How to learn things at 1000x the speed • Pranay Prakash

7-9 minutes

September 1, 2019

Throughout my schooling and career, I've impressed people over and over again with how quickly I pick up things, especially **hard** things. For instance, exactly 4y ago I did my first ever interview, got a simple question and failed miserably after spending 4h on it and finally asking the interviewer what "recursion" was. Since then I've created core UI components at Facebook used by billions, worked on the language spec for GraphQL and Relay, ported an optimization made popular by Haskell to JS by writing an optimizing compiler, built an extension for VSCode that has half a million installs and has been listed on multiple top 10 articles, scaled a botnet of over 2,000 bots that made money from a popular trivia game, shipped to production an IPFS inspired decentralized file server and the fastest Serverless Docker runtime, *almost* completed a design minor while successfully creating a brand that's still recognized by every CS student at my alma mater, and finally started a company funded by Y Combinator and Pioneer.

I built the right skillset for a generalist, which is highly regarded in Silicon Valley. However, I think I'm actually really good at only one thing - learning. It's the only skill with which I've achieved the 10,000 hour rule. In this post I want to share what I've learnt.

The System

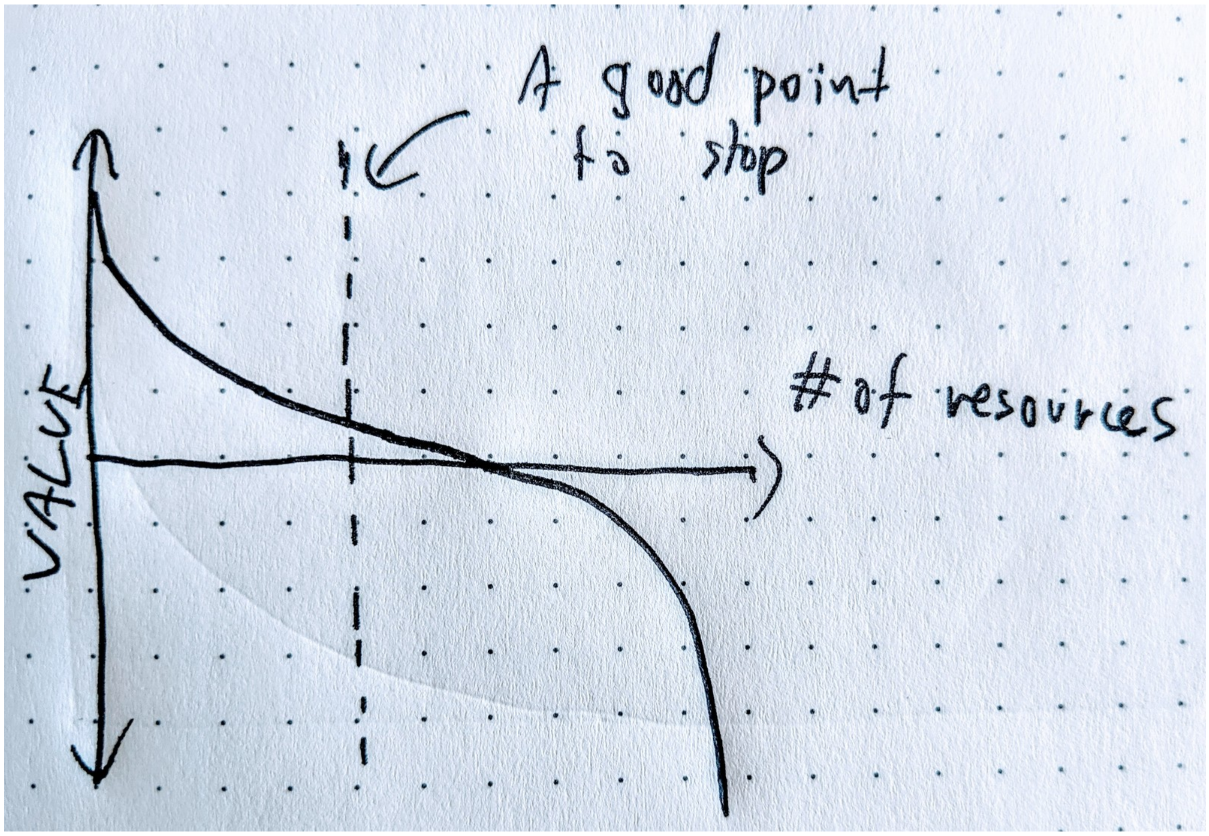
The more you learn, the faster you can learn. Keep this in the back of your mind as you read the rest 👇

In design, there's a well known concept of affordances (see Don Norman's "Design of Everyday Things"). When **someone** interacts with **something**, the **user** draws on all their past experiences to quickly learn how to use the **thing**. Well designed things optimize for affordances that would be evident to a user. The canonical example of this from Don Norman's book is the door - a vertical bar indicates pull, and a horizontal bar (or no bar) is push. You can learn to use a new door quickly.

