Welcome to the rest of your life

Objectives

- ✊🏽 **Build serious respect** with your manager + team
- ⌛ Quickly become **productive in a new codebase**
- 👨🏽‍💻 **Learn from coworkers** (even if they’re not helpful)
- ❌ Understand **failure modes** (+ prevention)

I’m Rahul

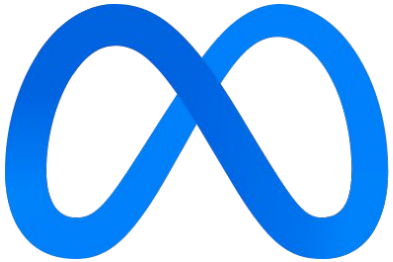
- **My goal**: ensure your success as the new engineer
- Onboarded 7 times in my career
- EM + TL at Meta and Pinterest
- Directly coached 100s of engineers through onboarding

My background

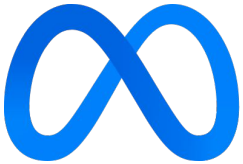
4 internships

3 people

400 people



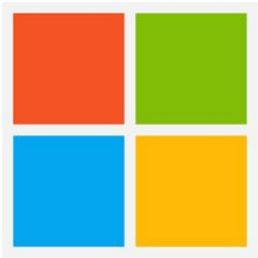
25,000 people



Exceeds Expectation (EE)

4/5 Perf Rating

Outstanding Impact (O)



Panicked Noob

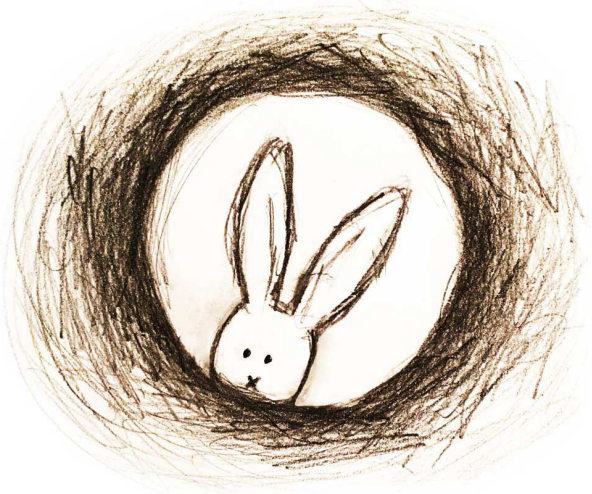
- Putting in tons of hours
- Trying to read all the code/docs/wikis
- Extremely focused on making a good impression
- Lacks understanding of where they should focus
- Working in isolation
- High stress

Complacent Noob

- No urgency to onboard
- Doing the work assigned but nothing more
- Not curious about past projects or current priorities
- Lacks understanding of where they should focus
- Working in isolation
- *Not enough* stress

Don’t fall into traps

- Find the right balance and tactics as you onboard
- Tons of company cultures and tech stacks – there is **no single right answer**
  - Determine what to focus on
  - But there are wrong answers (*“keep your head down”*)



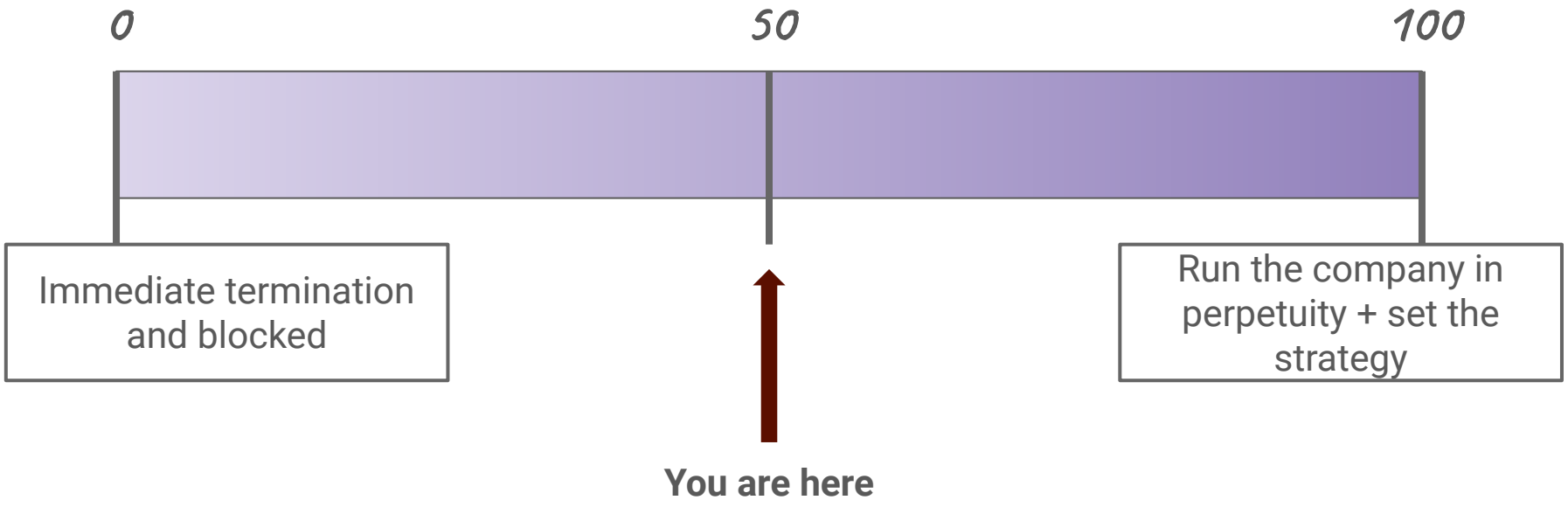
How this course works

- 🧠 The onboarding mentality
- 🙋 Asking for help
- 💻 Learning the codebase
- 🔑 Building relationships
- 🤷 Contextual onboarding tips

Why Onboarding Matters

FIRST
IMPRESSION

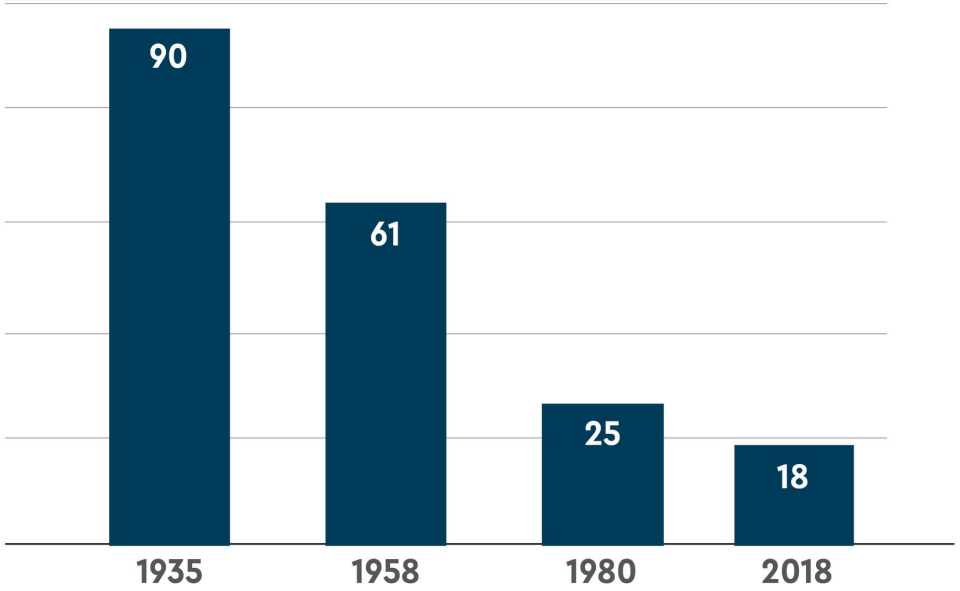
The **Trust Meter**



The tech industry changes rapidly

- Average stint at a tech company is 1.5 - 2 years.
- Even within the same company, you will likely switch teams multiple times. Re-orgs are common!
- Assuming you'll be able to stay put is a recipe for a stagnant (and stressful) career.

AVERAGE LIFESPAN OF S&P500 LISTED COMPANIES IN YEARS



INNOVEST™

Source: US Bureau of Labor | Statistics

Employees who leave within 90 days

30%

HIRE SLOW, FIRE FAST

Do it in a hurry

- You need to become productive quickly
- Build strong and robust relationships
- Act with urgency

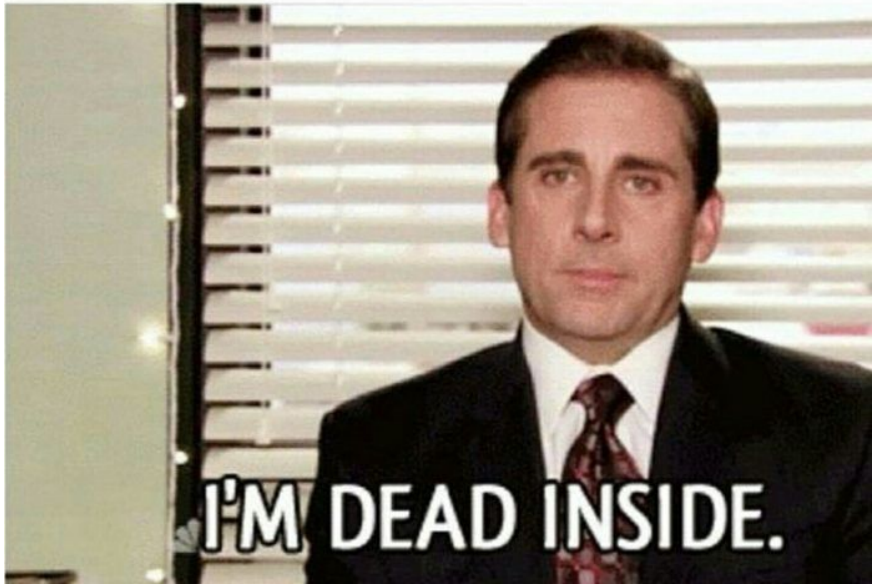
Traits of high performers

- Problem-solving: Ability to define a problem and then determine and implement a solution
- Communication: Effectively conveys ideas to others
- Self-direction: Self-starter, someone who initiates new assignments and challenges
- Drive: Desire to succeed, goes above and beyond what is asked of them
- Adaptability/flexibility: The ability to be flexible in a changing environment, take on new tasks

Attack your onboarding with a plan

Companies Suck At Onboarding

When people ask how I handle this job...



