of books that he recommends... But he recommended one to me that I think everyone should read right now called Manna, M-A-N-N-A, from Marshall Brain. It is a book, it's a book about AI. And I think the most interesting thing about it though is about a future in which the AI tells the humans what to do.

(01:31:05):

I think another book that I recommend to people... is Business Adventures from John Brooks... It just goes into a problem at such depth that if you can maintain your focus to get through the depths of each problem, you will just learn something.

中文翻译:

第一本是 Tobi 推荐给我的，Marshall Brain 写的《Manna》。这是一本关于 AI 的书，最有趣的是它描述了一个“AI 告诉人类该做什么”的未来。

第二本是 John Brooks 的《商业冒险》（Business Adventures）。它对问题的探讨非常有深度，如果你能保持专注读完，你会受益匪浅。

[01:32:11] Lenny Rachitsky

English:

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

中文翻译:

最近有什么喜欢的电影或电视剧吗?

[01:32:18] Farhan Thawar

English:

A recent one was Challengers, the tennis movie... And then one of my all time favorites is probably Halt and Catch Fire.

中文翻译:

最近看的是《挑战者》（Challengers），一部网球电影。我心目中的神剧是《奔腾年代》（Halt and Catch Fire），讲早期科技行业的。

[01:33:03] Farhan Thawar

English:

[Discussion about Meta Ray-Bans - Omitted]

... Do you have a favorite life motto that you find useful in work or in life?

中文翻译:

[关于 Meta Ray-Bans 智能眼镜的讨论 - 略]

……你有没有什么在工作或生活中觉得非常有用的座右铭?

[01:34:15] Farhan Thawar

English:

Okay, I do. And it's on a lot of my profile bios, and it is, everything you know is wrong. And the reason I like that one... is this notion of if all the knowledge you knew was incorrect, could you from first principles build up a view of the world?

中文翻译:

有的，就在我的社交账号简介里：**“你所知道的一切都是错的。”** 我喜欢它的原因是：如果假设你已知的所有知识都是错的，你能不能从“第一性原理”出发，重新构建对世界的看法?

[01:36:03] Lenny Rachitsky

English:

Final question. So you told me the story of this PhD you hired versus just a guy you met in a coffee shop. I read another similar story where you hired a waitress. Is that real? Okay, tell that story.

中文翻译:

最后一个问题。你讲了招博士和咖啡馆小哥的故事，我还听过一个你招了一名女服务员的故事。是真的吗？讲讲看。

[01:36:17] Farhan Thawar

English:

Yeah... we were at a restaurant and I saw a waitress doing a very, very good job... she was kind of doing a phenomenal job of organizing the entire crazy busy restaurant. And so in talking to her, I said, what do you do outside of this... Would you like to work at XtremeLabs? She first started off as our receptionist... then I brought her on as my admin and she became one of my recruiters. And I taught her how to recruit... The coolest thing about this is that she ended up taking over one of the HR functions for us... And now she's a director of HR at a company.

中文翻译:

是的。当时我们在一家餐厅，我看到一名女服务员表现得极其出色，她在疯狂忙碌的餐厅里把一切组织得井井有条。我问她：“你平时还做别的事吗？你想来 Xtreme Labs 工作吗？”她起初是我们的前台，后来成了我的行政助理，接着成了招聘人员。我教她如何招聘。最酷的是，她后来负责了我们的 HR 职能，现在她已经是某家公司的 HR 总监了。

[01:38:50] Lenny Rachitsky

English:

[Closing remarks and where to find Farhan - Omitted]

中文翻译:

[结尾致辞及联系方式 - 略]