Stanford eCorner

Deep Inside Facebook [Entire Talk]

Jocelyn Goldfein, *Facebook*

May 22, 2013

Video URL: http://ecorner.stanford.edu/videos/3160/Deep-Inside-Facebook-Entire-Talk

Director of Engineering Jocelyn Goldfein takes us on a trip inside the innovative culture of Facebook. In this illuminating conversation with STVP Executive Director Tina Seelig, Goldfein explains why code wins arguments, employees must have the right to take risks, and how Facebook strives to remain a hungry, yet humble, company.

Transcript

So, Jocelyn, welcome. Thank you very much. Thank you. Why don't you tell us - start out by telling us a little bit about your career path from a Stanford student to a Director of Engineering at Facebook. Well, I could probably spend the entire lecture just answering this question. When I was a student, I really had absolutely no idea what I wanted to do when I grew up. And I had majored in Computer Science because I loved programming; I love the logic and the sort of analytical nature of it. But I didn't really have like a super clear idea of what being a software engineer was like, even after multiple summer internships. And so I joined - straight out of college, I joined a company called Trilogy that was based in Austin, Texas. Mainly because Trilogy have a very different recruiting pips than a lot of the other big software companies, most of the software companies were saying things like well, come work for us and just make software.

And Trilogy was like well, come up first and do a little of this and little of that and we will expose you to a lot of things. And so I went and it was kind of true. It is a very chaotic company, but I made great friends and just was sort of put in situations where it was a very entrepreneurial environment actually because you just kind of have to figure out what to do and deliver and sort of not know any boundaries. There was not a lot of structure or processes to follow. You just have to figure out what to do to succeed. And I made great friends there, including three with whom I went on to a co-found a startup, which was again - I wrote code, I tested the code, I wrote documentation, I supported the customers, I deployed the software, I did everything. I hired people, I hired the front office manager, I stocked the fridge, so it was really terrific. And somewhere in there I also started managing people and I found that when I was managing engineers that sort of finally plugged in for me, that this was actually my right career path because I've been always fascinated by people as well as by technology, I was actually a minor in Symbolic Systems, Terry Winograd was my advisor, concentrated in HCI. So I was taking tons of psychology and linguistics classes at Stanford. You want to blow your mind, take a linguistics class the same class you take - the same quarter you take compilers like - and you'll find you're studying the same thing but from different sides.

And so being a manager I found was a lot about social engineering and it was a lot about understanding what made people tick and what made teams tick and they were just really interesting difficult, structural and systems problems to solve of the human kind as well as of the technology kind. So it was like, yes, at last, like every neuron in my brain is firing and I'm also just motivated to bring my A game by other people, by my teammates, by my colleagues. The reason founding a start-up was amazing was because I was working 120-hour a week, so I was working crazy time. But I was thrilled every minute because I was coming through for my team and they were coming through for me. The start-up though did reach sort of a point where we were doing a major strategic pivot and I was ready to move on, and I was ready to come back to Silicon Valley. My - I also married a Stanford grad, we were an - another Computer Science major and he and I had kind of made this pact that we would be in Austin for no more than two to four years, that two Stanford CS graduates were obviously destined to be based in Silicon Valley. We have been in Texas for over five years at this point what with the start-up and so we decided to come back to

California. And in California, I ran into a really good friend of mine from school, Jeremy Sugerman, who had joined a start-up founded by one of our professors, by my OS professor actually, Mendel, and it was called VMware and Jeremy had - I tried to hire Jeremy for my company, he wouldn't come, he went to Mendel's company instead and so this was a couple of years later, VMware was now 350 people, still pretty small. I'd only pulled an A- in the OS class, I had been doing nothing related to operating systems since, but I said what the heck. I went in to meet VMware, the recruiter told me, boy, if I had found your resume on my own, like I would have thrown it in the trash can basically because you've no operating systems experience, and that's what I've been told to look for, but on Jeremy's recommendation they talked to me and we just clicked.