Don’t expect an amazing onboarding

- “Being new” is not a problem that existing employees have
- Many companies have a non-existent onboarding process

Speed of onboarding

“Here’s your laptop, good luck”

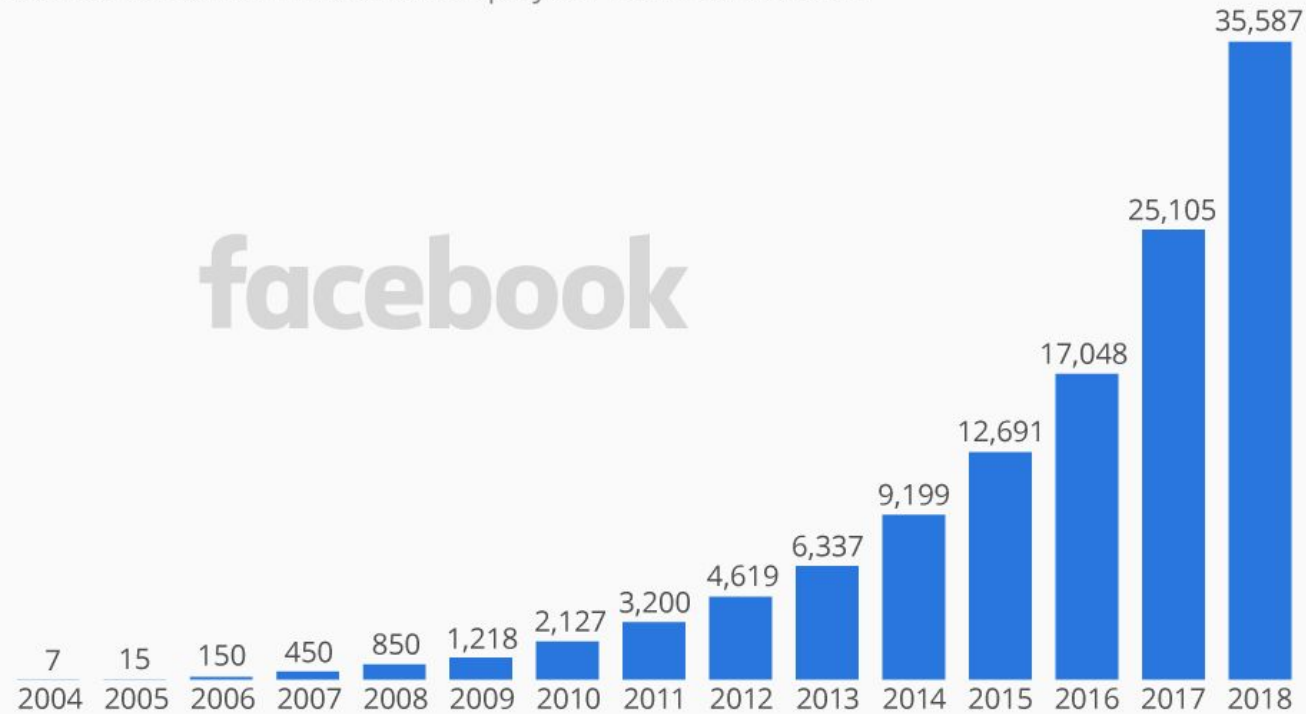
“You have to complete 3 months of training before coding”

Technology changes quickly

- Software as an industry is pretty new → rapid changes
- Companies that are hiring are also changing rapidly
- Up-to-date onboarding is challenging

Facebook: from Harvard dorm to 35,000 employees

Number of full-time Facebook employees from 2004 to 2018



 Source: Facebook

statista

Companies do not incentivize slowing down

- Short-term thinking focused on immediate output
- Onboarding:
  - Need to take 1 step back to go 2 steps forward
  - Important but not urgent
  - Not “assigned” to anyone, so it never gets done

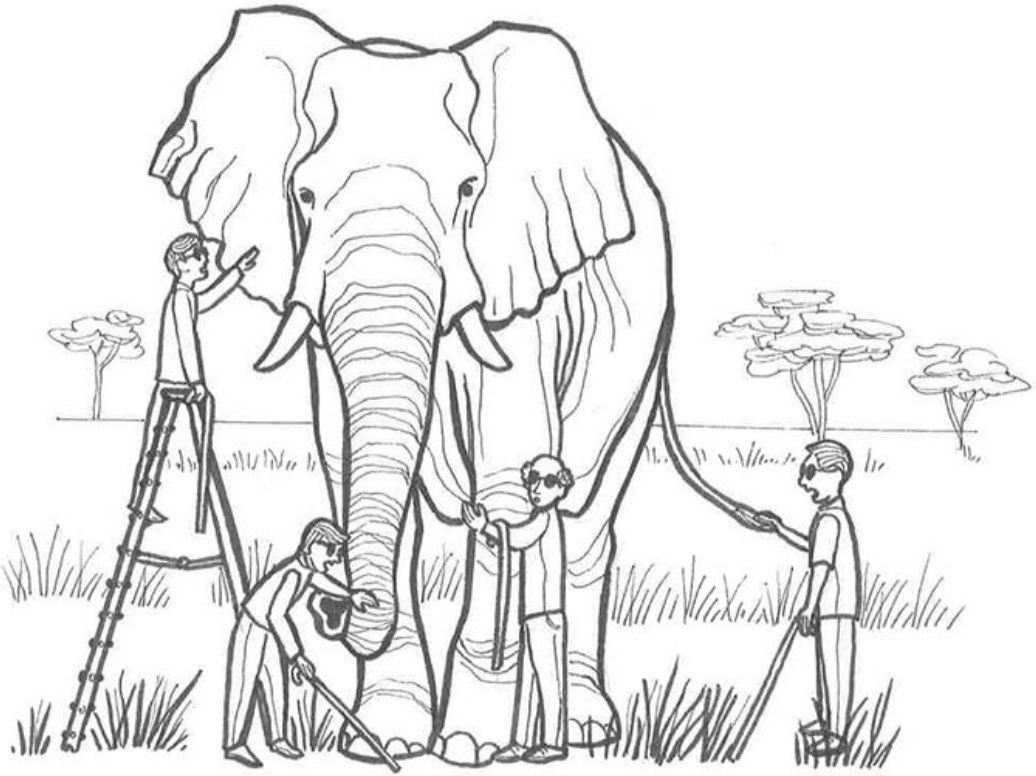
Poor onboarding is an opportunity

Being New Is A Superpower

Take advantage of your newness

- Nobody wants to be mean to the newbie
  - → Request resources aggressively
- You’re not expected to have impact immediately

Blind men and the elephant



Putting it into practice

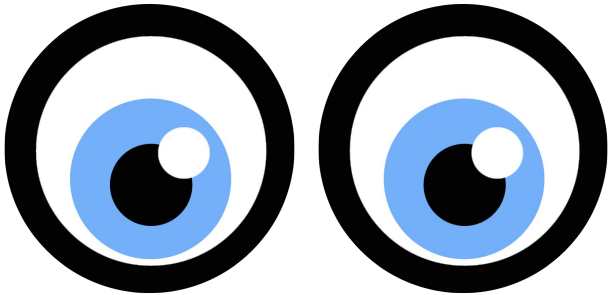
- You can connect the dots between different tools/teams
- Revamp a resource (wiki, code, etc) to be easy to understand for yourself
  - → make it easy for EVERYONE to understand

What that means for you

- Work with urgency → your ‘newness’ superpower will degrade
- **“You will never have fresh eyes again”**

The Onboarding Mentality

Talk & Observe



Talk

- ❏ Weekly/quarterly priorities
- ❏ Major pain points
- ❏ Time spent breakdown
- ❏ Who else should you talk to?

fb.com/boz
twitter/@boztank
instagram/@boztank
mailto:boz

A Career Cold Start Algorithm

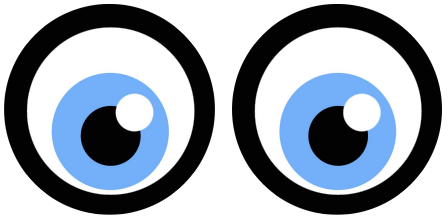
Several times in my career, I've joined a team whose work was already well under way, where I had a massive knowledge deficit, and didn't have pre-existing relationships. None of those excuses relieved me from the pressure I felt to establish myself and contribute. Over time, I realized that the natural instinct to push for early impact leads many incoming leaders into challenging relationships as they expose their knowledge deficit and waste time. So, I developed an algorithm that has helped me ramp up quickly — and in several cases — have an impact in a relatively short period of time, while minimizing collateral damage.

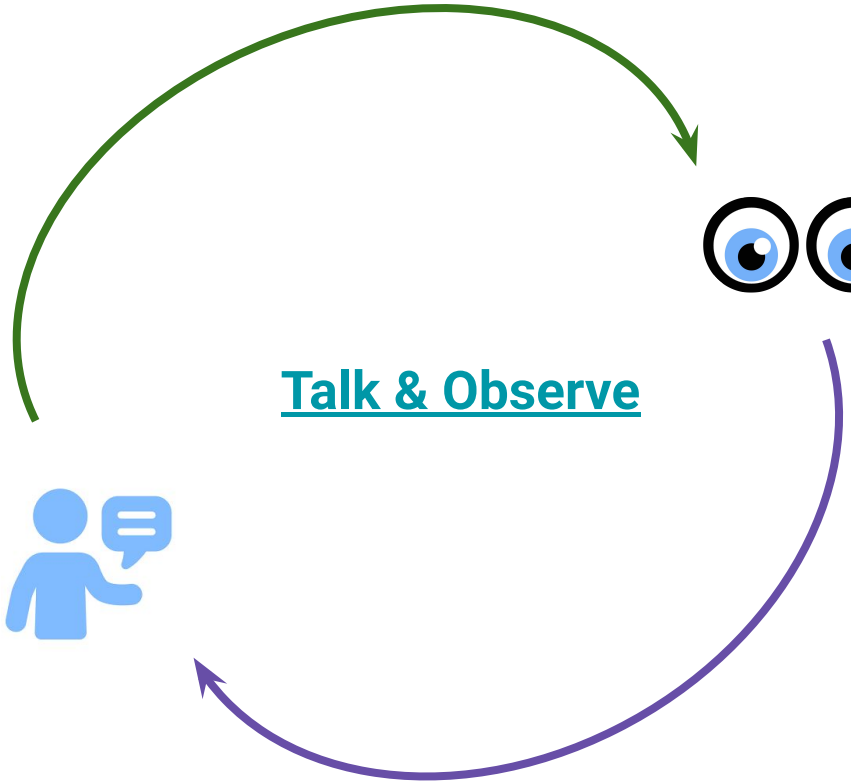
The first step is to find someone on the team and ask for 30 minutes with them. In that meeting you have a simple agenda:

- For the first 25 minutes: ask them to tell you everything they think you should know. Take copious notes. Only stop them to ask about things you don't understand. Always stop them to ask about things you don't understand.
- For the next 3 minutes: ask about the biggest challenges the team has right now.

Observe

- ❏ Code contributions
  - ❏ Authored and commented
- ❏ Design docs
- ❏ Active experiments
- ❏ Calendar blocks





Observe

code, designs, experiments

Talk

priorities, problems, time spent

Create An Onboarding Doc

13

What should be part of an Onboarding doc?

Principal Software Engineer [L7] at Target
a year ago

Onboarding | Principal Engineer | Target | Working With Your Manager

I just joined Target at a Principal level. My manager is spread very thin and wants me to take initiatives and has told me to start networking heavily in the first few weeks. My plan is to create an Onboarding doc and share it with my manager. I'm going to use this doc to manage expectations and use it to review together 3 months down the line. How can I structure this doc? What pieces of information are more relevant/vital for such a doc. Any pointers?