Let's break down this system. A well designed object **teaches** a user how to interact with it by drawing on their past experience. This means simply using something well designed actually teaches you how to learn other things. Conversely, a poorly design object doesn't make proper use of the user's past experience, and may actually hinder your rate of learning by teaching you the wrong thing (see The Backwards Brain Bicycle). Replace "object" above with "learning resource" (ex: blogs, videos, screencasts, podcasts, docs, etc. - any learning resource) and you'll see where I'm going with this.

If you pick the best resource to learn something, you'll not only learn fast, you'll be able to cut through the crap on other resources and be able to learn even faster.

The Process



1. Find the **best** resource on something, usually you won't be looking for it when you find it.
2. Extract everything you can from it. This usually means you should try to implement its lessons immediately because you need to know what the complementary skills you're missing are.
3. Because you now know enough "lingo" about the skill, you can easily find the **best** resource for each complimentary skill. Here's where you unlock real power, because you've prematurely graduated past the "beginner" guides - and that's a great thing.
4. Stop. This is counter-intuitive, but unless you want to become a master at **something** (besides learning), you really just want to know **everything** at an intermediate level.

The steps above will get you to a good level of proficiency in one **domain**. Usually when you start learning about a new domain, it's because you know nearly nothing about it and your view is limited to what "the public" knows (which leads you to believe there isn't much to this domain and it could be "easy"). The aim is always to learn just enough that you can have an interesting discussion with a domain expert and gain their respect while realizing you know nearly nothing compared to them (see the original paper on the Dunning-Kruger effect but know that what I described has been liberally extrapolated from this paper).

I discovered a hidden benefit here. If you know a lot about a domain, you'll be able to relate to that knowledge when diving into another domain. I'm not talking about a vague subconscious effect, but something you should consciously try to do - come up with analogies to the domains you already know about when learning something new. You can then explain this new domain to people proficient in other domains you've learnt about (see The Feynman Technique).

Schools often teach you a lot of things that aren't obviously useful, especially in domains you couldn't care less for - but this is the benefit in knowing everything.

This is also why learning a bit about a lot of things helps you learn a new domain better faster than just focussing on mastering a single domain.

I left an important part out of the process above - how do you actually find the **best** resource. I don't know.

That said, I've gotten "lucky" enough times that I think I know how to optimize for finding it.

Finding the best resource

1. Figure out what resource **type** works best for you - Video? Screencast? Books? Docs?. For me, videos get me interested in a topic, but jumping into a hard project in a new domain and clawing my way through documentation is the fastest way to get good. I can't read long-form text, but audio books work.
2. Keep exposing yourself to that type of content and ignore all else, unless there's a widely considered "great" resource for your domain in a form you dislike. An example for me was reading Learn You a Haskell.
3. As you process the content, see if you can come up with analogies that help you understand it - if you can, and if what you come up with blows your mind, that's a good sign.
4. Follow the previous process, figure out what the complementary skills are, and repeat step 1.

The **best** resource is a bit of a lie - it's really just the one that's *best for you*, given your prior experiences and the medium you enjoy. You'll usually only know you found it in hindsight. For example

1. This course by Jeffrey Way alone had the biggest impact on my career - it was my gateway into JavaScript (I'm one of those people who learnt jQuery first), then
2. This course by Dan Abramov OR this video by mpj (I can't decide) was my gateway into functional programming, and eventually
3. Hacking on eslint-plugin-relay as a side project while at Facebook was my gateway into compilers and language theory

Together, this chain of events got me to giving this talk less than a year later.

More thoughts

There are a lot of things I haven't learnt yet. I keep trying while following a breadth-first-search approach to domains and resources as described above. Consequently, I've failed at learning things. Sometimes I can't find the best resource, but more often, I haven't given a good resource an adequate amount of time.

Writing is a great example of something I've constantly failed at (multiple attempts at the SAT and ACT proved this to me, and this piece might be evident of that fact). I may have found a good resource to help me with that, so I dove right in.

1,349

Kudos

1,349

Kudos