So joined VMware as a manager, I dusted off my operating systems textbook and remembered what Scatter-Gather was all about and by joining VMware I kind of get on a rocket. VMware does not have maybe the consumer brand, sex appeal of a Facebook or Google, but they absolutely were doubling revenue and headcount every year and they proceeded to do that for four, five years in a row. So I joined in 2003 when it was 350 people and when I left in 2010, it was over 10,000. And I really grew up with VMware and I still have that same quality of wanting to come through for my team and of taking no prisoners of like really just wanting to have the most impact I could have. And something that was really exciting about being at VMware was our customers were all geeks and they loved us. They love VMware. We were transformative for what they were trying to do. It was like science fiction because they could solve problems they couldn't solve without us. And so that was just an amazing feeling that we were building something that was changing how people work and making their lives better. But as VMware grew to over 10,000 people I guess my career was - I should talk about the fact that when I joined VMware as a manager and I left as a Vice President, but I never really thought - there was never a moment where I walked into my bosses' office and said where is my next promotion coming from? I was always just thinking about how do I have the most impact I can have on the company, on my team, on our users.

And I think that by just trying to have impact for my career, kind of take care of me. And so it was, I never really thought of myself as consciously managing my career. But I left VMware finally when it was over 10,000 people and I decided I was ready for another really big shift and I wanted to make the jump from enterprise to consumer and where better to discover consumer than Facebook. And I actually originally planned to go to a very small company and I was not looking to join a company, even then Facebook was almost 2,000 people. I have never voluntarily joined a company that big. I have joined companies that were a few hundred people or started my own and so really kind of just talk to Facebook as a really kind of a just to get aware, to learn more about the space. And Facebook's Head of Engineering, Mike Schroepfer, was a classmate of mine from Stanford too. He was a fellow 1997 grad and a fellow section lead too. And so we had lunch and I just kept talking to start-ups around the Valley and I kept talking to Facebook and I go sleep at night and talking there with my husband then Facebook had more of the qualities of a start-up that I was looking for than the small start-ups I was talking to. And there was a no question after I met him that Mark Zuckerberg was by far the most impressive founder of all the founders I met, which is sort of a trite thing to say in hindsight like that's sort duh.

But Mark was amazing and Facebook had the culture of a start-up and so I gritted my teeth and said, well, this doesn't match any of the - it doesn't check any of the boxes. I said I had other than consumer, but what the heck, this is clearly what I want to do in my gut, so I went to Facebook. Have been there three years, have had an amazing round, I've gotten to work on really just great products that I use every day, photos, newsfeed, now on mobile platform, so I couldn't be happier. So let's drill down, you said the culture of Facebook was amazing and you walked in and you felt so comfortable. Can you paint a picture of the culture of Facebook? I mean, imagine, we are all visiting, what would we experience? Yes. You would experience unfinished ceilings, concrete floors, desks everywhere, no offices, not even Mark has an office and writing all over the walls and posters. And you'd see company slogans like "this journey is 1% finished" or "move fast and break things" or "fail harder". And Facebook was born out of disruption and was born out of trying things and seeing what happened and it not working out, and trying again and trying harder and never being daunted by failure. Doesn't mean we set out to failure, it just means like we're not afraid of it and we are willing to keep taking risks. And the entire environment is meant to keep you from feeling complacent or comfortable or like we have won, we never want to feel like we've won, we always want to feel pretty hungry and like somebody could come eat our lunch tomorrow because somebody could.

And we never want to take it for granted. We want - and so for - it's the most humble, successful company I have ever known and it may sound strange to say that Facebook is a humble company, but it really is. We really don't take success for granted and we think that our users are choosing to be there and they could just easily choose not to be there if we don't deliver a great service. I love the fact that first thing you said when I ask you to describe the culture was describe the space. Yes. Now, it's something we spend a lot of time thinking about in the classes I teach on creativity and certainly we're in this really remarkable space here at Stanford. How important is the space to the culture there and I know I've spent time at Facebook and it's really interesting especially with the new buildings. What sort of things did people think about in creating this space? Oh, it's very deliberate. It's absolutely deliberate. Culture, you cannot just think of the culture you want and then create it.