1K Views 4 Comments

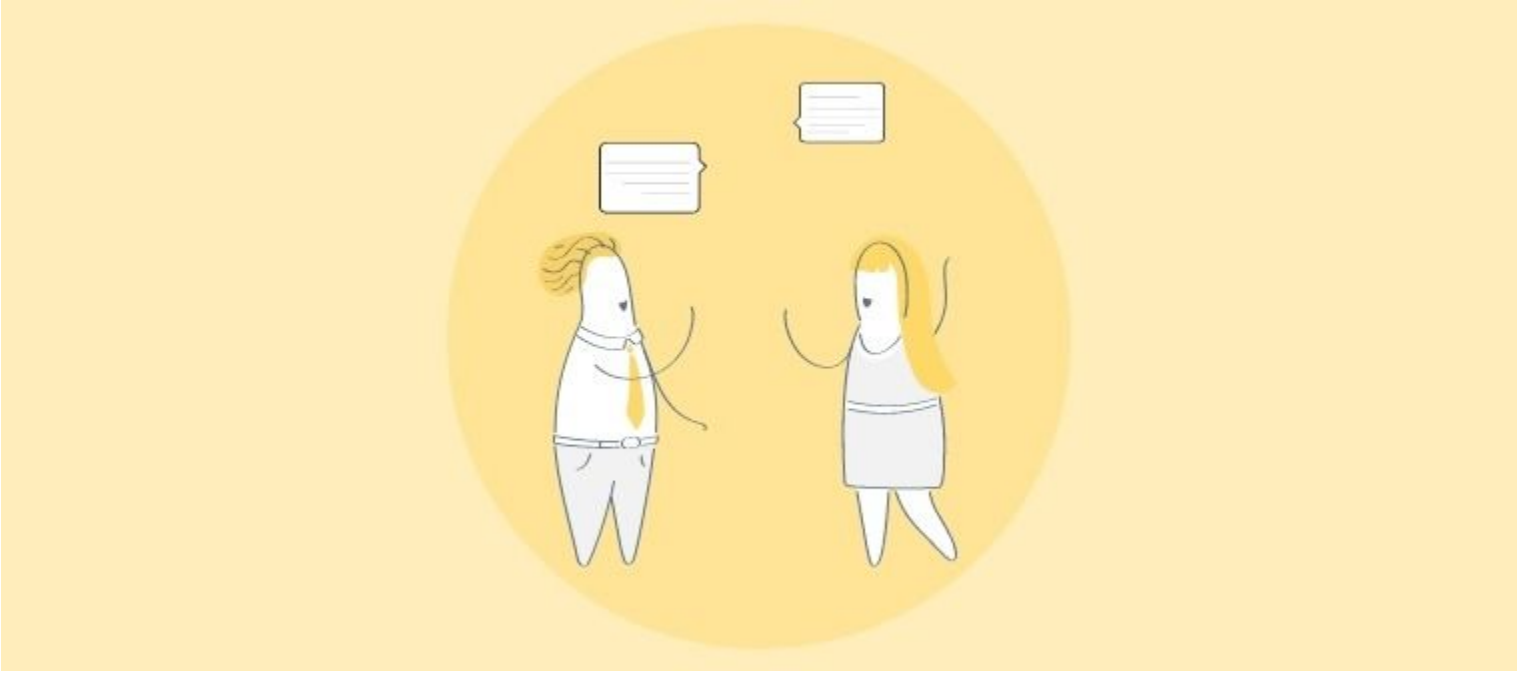
Onboarding doc

- Team charter
- Partner teams + people
- Expectations + milestones
- Project ideas (with t-shirt sizing)
- Engineering rubric / career matrix

You can (should) create this

- Use this as a way to collaborate with your manager
  - Ask questions
  - Ask for input
- Make sure it’s written down
  - Remove ambiguity

Have The “Meta Conversation”



Conversation about conversation

- *How* should you communicate?
    - How often do you want to meet?
    - How often do you want feedback?
    - What is your preference for written vs verbal updates, and the format?

“Can we talk about how you prefer us to work together over the next few weeks?

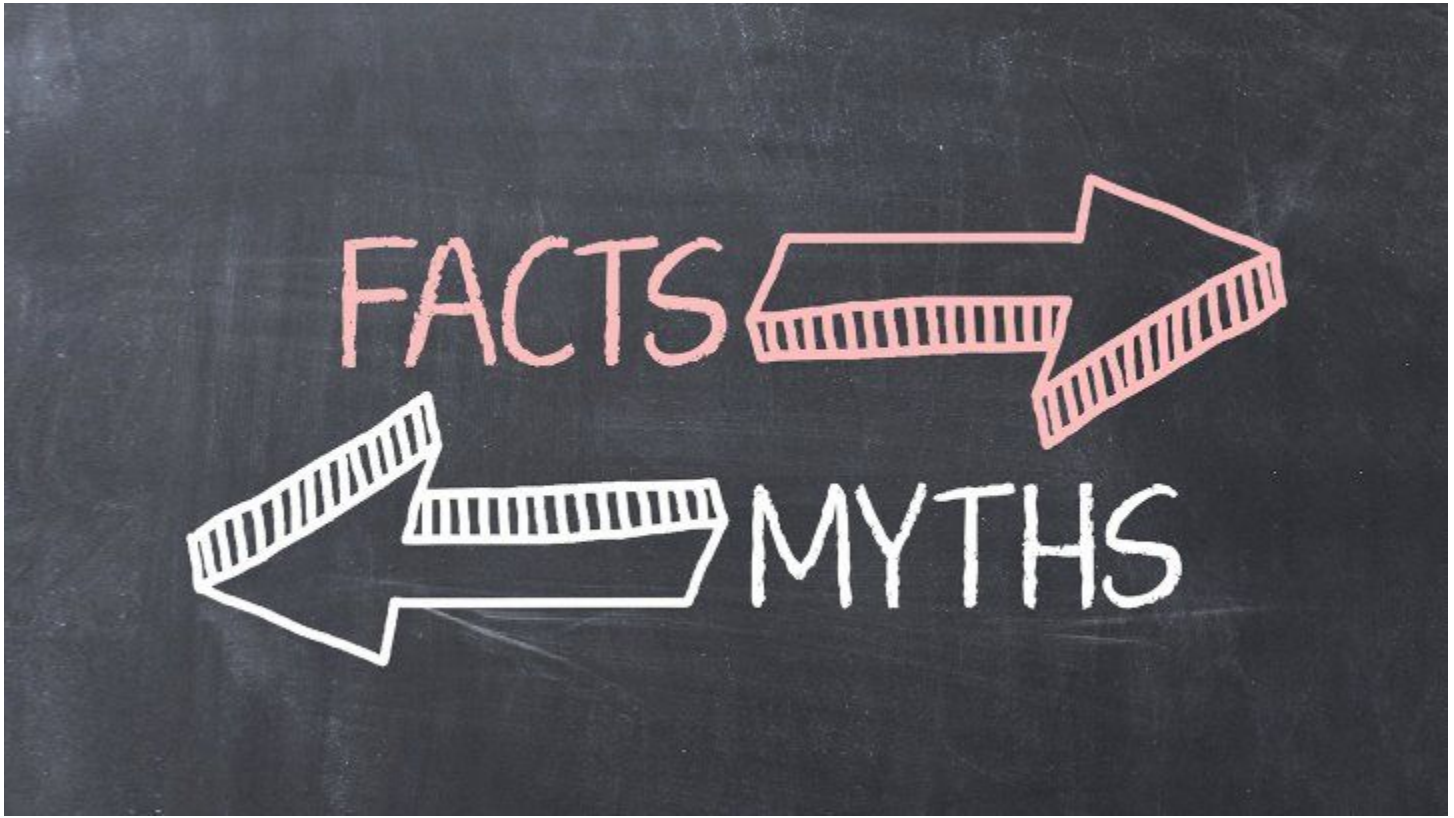
I want to ensure I’m communicating timely updates and help requests without being disruptive.”

Prep For Onboarding Success

- ❏ Leverage the **Talk & Observe** framework 🗣️
- ❏ Sharpen your **onboarding doc** 📄
- ❏ Have the **meta-conversation** 💭

Asking for help

Misconceptions With Help



Myth: Asking for help is a sign of weakness

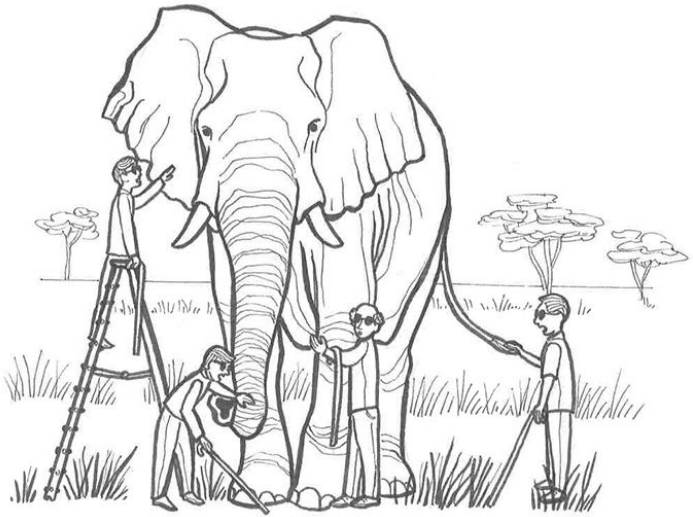
Reality: You’re expected to ask questions to gain knowledge and foster interaction

Myth: Everyone knows so much more than me, I'll waste their time

Reality: Each person's knowledge is incomplete

Taro Expertise Theory: You can become more knowledgeable than *anyone*

- Within a group of **people P** (e.g. *the team of 10*)
- Within a certain **domain D** (e.g. *the build system*)
- Within a certain **time T** (e.g. *a week*)



Get the support you need

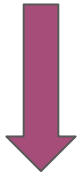
⏰ Treat your setup as urgent → ping people

- 😪 Don’t take this to an extreme and be lazy
- 🪞 Reflect if you could have answered the question on your own

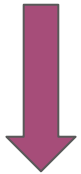
“The man who asks a question is a fool for a minute, the man who does not ask is a fool for life.”

— Confucius

Good questions

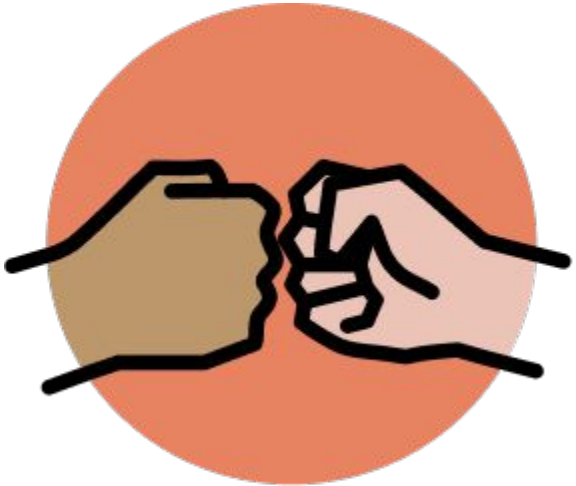


Deeper learning



Job success

Your Onboarding Buddy



What is an onboarding buddy?

