medium.com /8vc-news/lessons-from-peter-thiel-b4fb0851f64e

Lessons from Peter Thiel

8VC

7-8 minutes

These lessons summarize what Joe Lonsdale learned from working over many years with Peter Thiel, a chairman and founder of Palantir. These are very much worth reading — they will change the way you think.

The nine lessons:

1. Divide reasoning into separate parts and clearly identify the most important factor.

If there's no single reason that can cause you to do something, you should think carefully about whether it's important or not. Oftentimes we'll want to do something, and we'll give multiple reasons for it without thinking hard about them. If you can't give a single reason that justifies doing something on its own, you should be very wary that you aren't exercising sufficient intellectual discipline.

2. Don't divide your attention: focusing on one thing yields increasing returns for each unit of effort.

At a micro level, an extra hour of focus on the current project has a much higher return than an hour on something new, or worse, 5 minutes each on 12 new things. Before you ever do something new, you should understand the opportunity cost vs. existing things. Don't rationalize that something you want to do is complementary when it's not!

At a macro level, understanding that applied effort has a convex output curve is a very useful discipline when considering new market areas. This convexity means that the opportunity cost of transferring resources from existing projects to new ones is high. Unless the new area is incredibly valuable, anything we can do to extend an existing convex curve is worth so much more.

3. In hiring, value intelligence highly. Like focus, intelligence yields increasing returns.

Like focus, intelligence has a convex output curve — the smartest people can be an order of magnitude more productive than others who are only somewhat less smart. The key in hiring is to value potential skill rather than currently existing skill — and potential skill is based on intelligence rather than training.

4. Obsess over perfection.

If you are designing something that a customer is going to use or that will represent us in public, it's not good enough unless it's flawless and extraordinary.

Especially in software, many situations have winner-take-all dynamics due to network effects and switching costs. Being the winner means being in the 99.99-percentile. A winner at the top takes nearly everything, and only a pittance goes to the others — so being 99.99-percentile is worth an order of magnitude or two more than being just 98-percentile. If it's 1am and you've already got something that is very good, this is why it's worth spending the next couple of hours to make it amazing.

5. Use small details as indicators to point to the larger truth — and be alert when they point to conclusions you don't like.

Despite the danger in this approach — indicators are not always right — more often than not you can derive a lot of wisdom from subtle indicators. How competent a single person is or how hard one person is working, or what a couple smart people think, might tell you a lot about a group before deciding whether to work with them. It's an important discipline to be open to any indicators that you find whether they confirm your biases or not. In fact, you should be open to them especially when they do not confirm your biases. Indicators can be about all sorts of things, such as about the importance of a contract, the effectiveness of our software in different scenarios, the opinions and biases of key decision makers, etc. Few people pay as close or as honest attention to them as they should.

6. Don't waste time talking about what you plan to think about; instead, work through it immediately.

Intellectual laziness can easily sneak up on you. If you are sitting there talking about problems you plan to solve later, there's a good chance you are being inefficient. Similarly, in GTD, you don't put off tasks that only take a couple minutes. In many cases, you can outline and solve or at least clarify any decision or problem you're confronted with in just a few minutes.

7. Take the time to listen to smart people with whom you disagree.

This is not easy to do. Over time, any firm will find that certain methods and biases tend to work very well, and the more successful it is, the more it will develop its own unique mindset. We couldn't succeed without the methods and principles that we learn over time, nor can we constantly re-articulate why they are the correct approach to everyone who comes along.

A sort of pride rightly develops around the successful structure, but this mindset cannot be allowed to ossify into an orthodoxy. So we have to use our judgment to seek out intelligent people who disagree with us — or even intelligent people who have simply taken a different approach — and be open about what we might learn from them. Despite his very strongly developed and out-of-the-mainstream views, Peter does this constantly, and it makes him extremely effective.

8. Be skeptical of any sort of external allies, and don't trust the execution of anyone outside Palantir.

In the abstract, this is because the incentives of the other company will not line up with ours, and even if they do for the moment, they no longer will once the situation changes. In specific, other companies don't have the same culture of execution that Palantir does, and we don't have the power to instill that culture in them. Very early on, Peter saved Palantir from a partnership that would likely have destroyed the company.

In practice, that skepticism has led us to force others to pay upfront as one of the ways of proving value, which has worked well. Ultimately, only our own execution can make us win.

9. Return to first principles and act quickly on your new conclusions.

It's very easy to take the world as it is, as opposed to envisioning it as you want it to be. For example, when re-designing a feature, one approach is to take what you have, and imagine small changes that will solve the problems with the feature. Instead, it's often instructive to imagine that you were working from a clean slate and design the feature from scratch. In either case, the right approach is almost always to re-use the existing codebase in your implementation — but starting from first principles frees your mind from your assumptions about implementation complexity and forces you to re-consider the assumptions that stand in the way of the best solution.

Just like questioning our own assumptions, returning to first principles and building the argument up from scratch is a very powerful intellectual device that helps us to recreate our models as you receive new information, and to uncover options and opportunities you'd otherwise miss. It is not surprising to see that the largest hedge fund in the world, now one of our customers, also emphasized this as their key methodology.