You have - culture arises from so many small things. Someone actually, a Stanford professor, gosh, I wish could remember his name, but from the business school one said something that really stuck with me which is culture is the behaviors that you reward and punish. At the end of the day, people look around and they mimic the behaviors that they think will be successful and they try to avoid the behaviors they think will be unsuccessful. I think that's really true actually and deeply true. But you've also got to try to show people what behaviors you want and what behaviors you don't want. And the space is one of those things that just sort of sneaks right past all your sort of human defenses and cynicism and processing and it just goes straight to the hindbrain and tells you and you understand it on a gut level, like when you walk on those concrete floors, you know, oh, we're not finished, we're not luxurious, we're not taking it for granted, like we're scrappy. And the open space is another huge one, you guys are all, I think many of you are probably Computer Science majors; hopefully, you've all taken 106, if not take 106 before you leave Stanford. This is the best opportunity of a lifetime. And you know if you've programmed that like you need focused attention, right? You need flow time, you need - and you know that even small interruptions like it takes you a long time to get back into the stream of things. And so the idea of having programmers sit out in open space, at open desks, with desks all around them and talking in conversation in foot traffic, that's controversial.

For many years it was the gold standard in Silicon Valley was to have offices. Engineers were housed in offices, preferably a single office, sometimes two people per office, it's what VMware did. And when we took over the campus from Sony, it was all offices we told the builders to actually start by knocking down every wall that wasn't structural and make as big and empty space in the building as we could and then we sort of fill it from there. And why - why was it so important to us that we would actually sacrifice engineer productivity, which is what we're doing to have everybody out in the open space like that. It's not density, it is not to save money. Well, it's because one of the key values of Facebook is be open. It's what the product is for, the product is for sharing a communication and that's also really fundamental to our DNA as a culture too. It's actually that we expect every individual to be informed and in the know about what's going on and that empowers you to make good decisions and so we sort of set the expectation that everything is out in the open, everybody is plugged in, everybody is aware of what's going on. And then we put headphones in the vending machines and try to create lots of private spaces where people can hide and give everybody a laptop and have a pretty loose work from home policy. So we do everything we can to mitigate the productivity impact of that openness, but ultimately every time we have to choose, we choose openness.

So what is the on-boarding process? I mean somebody gets hired... Yes. ...and you can't just plug anyone right into this type of culture, what happens to take a normal person off the street, plug them in and they became a Facebook employee? We call it bootcamp, literally. It is a six-week on-boarding program and in bootcamp and everybody goes into it. I was VP of Engineering of VMware, I had not written code in seven years. When I got to Facebook, they said here is your desk, here is your laptop, here is your UNIX account and here is five bugs that are assigned you to fix. And you spend your first six weeks at Facebook, fixing bugs and implementing small features all over the site and attending lectures, you have a bootcamp mentor who is a full-time software engineer, whose job is to help you. They help you get code reviews, they help you figure out what tasks to work on and at the end of the six weeks, they will help you find a team. And I think that bootcamp is so - it's such a good program for so many reasons. I actually gave one of the on-boarding - I actually give Facebook's introduction to culture for bootcampers.

And one of the many great things bootcamp does is it builds empathy, before you identify with a particular team, you identify with Facebook and you fix code and even if you walk in determined to do back-end services, we'll make you write some web code, we'll make you fix an issue on mobile. No matter what you're going to do, we're going to give you at least the tools to inspect, to investigate, to know about what's going on in other parts of the world. And it really sends the message we want to send, which is everything connects, all engineers are connected to one another. The Facebook employees is just another subset of the social graph. And so in bootcamp, you hook up your social graph to your first set of co-workers who then spread out to the four corners of engineering and give you connections in every part of the company. But you know a little bit of something about everything after bootcamp and that's an incredible asset for a new hire. So for six weeks, you are just doing bootcamp? That's right. So what are you looking for when people are coming to interview? I know that the standards are incredibly high and it's one of the most attractive places to work in the world. How important is technical skills as well as other things like creativity, and being able to work on teams? What are you looking for? The number one thing we're looking for in hiring a software engineer is the ability to write code and is the ability to reason and to be analytical and to find mistakes and fix them quickly and to analyze the running time so that when you are later on building systems, you demonstrate that you have the potential to think about hard systems issues. If you are a new grad, frankly that's the primary thing, is that ability to write code.