Someone who can help you adjust to cultural norms and direct your questions as a “catch-all”

Why an onboarding buddy is essential

- Your manager is usually not equipped to help you with in-the-weeds details for programming or tooling
- If an onboarding buddy is not assigned to you, ask for one

Figure out your support channels

Reactive

- Answer tech questions as they come up
- Offer initial code reviews
- Navigate team/org dynamics

Proactive

- Regular check-ins about your progress
- Team integration (lunches, meetings)

Sample 1:1 agenda

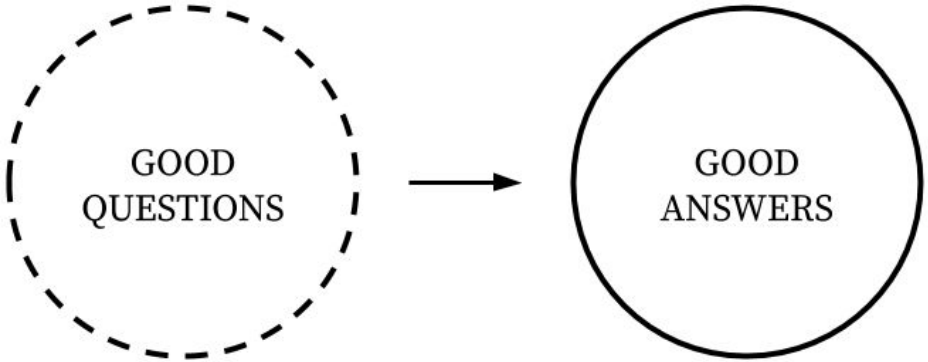
- What did you work on? **(fact)**
- What do you plan to work on? **(fact)**
- What's top of mind for you? **(emotion)**
- What has been surprising? Any feedback? **(emotion)**
- Action items

Ask Questions Well

What makes a high-quality question?

Get the help you need

Build trust



Adding context

- Add context in a succinct way
- Requires some judgment to identify what is relevant or not
- Make it easy for people to help you

“Hello, are you there?”

Quality: 10/100

My app is crashing when I delete the contents of the base edit text.

Quality: 40/100

My app is crashing when I delete the contents of the base edit text **with a NumberFormatException**.

Quality: 60/100

My app is crashing when I delete the contents of the base edit text with a NumberFormatException. **Here's the stack trace and the relevant code from MainActivity.kt.**

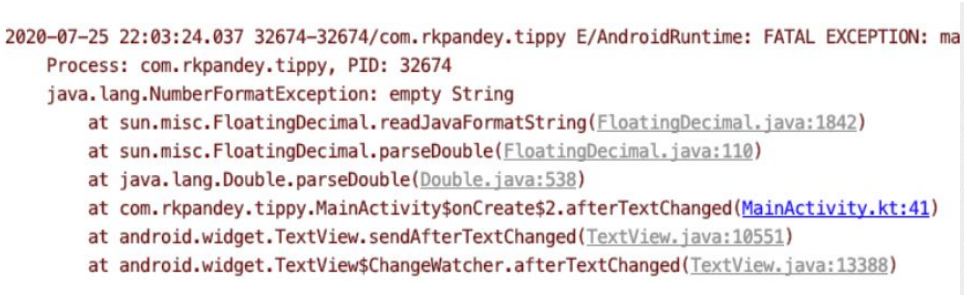
Quality: 80/100

```
2020-07-25 22:03:24.037 32674-32674/com.rkpandey.tippy E/AndroidRuntime: FATAL EXCEPTION: ma
    Process: com.rkpandey.tippy, PID: 32674
    java.lang.NumberFormatException: empty String
        at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1842)
        at sun.misc.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        at java.lang.Double.parseDouble(Double.java:538)
        at com.rkpandey.tippy.MainActivity$onCreate$2.afterTextChanged(MainActivity.kt:41)
        at android.widget.TextView.sendAfterTextChanged(TextView.java:10551)
        at android.widget.TextView$ChangeWatcher.afterTextChanged(TextView.java:13388)
```

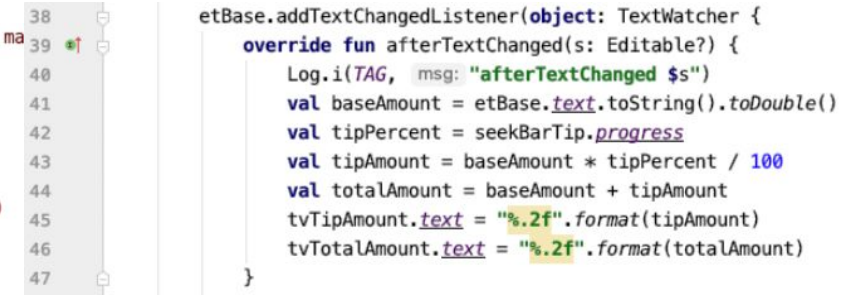
```
38
39  etBase.addTextChangedListener(object: TextWatcher {
40      override fun afterTextChanged(s: Editable?) {
41          Log.i(TAG, msg: "afterTextChanged $s")
42          val baseAmount = etBase.text.toString().toDouble()
43          val tipPercent = seekBarTip.progress
44          val tipAmount = baseAmount * tipPercent / 100
45          val totalAmount = baseAmount + tipAmount
46          tvTipAmount.text = "%.2f".format(tipAmount)
47          tvTotalAmount.text = "%.2f".format(totalAmount)
    }
```

My app is crashing when I delete the contents of the base edit text with a NumberFormatException. Here's the stack trace and the relevant code from MainActivity.kt. **I made sure this is not a null pointer exception, since adding null checks around the edit text doesn't fix the issue. So I suspect the issue is either (1) with the string to number conversion or (2) the EditText widget has a bug in Android.**

Quality: 90/100



```
2020-07-25 22:03:24.037 32674-32674/com.rkpandey.tippy E/AndroidRuntime: FATAL EXCEPTION: ma
    Process: com.rkpandey.tippy, PID: 32674
    java.lang.NumberFormatException: empty String
        at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1842)
        at sun.misc.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        at java.lang.Double.parseDouble(Double.java:538)
        at com.rkpandey.tippy.MainActivity$onCreate$2.afterTextChanged(MainActivity.kt:41)
        at android.widget.TextView.sendAfterTextChanged(TextView.java:10551)
        at android.widget.TextView$ChangeWatcher.afterTextChanged(TextView.java:13388)
```



```
38
39
40
41
42
43
44
45
46
47
etBase.addTextChangedListener(object: TextWatcher {
    override fun afterTextChanged(s: Editable?) {
        Log.i(TAG, msg: "afterTextChanged $s")
        val baseAmount = etBase.text.toString().toDouble()
        val tipPercent = seekBarTip.progress
        val tipAmount = baseAmount * tipPercent / 100
        val totalAmount = baseAmount + tipAmount
        tvTipAmount.text = "%.2f".format(tipAmount)
        tvTotalAmount.text = "%.2f".format(totalAmount)
    }
```

Make it easy to receive help

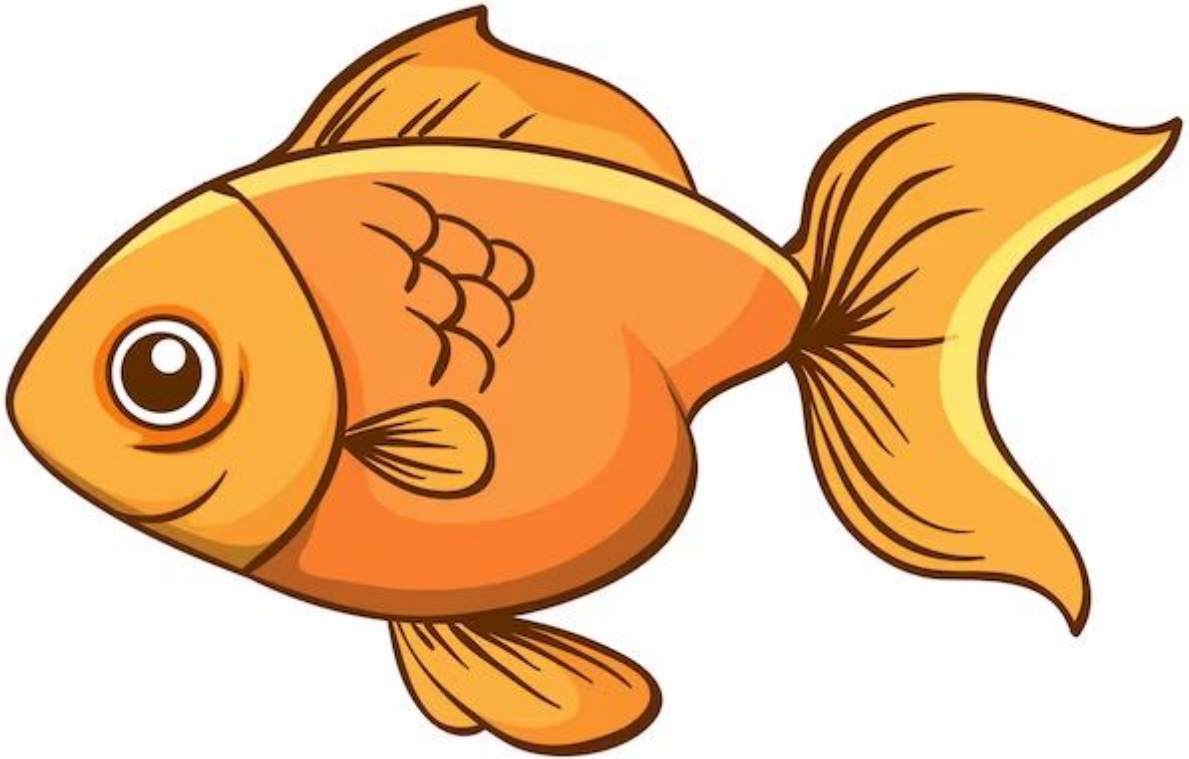
Ask in public forums instead of individual people

- Asking a single person is a SPOF
- Individual people will be slower than the “wisdom of the crowd”

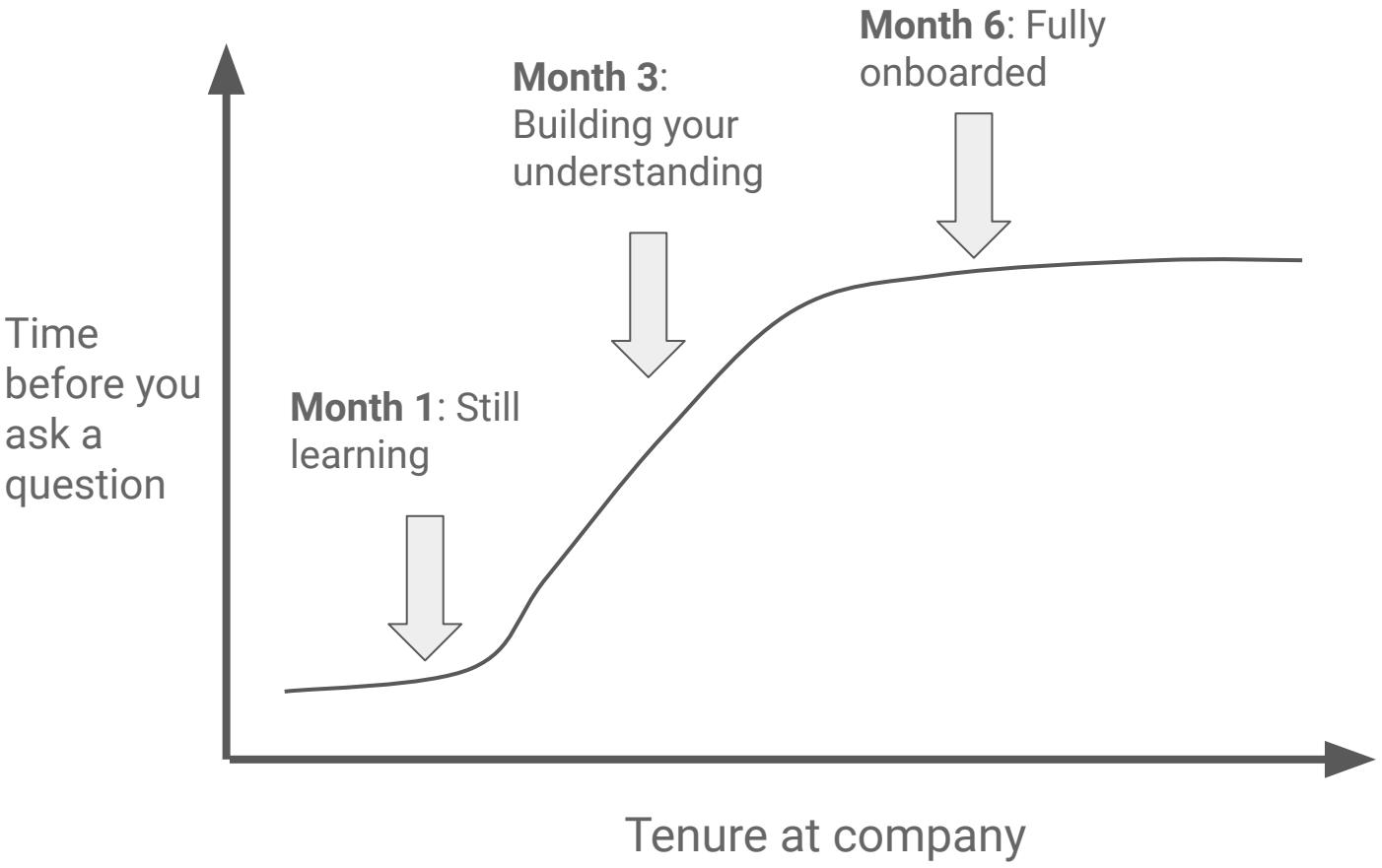
Help others while you help yourself!

- Other people may benefit from the question
- Your questions are a valuable signal to the team

Extract Maximum Value Per Answer



“Learn how to fish” instead of “receiving fish”

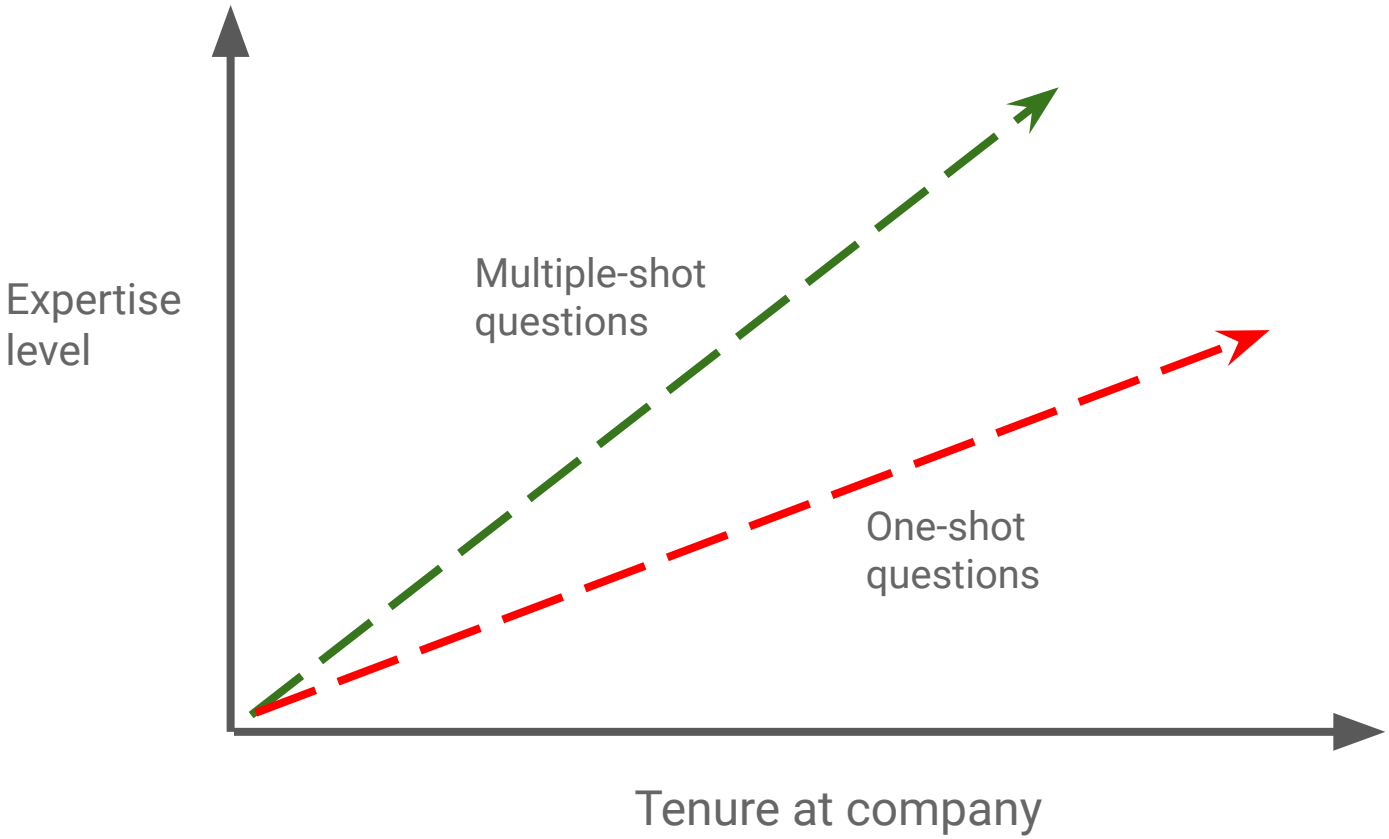


Keep the conversation going

- Ask follow-up questions
- Understand how they arrived at the solution

“This fixes your bug.”

“What process did you use to find this fix?”


Expertise level
Multiple-shot questions
One-shot questions
Tenure at company

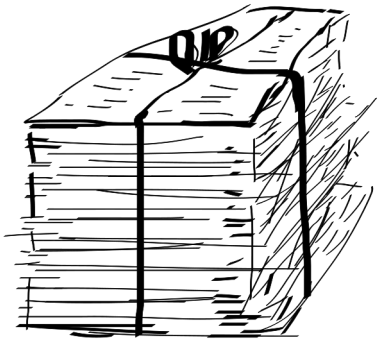
Case Study: Environment Setup

Define the objective

- Your goal is **not** to setup your environment
- The goal is to **do something** that environment setup enables
  - e.g. make a copy or color change
- Act with urgency and ask for help

Don’t try to understand everything

Abstraction is powerful and essential when you’re working in a large codebase.



Don’t get sidetracked

Forecast ahead

- Find another person onboarding, work with them
- Continue reading the setup guide so you’re not constantly blocked
  - e.g. if you’re waiting on database access, try the next step
  - Error messages can be valuable

Failure Mode: Excessive Handholding

63

What does it mean to not require handholding anymore?

Entry-Level Software Engineer [SDE 1] at Amazon
a year ago

Amazon | Asking Great Questions | Editor's Choice | Junior Engineer | Mid-level Engineer | Onboarding

I've heard Alex, Rahul, and other engineers within Taro talk about handholding when joining a new company for all engineers and for newer engineers in general.

What does it mean that an engineer doesn't require handholding anymore? Does this mean the frequency of the questions gets diminished or is it more about needing as much initial help to start tasks or something else entirely?

3.2K Views 2 Comments

Question A

Question A’

Question A*

Question A

Question B

Question C

Questions need to evolve over time

Code reviews need to evolve over time

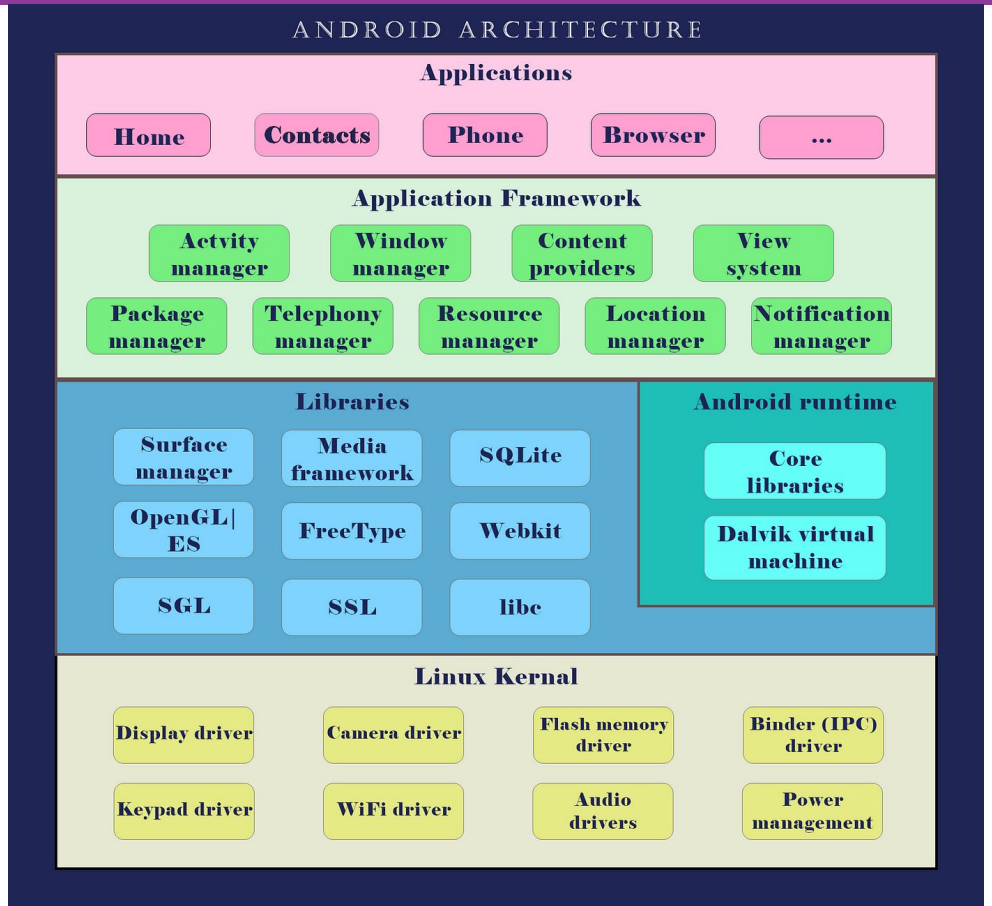
Reflect

Measure your progress

- Look at earlier code comments you received
- Write down questions you've had
- Don't make the same mistake twice
- Ask your onboarding buddy to look for patterns