We also want to make sure that you're not a jerk. We definitely like want people at Facebook who are nice and we will reject people who have great coding skills just because they seem like they would be appalling to work with. But we want people who have enthusiasm, who have a fire in their belly, who are not going to take no for an answer, we want people who are not going to be afraid, we want people who are going to attack, making software with gusto. And we are looking for creativity, so we've actually - we've meddled with the interview process a lot. I have actually personally hacked on our interview process a lot. Another reason you find so much graffiti on the walls at Facebook. Actually, we invite our employees to write on the walls. It's because we want people to feel like if there is something at Facebook that you don't like or that you think should be different like, pop open the hood, like the cement is never dry, you can mess with it, you can change it. And I've messed with our interview process and one of the things that I did is I noticed one thing that made engineers really successful at Facebook was when they had just really good intuition for what would be a good feature or a good product. We obviously have designers and product managers that we work with, but the way we work is so iterative like we try things, we test them, we try something else we test them and so the difference between a good intuition and just trying things and testing them you might try three things or try 10 things, right, and save months, if you just have great intuition.

And so we started interviewing, consciously interviewing for that and we won't hire you on intuition alone, if you're not also a great coder. Well, we might in product management, but we wouldn't in a software engineer, but it turns out to be something that distinguishes good from great software engineers. So we started interviewing for that. So obviously we all know about Sheryl Sandberg's book, Lean In... Yes. ... okay, and I want to ask you just between you and me, is Facebook culture designed to be supportive of women really? I don't think it's designed in that way, but I think it is supportive to women. And I think that it's supportive to women because everything happens out in the open. Like everybody is talking live, you see how decisions get made there is no sort of secret backroom. And so there is not a - what do I want to say, it's - I don't know, I can't say of every work environment that I found super unfriendly to women.

So it's hard to contrast, but I just feel like at Facebook we're having the conversation much more and we are also like going out of our way to structure relationships and social bonds between the woman in engineering like we are talking about it and I think that's probably because Sheryl is there and is writing books like Lean In or giving her Ted talk. But we are vocal about it, we have a great community of technical women at Facebook who are sort of - and the range of opinions span widely, because women aren't at all the same. So it's one of the theme, like men are always trying to be like what should we do to make Facebook more friendly, and I'm like, well, I've devastated to tell you that women don't all want the same thing. We're not all like and be friendly to woman does not mean any one thing, but I think it is fundamentally a meritocratic environment, it's an environment where ideas win, it's an idea - it's an environment where code wins arguments. Yeah, I have found it an extremely great - but probably the best thing about it just is the fact that we are so open about having this conversation and it's so welcome to talk about it. Whereas I think it was a little bit, it's just kind of taboo in a lot of places. It's just you don't talk about it. If you bring it up, you're a whiner, right, and at Facebook like we talk about everything. So have you personally made any choices that have allowed you to have a very intense professional career and also to have a family? Have I made choices? Or how about - has your family made choices, have you...? Well, I think I just - the best choice I made by far, I didn't even know I was making and that was who I married. I mention I married fellow alum.

And my husband has been incredibly supportive of my career and after we had our second child, he actually made the decision to be a stay-home-dad. And the fact that he is home with our kids really does enable - it brings a balance to our lives. And it allows me to hit it that much harder in the office, in the workplace. But I think another choice I made, coming back from Texas to California when we did part of the reasoning for that was because we were also starting to think about having kids and I thought if we have kids, I want to be near my mom and my mom happens to be here in Palo Alto. And so in some way, it's just dumb luck that I have - that I happen to marry a guy who is going to be willing to be a full-time parent, that I happen to have my mom living in a location that was like the best place in the world for my career to be. So I do feel like I won the lottery, multiple times, let me be clear. But that support that I've had from my mom, from my husband, definitely I mean, women make it work, I know a woman who is also a VP of Engineering at VMware, who is a single mom of three in Cambridge, Massachusetts and I'm in awe of her, but I think that my path has definitely been easier, because of the support I've had. So is there other advice you would give to young people who are looking ahead, I mean, this is often a discussion I have with my students. They are really struggling to figure out how to have it all. Yes.

How do I end up having a challenging career, wonderful friendships, have a family, it all feels like these are full-time jobs. Yeah, I mean I will tell you what I did. I mean I think I did luck out in who I married, but I think you should absolutely be looking to marry someone who is supportive of your career, who believes in your career and who is going to be supportive of the home life because it does take two people. I think you should hate housework. I hate housework actually. And I can't tell you the number - and I - seriously and you are a Computer Science major from Stanford, your earning power is incredible. An hour of your time is so valuable you can pay someone to do the dishes and the laundry. We do not because my husband is psycho and doesn't want help in the house and so he does the dishes and laundry, but that's the deal like I would pay someone to do it if it were my job. So I can't say how many women I know who are - who share my husband's view that they don't want household help and I think that why, like the things in my life like I love my job, like you should pursue work that you love that you're passionate about because otherwise it's not worth it, it's a trade-off. And you should have a family if you're excited about having a family and like those things that you should be passionate about and then you should just like ruthlessly eliminate from your life, time spent on things that you're not passionate about whether that's like I have always chosen to live close to work, so I have the shortest possible commute; time in the car is dead time.