Learning The Codebase

How To NOT Learn A Codebase



What NOT to do

- Excessively reading through the code and documentation is inefficient

Problem is exacerbated at fast-moving tech companies

- Docs can’t keep up

What NOT to do

Excessively reading through the code and documentation

What to do

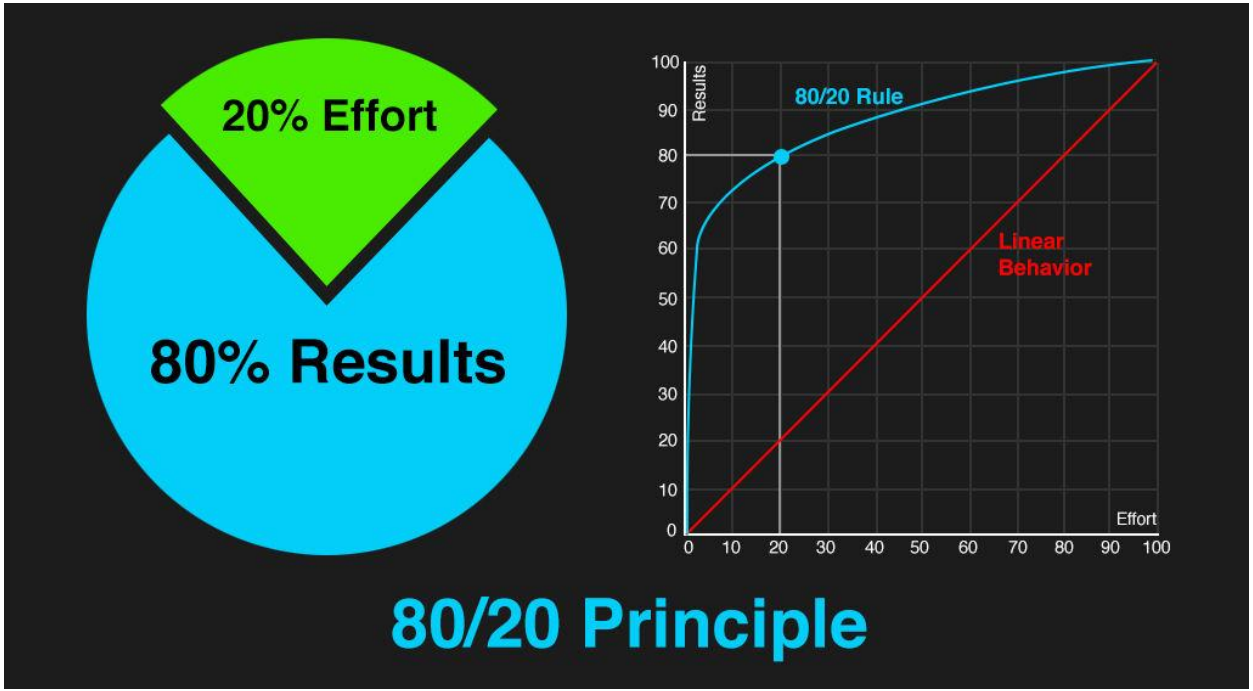
- Just dig in and start coding!
  - Break things
  - Add all the print statements ever
  - Get the debugger working
  - Run code and see what happens

Pair program **a lot**

- Learn the codebase
- Learn quirks of the tooling and workflow
- Build a relationship with the paired engineer

Get your hands on the keyboard

Figuring Out What Matters



Millions of lines of code…

- Code follows 80/20 rule: 20% of the components account for 80% of the impact
- How to figure out that 20%?
  - Talk to people
  - Use blames to find hotspots
  - Look for patterns in code reviews
  - Alerts

rpandey1234 / taro-web

Type / to search

Code | Issues | Pull requests 2 | Actions | Projects | Wiki | Security | Insights | Settings

Show the parent video from the video page #975

Merged cvuong merged 2 commits into main from show_parent_video on Nov 9, 2023

Conversation 1 | Commits 2 | Checks 1 | Files changed 1

cvuong commented on Nov 9, 2023

This PR will show the parent video thumbnail in the sidebar for a video that has a parentVideoId present.

Show the parent video from the video page ✓ 40576c7

vercel bot commented on Nov 9, 2023 • edited

The latest updates on your projects. Learn more about Vercel for Git ↗

Name	Status	Preview	Comments	Updated (UTC)
taro	✅ Ready (Inspect)	Visit Preview	💬 Add feedback	Nov 9, 2023 10:58pm

Changes ✓ 49e3a6c

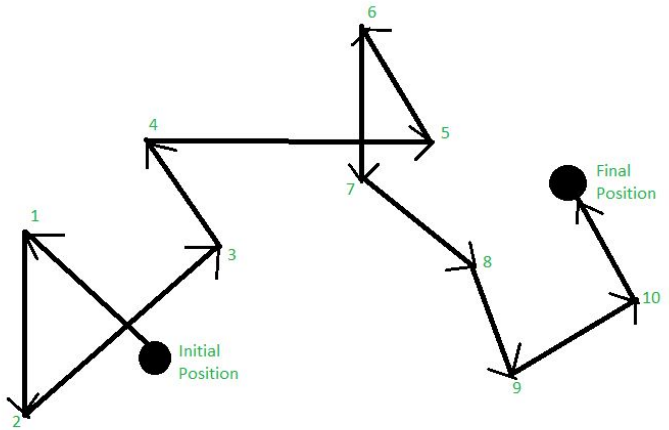
```
97        lessonDescription,
98        relatedVideos: relatedVideosSlice,
99  +     parentVideo,
100     },
101     revalidate: 1, // seconds
102   };

123   lessonTags: LessonTag[];
124   lessonDescription: string;
125   relatedVideos: Lesson[];
126 + parentVideo?: Lesson;
127 }
128
129 function VideoLessonComponent({
130   initialLesson,
131   lessonTags,
132   lessonDescription,
133   relatedVideos,
134 + parentVideo,
135 }: VideoLessonProps) {
136   const { isAuthLoading, signedInUser } = useContext(UserContext);
137   const [loggedHistory, setLoggedHistory] = useState<boolean>(false);

256           />
257         </div>
258         <div className="shrink-0 lg:ml-10 lg:block lg:basis-[320px]">
259 +         {parentVideo && (
260 +           <div className="mb-4 flex flex-col space-y-2">
261 +             <Text size="lg" bold>
262 +               Clipped From Main Video
263 +             </Text>
264 +             <RelatedVideo lesson={parentVideo} />
265 +           </div>
266 +         )}
267           {lesson?.firebaseId &&
268             playlist &&
269             (playlistLessons || []).length > 0 && (
```

Randomly investigating a class is unlikely to be fruitful

- Your team may not have context, so you'll get stuck asking for support and chasing people down.
- Focus more on throughput than complexity or buy-in when you're new.



Rack Up Quick (Small) Wins

Ideas for quick wins

- Share a learning
- Update docs and tell people about it (not to brag, but to invite feedback)
- Wikis often don't have screenshots.
  - Add a screenshot or video of the setup flow
- Fix a “papercut”, e.g. a misspelling in the code or minor bug

Decomposition FTW!

- Create a branch
- Write code
- Deploy and test locally
- Submit for review
- Land code
- Monitor in production

- Small, quick changes
- Quality of life improvements for your team
- Non-production updates
- Knowledge sharing

- Ambiguous changes
- Controversial changes
- Major refactor involving multiple teams

- Small, quick changes
- Quality of life improvements for your team
- Non-production updates
- Knowledge sharing

- Ambiguous changes
- Controversial changes
- Major refactor involving multiple teams

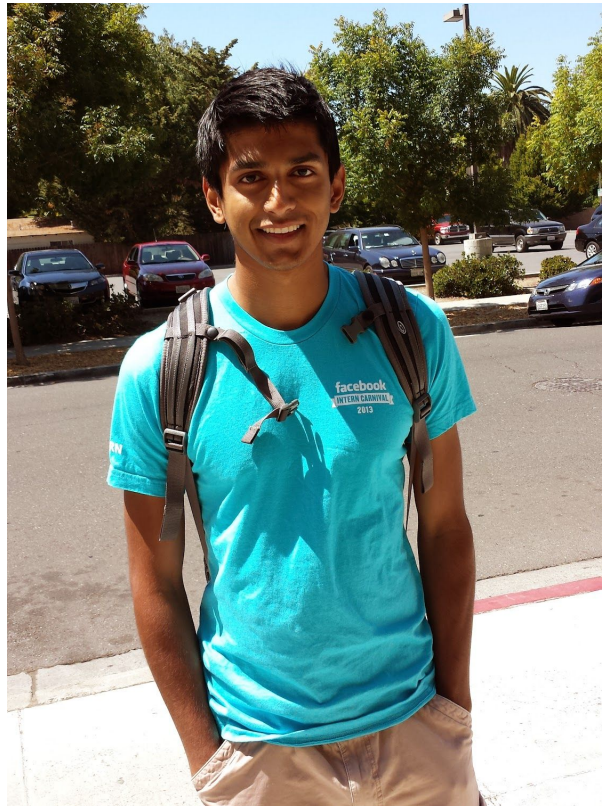
Failure Mode: Fake Learning

What is fake learning?

Feel good about consuming an endless barrage of:

- Tutorials
- Books
- Blogs

Past Rahul wasted weeks



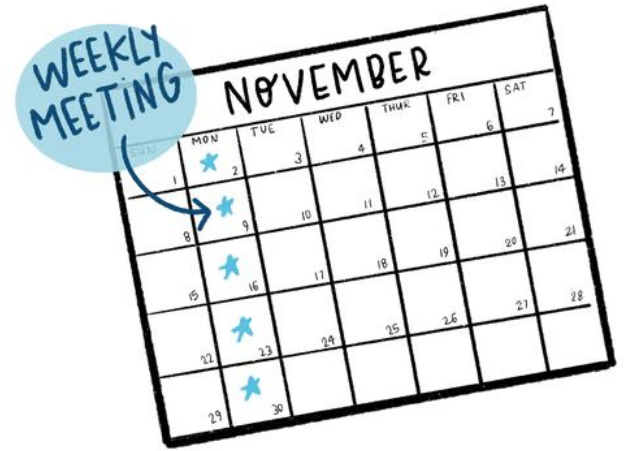
Focus on action, not theory

Building Relationships

Working With Your Manager

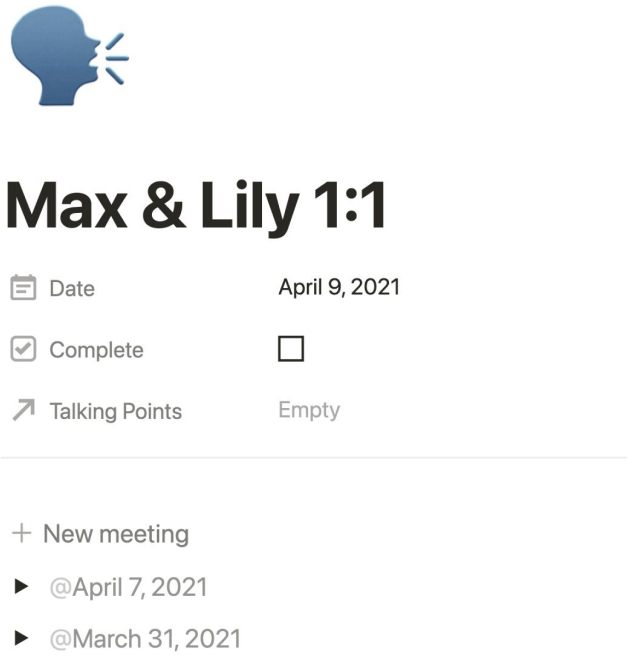
Your manager is critical

- This is the most important relationship you have at work
- Bare minimum:
  - Recurring 1:1 meeting
  - Weekly or biweekly
  - 30+ minutes
- Let them know you're eager for feedback



Keep a record

- Write down your thoughts, feedback, and action items
- Keep track of history, along with context
- The act of writing communicates that you care



Adding structure

June 10

- **Updates**: …
- **Reflections**: …
- **Action items**: …

June 3

Adding structure

	L5 Behavior	L6 Behavior	Commentary
Project Impact	…	…	
Direction	…	…	
Engineering Excellence	…	…	
People	…	…	

Meet The Team!