And I outsource housework. If my husband didn't want to be doing the housework, I would be paying someone to do it. I just think that like if you want all the great stuff then be ruthless about and not wasting your time on stuff that doesn't matter and household really doesn't matter. So let's go back to the culture of Facebook. Yes. I love the fact that there are all these mottoes around: move fast, break things. Yes. Okay. What happens if you really break something? I mean, it's really nice to have it on a nice poster around ... Yes.

... you know but what - give me - can you tell us some stories about some things that have actually broken and what's happened? Because we all know that people are watching as you said it's kind of like a big game and the incentives are there and people see what happens. If someone sees, if someone does something and it doesn't turn out and they get punished, they are not going to take a risk. So can you give us some examples? Right. So I think that was just a poster on the wall for me until I broke something myself. So I'll tell that story. This was when I was in bootcamp, it was week five and so I was brand new to the company, nobody knew me from Adam and I noticed that my first few weeks in bootcamp I was assigned a bunch of bugs to fix and it turned out like three out of the five were obsolete. Someone had already fixed them or the code had been refactored and just wasn't an issue anymore. And so I realized that there were actually a lot of bugs in the bug database that were kind of - they were Kroft basically. And if you - like, bugs don't sound like a very sexy thing, but if you are determined to ship high-quality software, one of the things you need is information about what is your state, like what is the state of my software.

And a bug database can be a faithful representation of the state of the world, if you are really scrupulous about your bug hygiene and about closing out bugs that - at about tracing a bug, so about looking at every bug, and about closing out the ones that aren't relevant, including ones that exist, but you're never going to do anything about because they don't matter. And so at VMware, I built a name for myself by being like this monstrous triager of bugs, like I triaged like 1,000 bugs in my first month at VMware. And at Facebook, I was walking into a much more mature environment that already had a lot of bugs open, and then I obviously had sort of reflection in hindsight that it's just not - what I did at VMware, didn't really work for me, but also wasn't incredibly scalable. And so I wanted to try something new, I wanted to try automating bug triage in a way and writing a script that would essentially if a bug had been untouched for three months. At Facebook, things move so fast, if something has been untouched for three months, it's a pretty good sign that is irrelevant. And so after three months, it would sort of just post an update to the bugs saying, hey, is this thing still relevant or can you close it out? And that would trigger an e-mail to everybody CC'ed on the bug and then if another three months went by, and nobody responded to that or updated the issue, then it would just auto close it. So I was building this I call it the Task Reaper and I was writing this script and I was testing it and I wasn't careful enough in my testing and I accidentally - just was moving some code around, it was copy paste error, and I moved something out of an If-block, and you're probably guessing where this is going - I accidentally ran the test on live data and it pinged 14,000 bugs, because the first time you run this, there is a lot of untouched bugs. It pinged 14,000 bugs. And that means it generated email to everybody on the CC list of all 14,000 bugs, which meant I basically launched a denial of service attack on our e-mail infrastructure. So this brought the bug system to its knees, but it also brought email to its knees for the whole company, which is like a pretty big deal.

I mean, it doesn't take the site down for users, but it means like our sales team can't interact with customers or like the engineers aren't - they can't do code reviews. The email's kind of the lifeblood of the company, even more so in those days. We use Facebook itself more for communication now. And so I was just kind of blankly terrified, like, what I have done and I really expected to be tarred and feathered for this. And there was definitely a vocal company response to what just happened to me, why do I have 200 emails from the Task Reaper. But what was really visible to me, two things were really striking to me. One was the Exchange team and the bug tools team, like the people I expected to be most mad at me, actually just sort of rolled up their sleeves, waded in and started fixing the problem. They spun up another process to process the e-mails faster; they threw extra capacity onto the server, like they just got in the trenches and had my back. And I was a stranger, they didn't know me like, they didn't like I had no right to be messing with their systems, they just had it. And then on the flipside the communication from the company, there was no one saying how dare you, you're new, this you didn't - why didn't you ask permission, nobody said that.