Software Engineering is a team sport

- Don’t rely too much on your manager
- Book 1 on 1s with teammates proactively
- For close teammates, look into recurring meetings

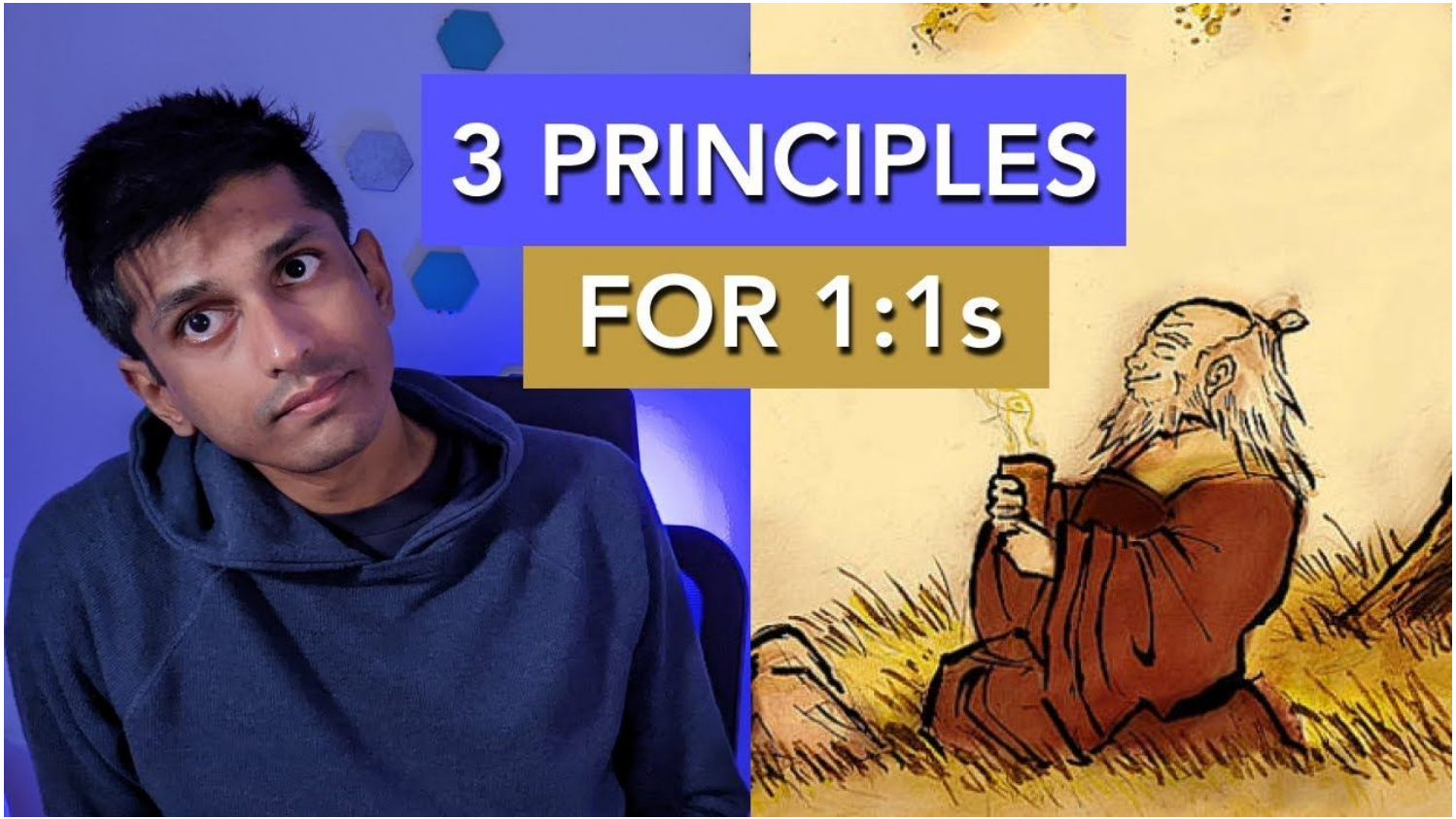
Understand the team

- Leverage team events
- Figure out who can throw you under the bus and make them like you (***pre-mortem***)

Prefer IRL

- If you’re hybrid, try to come in as much as you can
  - Align days with people in office
- If you’re remote, get to the office (company or you pays)
- Spend extra effort to engage
- Trust is 10x easier built in person compared to remote

Uplevel Your 1:1s



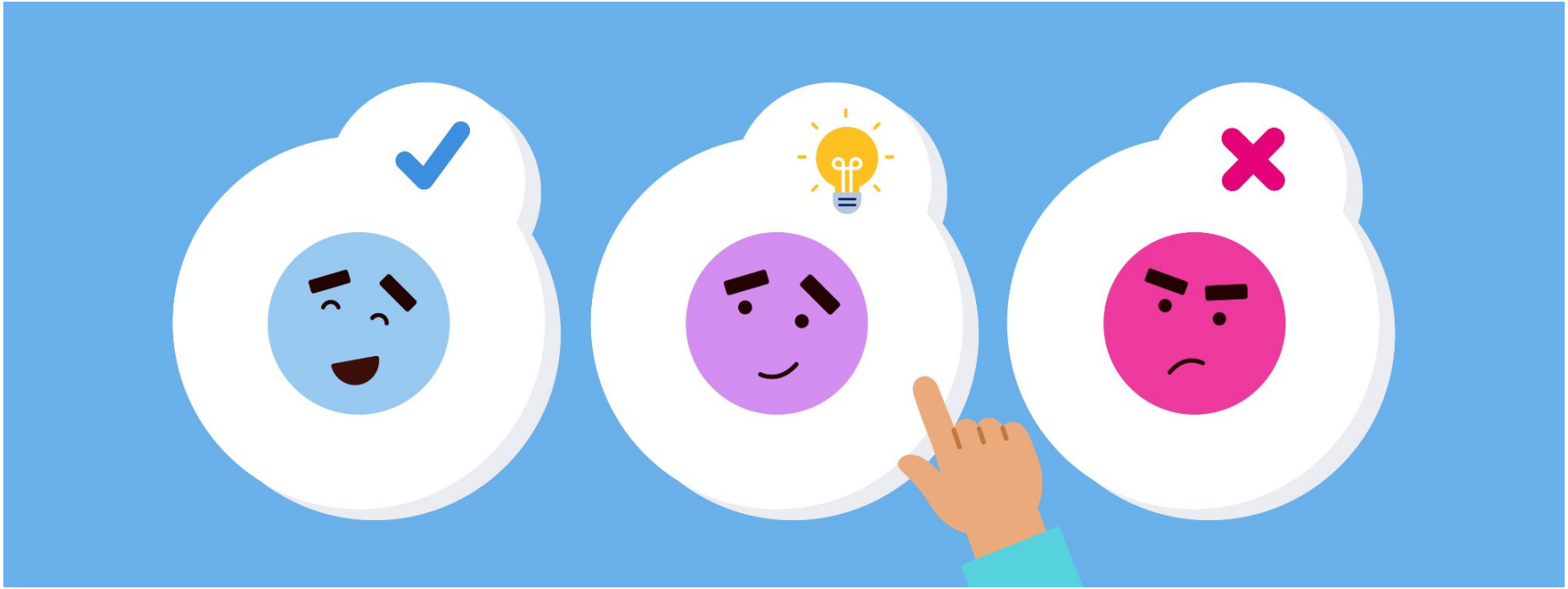
3 Principles For Productive 1:1s

- Pursue awkward 1:1s
- Go beyond status updates
- Write down takeaways

Things to talk about in your 1:1s

- ❏ Reflect on what's going well and not going well for you
- ❏ Reflect on what's going well and not going well for the team (and team priorities)
- ❏ Ask for opportunities or resources
- ❏ Give feedback or ask for feedback
- ❏ Brainstorm
- ❏ Career/performance review planning

Feedback



Seeking Out Feedback

- Course correct **rapidly** while onboarding
- Ask along various dimensions:
  - **Task-specific** - what's one thing that would have made this stronger?
  - **Behavioral** - what's something I do well that I should do more of?
  - **Meta-level** - what's the best way for us to share feedback with one another

Areas To Focus

- Start open-ended
- Areas to get feedback on:
  - Velocity
  - Code quality
  - Quality of questions

“Do you have any feedback for me?”

“What could I start or stop doing to more quickly ramp up on the team?”

How To Get More Feedback

- Schedule dedicated 1:1s for feedback
  - Make your intention known
- Share your own areas of growth and weakness
- Follow-up with feedback giver

Thank People (A Lot!)

Getting people to like you

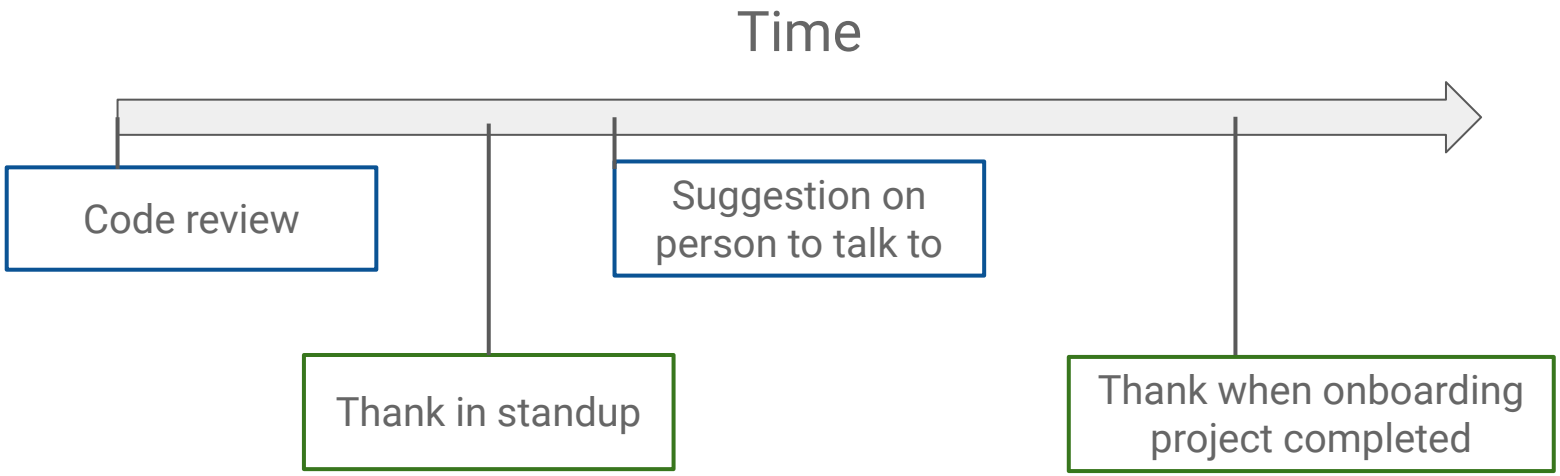
- Properly thanking people is how you build social capital
- Social capital is how you advance your career and get promoted