People were saying what was the point of this? And they were saying could you may be send a digest email instead of an email for each one? And they were like - but no one acted like I didn't have the right to try, like they clearly didn't like the result that their e-mail was down, but they didn't act like I shouldn't try. And it was like a light bulb went on in my head, I realized oh, my God, this is what lets this company still innovate, even when it's already - at that time it had just passed 0.5 billion monthly users and like if any company - like it was just success for them. What is - why did Facebook still have appetite to mess with this thing that is so successful, why do we launch like complete redesigns of the homepage, when the homepage is already the most trafficked piece of property on the web? And I sort of got it in that moment; it was because we are willing to take risks. We are willing to face up to the consequences of failure, if it was in the spirit of trying for something, of trying to innovate. So that was just like an amazing experience for me and I think, it's not that you can come to Facebook and like be incompetent or do things wrong all the time. I mean, I think we - there is a lot of feedback and actions if that's happening. But we view it as every employee's right to try things to take risks. And we expect you to deal with the consequences of your failures, but we also rally and help you and have your back when you fail and we expect you to rally and have our backs when we fail. So it was kind of magical introduction to the company. Yeah.

How long you have been there when this happened? Five weeks. Five weeks, okay. Great. Thank you. So I guess you've now had opportunities for other people who work for you to make similar sorts of errors? Yes, yes. I tell them the story. Yeah. It makes them feel better. Yeah. Does it make people want to try more things? I mean do you find that there is a boldness that comes from saying that this type of...? Yes, yes.

Oh, I really think so. I mean writing "be bold" on the wall is one thing. I think that's helpful, but like actually seeing that you can try things and have them not work out and then still thrive, definitely contributes. What I tell bootcampers in my on-boarding is listen, innovative ideas by definition look like bad ideas. If they looked like good ideas, they would be obvious ideas. And so to be innovative, to be unobvious, something about them has to look stupid or dumb or impossible and so what gives people the courage to try dumb ideas? Especially because for something - like what's your expected success rate? I mean, if you are a VC, you're going to fund 10 companies hoping that one of them will be a huge success and the other nine may not, but that's a great portfolio strategy, right? You'd rather fund 10 companies trying bold things with the outcome that one of them is twenty-fold successful than do 10 kind of incremental easy things. And if you're a company, you actually have exactly the same calculus. As a company, you would rather be trying 10 things and having nine of them fail and one be twenty-fold success - like that's just good ROI for anybody who owns the portfolio. The problem is it's bad ROI for the actual individuals on the nine things that failed. And if you are a human being, I really think the biggest thing that kills innovation is not that companies suddenly wake up one day and say, oh, I don't want to innovate, right.

Like companies all want to innovate, they want their employees to be bold and to try risky things. The problem is innovative ideas are going to fail at a pretty high rate and if you are a human, you hate failure. You are going to try something like the Task Reaper, you'll have a horrible experience and not try it again. Maybe if you have exceptional grit, you'll try two things that fail. But you have to be willing to write - try 10 things in a row that fail, if you are going to be a successful entrepreneur. And I do think that's what distinguishes successful entrepreneurs is that particular brand of insanity that is ready to keep trying and keep trying and keep trying in spite of failure and keep believing in yourself. But ordinary humans do not and so I think that you just have to provide incredible cultural back pressure to support people in that instance of failure if you want them to keep trying crazy ideas. That's terrific. So let's imagine that you are flashing back in time and you're now a student here at Stanford again. What advice do you wish someone had given you? I mean, imagine you were sitting in the audience watching someone who was in a position at the Director of Engineering at a really impressive company, what things do you wish they had told you that would have had an influence, both on your career path, but also on your mindset? Oh man.