“Thanks for the pointer”

“That was so helpful, you unblocked a critical milestone of this project”

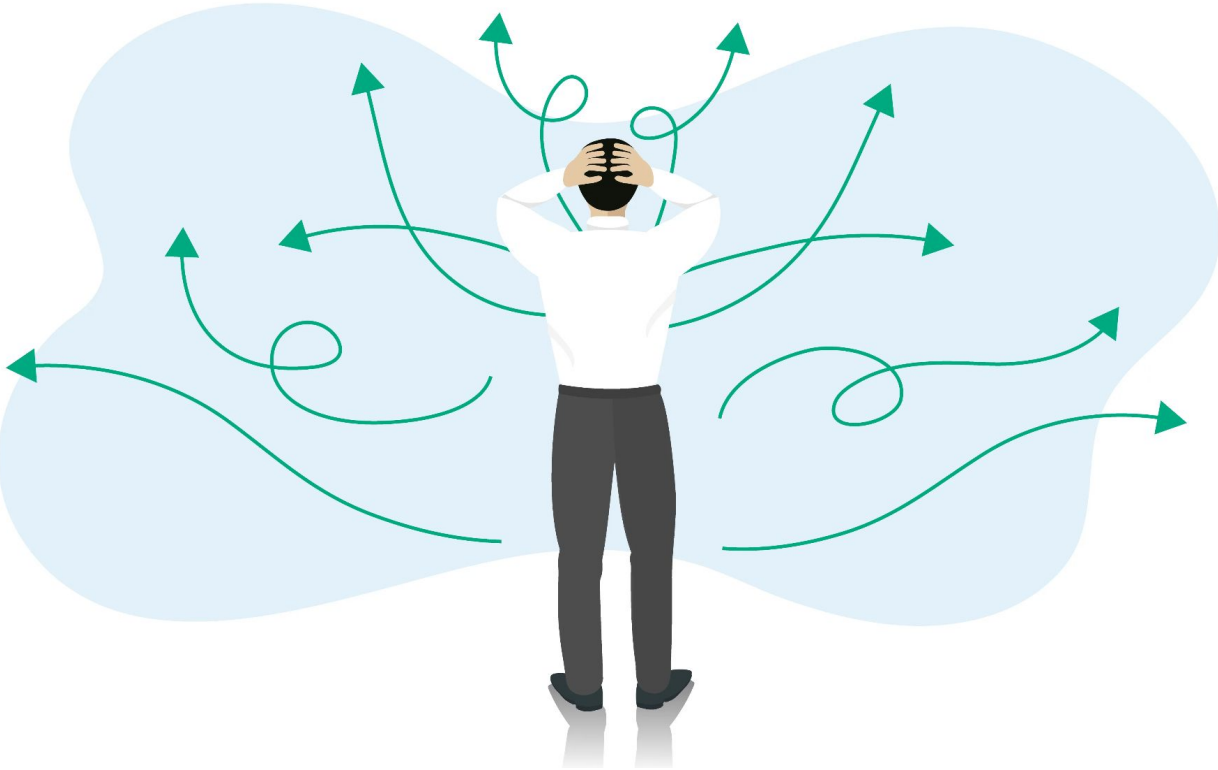
Thank People. A Lot!

- Basic: Cookie-cutter 1-sentence thanks over DM
- Better:
  - Thank them in-person or over a call
  - Write a heartfelt thank you note (1 paragraph)
  - Let their manager know
  - Thank in public forums
  - Recurring thanks


Time
Code review
Thank in standup
Suggestion on person to talk to
Thank when onboarding project completed

Contextual Onboarding Tips

How to know if you’re doing well



Panicked Noob

- Putting in tons of hours
- Trying to read all the code/docs/wikis
- Extremely focused on making a good impression
- Lacks understanding of where they should focus
- Working in isolation
- High stress

Complacent Noob

- No urgency to onboard
- Doing the work assigned but nothing more
- Not curious about past projects or current priorities
- Lacks understanding of where they should focus
- Working in isolation
- *Not enough* stress

Doing well

- Are you adding value? (help newer people)
- Improve the onboarding process
- Understand/contribute to code reviews

Help needed

- Not engaging with people on the team
- Not getting assigned work
- Need constant support

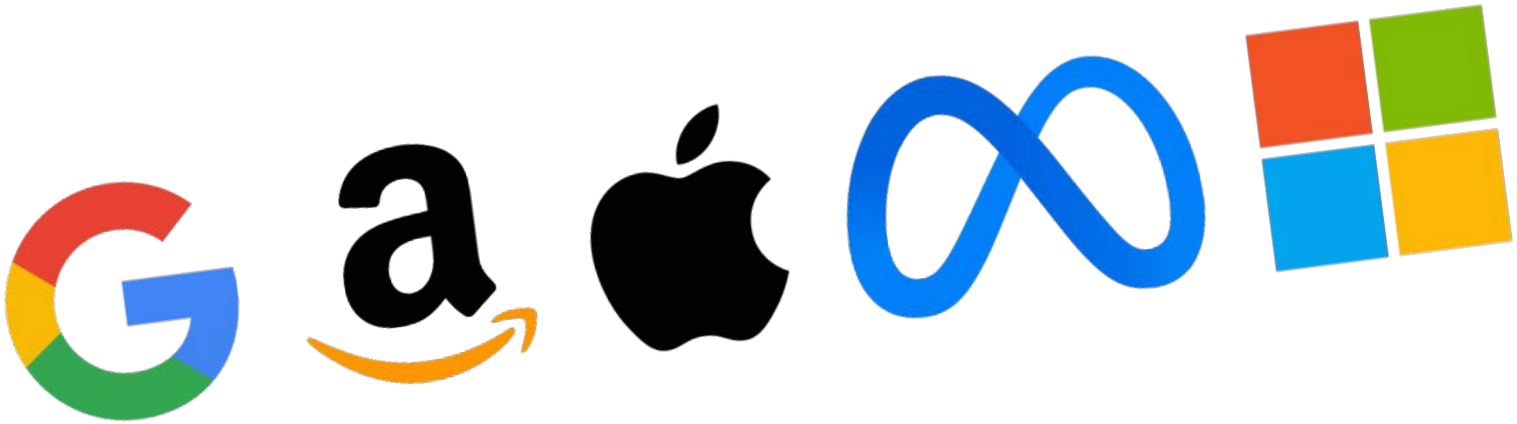
Connect your work to the bigger picture

- Start building a priority map
- High growth company → lots of areas are neglected
  - This leads to opportunity
- High quality of work leads to trust to tackle problems

Small Companies

- Less process → less formal onboarding
  - More important to get stuff done
  - Land code within a week
- Knowledge will be in the head of a few people
  - Find the tenured people
- Don’t assume processes or people exist

Large Companies



- Expect more process
  - Don’t let this be an excuse to move slowly
- Invest in relationships
- Understand how priorities are translated by team/org
- Disperse and collect knowledge from around the company

MK

PSC

Little r me

LPs

Connect

“Mark”

Workplace

GRAD

Junior Engineers

Seek feedback

- And act on it!
- Don’t “solo carry” a project
  - Create milestones
- Pair programming is especially valuable

Senior+ Engineers

Engineering Expectations by Level

Level	L#	Scope
Entry-Level	L3	Task
Mid-Level	L4	Feature
Senior	L5	Project
Staff	L6	Product / System
Senior Staff	L7	Product Group / Org
Principal	L8	Company
Distinguished	L9	Industry

Share your opinions

- Expectation of having domain knowledge
  - Contrast how current company compares with past companies
- “Respect what came before”
- Win with quality!

Cross-team impact

- You need to lay the foundation for team and org-level impact
  - Build relationships
- Meet other senior/staff/principal engineers

Probation Periods

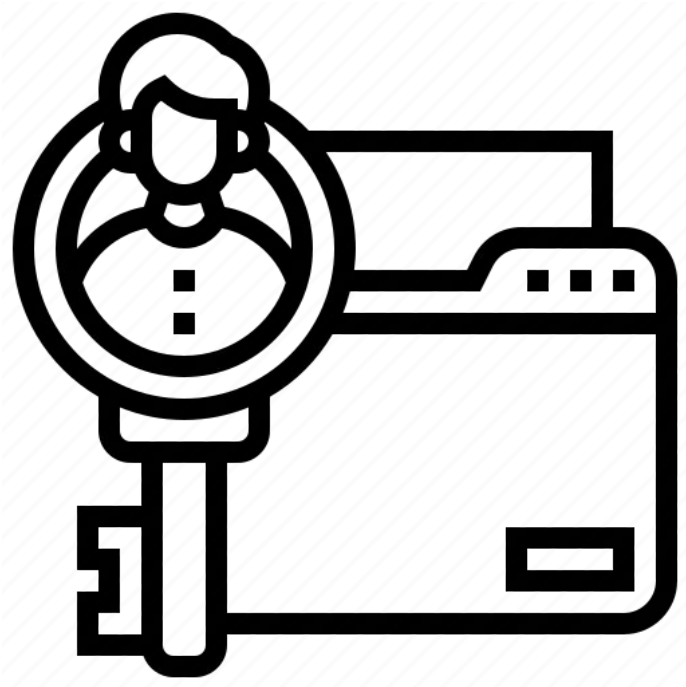
Probation or notice periods

Labor laws outside the US can make it difficult to fire someone

“Contract-to-hire” is also common

Best practices

- Chat with your manager early (even before you join)
- Information access may be limited, so call out when this happens
- Understand the criteria you’re being judged on, e.g. code, design docs



Conclusion

Pass It On

“From one engineer to another”

- Make life easier for the next person
  - Improve the onboarding process
  - Support more junior engineers
- Build more relationships
- Have more impact

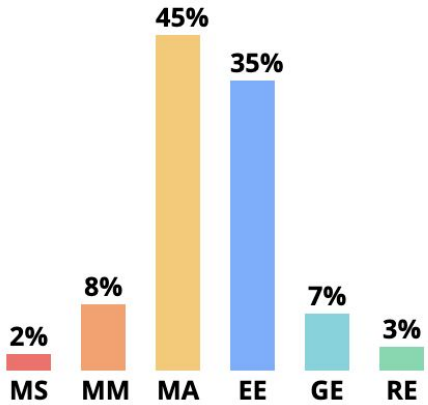
Onboarding is about trust

- Trust is the currency through which you grow your career
- You’re always gaining or losing trust
- Influence is essential for further career growth

What Happens After Onboarding Success?

- Revisit any parts of this course, no need to memorize anything
- Talk to your teammates!
- Apply the knowledge in the **next week**

Understand performance reviews



RE	Redefines Expectations
GE	Greatly Exceeds Expectations
EE	Exceeds Expectations
MA	Meets All Expectations
MM	Meets Most Expectations
MS	Meets Some Expectations

Meta

Attack your onboarding with a plan

Figure out your support channels

Make it easy to receive help

Get your hands on the keyboard

What next?

- Share + get support: joinTaro.com/questions/create/
  - 🧠 Get 10x better advice than anything on Blind or Reddit
- Find community: joinTaro.com/events/
  - 👩‍🏫 Join me for Group Office Hours!

Welcome to the rest of your life

team+onboarding@jointaro.com