Is that a hard question? Yes. What advice would I have given myself? I think that - you hear this all the time, but be brave like the world is your oyster, so like you really don't have any bad choices right now, you cannot go wrong. Now is the time in your life to take a lot of risks. I think that I would have said, don't - I wouldn't have listened, but I would have said, don't worry about what you are going to be when you grow up, just be and it will come and find you. I feel like when I was in high school and college, so many people were like follow your passion, follow your dream and I was like, how do I find my passion, how do I find my dream? No one had particularly good advice for what to do to figure out what you're passionate about. And so I would say it's okay not to know. Just keep trying things and you will find stuff that you're passionate about and excited about. I would have told myself not to be afraid of writing code for living, I was afraid of that. I was afraid that other people would be better at it than me, I was afraid that I would be buried in detail and not get to be strategic and above the fray, and I learned over time that writing code is actually one of the most strategic things you can do. And every time I meet a college student who wants to skip being a software engineer and go straight into product management, I'm like oh.

It's not that you won't someday want to do those things, want to be a manager or a product manager or a designer, but you learn so much from writing code. The devil is truly in the details, the what is possible, the what is feasible, the systems thinking; just embrace it, relish it. You may or may not - you may love it so much that you do it forever. You may not do it forever, but every minute you spend coding is going to make you 1,000 times better at being a manager or a product manager or anything else. That is better - your time is better spent, even if your goal is to do something else, your time is better spent writing code as preparation for it, if you want any career in software, any career, even CEO or sales or whatever it is. So I would not have been so anxious to sort of figure out what was after software engineering. I would have spent just - I would have lived more in the now, I guess, or I would advise myself to live more in the now, I would have had trouble taking that advice. Well, that seems like a great segue to open up the questions to the audience. So who would like to start? Great. So I definitely liked what you said about Facebook being flexible and kind of adaptable to handling failure and encouraging innovation, but where do you guys really draw the line between when okay, great job, you're doing bold things, you're making some mistakes, but that's fine versus like this is terrible, like you need to stop doing this, like how do you guys distinguish between those? Because it's definitely a grey area from my perspective.

The question is where do you draw the line? It's one thing to say you encourage people to fail, but where do you say okay, this is too much, dial it back? So I think that first of all, there is some bright lines. I mean, there are things like ethical violations that will just get you walked out the door, right. So there are certain failures that are not I tried something innovative and it didn't work out, but that are just like I did something wrong, that are just wrong. So I think that when we think about sort of booing people in the event of failure, we definitely think about helping them solve the problem. We try to actually encourage people to fail fast, right? Like we don't want you to spend - like it's terrible to spend a year developing a product and it is the wrong product. It's much better if you spend a week or even a night at a hackathon, throw a prototype together and we can see on face value, hey, this is not great, like let's course correct, let's try something else. I think if the failures - some of it is repetition too, right? If you make the same mistake over and over again, that's just a problem, right? If you're bringing the site down because of a careless error that, again, wasn't because you had an idea and didn't like - even my error was not like that the Task Reaper was a bad idea; my error was like I failed at execution, like I made a mistake in how I was testing it. If somebody was making that same error over and over and over again and wasn't learning each time - life is feedback loops also, like you learn and grow by doing stuff and seeing what happens and taking the information and then trying it differently the next time, like life is iteration, life is spirals upwards. And so you've got to be making different mistakes each time, not just repeating the same failure over and over again, I guess. So you obviously were a very successful manager at VMware, that's why they made you manager.

Then going back and not being an engineer, you have - and you also have the benefit of time. So where do you feel you actually met - your talents are most suited, towards the managerial side or towards the engineering side? And if Facebook wants to promote you to be a manager, do you go there or do you stay engineering? What's your ...? Well, I'm a manager at Facebook. So as a manager, I spent the first six weeks in boot camp, just - everybody does that. But I mostly manage at Facebook. But I will say - oh, I failed to repeat the question. The question was is your time better spent as a manager or as an engineer and how do you decide? I will say that at VMware at the end, so I was always in the engineering department and I rose to Head of Engineering and then in the last year and half at Vmware, we decided we were going to have a business unit structure and so I became the General Manager of desktop business unit, instead of just the Vice President of Desktop Engineering. And being a General Manager meant that in addition to managing engineers, I also managed product managers, marketing people, I owned a P&Land I visited lots of customers to help sell our products, who were buying desktop products. And one thing I learned from that experience is I love building stuff for customers and talking to customers, I don't love selling to customers. And I consider myself a good manager and I think I can manage people who aren't engineers, but at the end of the day, engineers are my people. And I love managing engineers better than managing marketing people.

And because I think at heart I want to make stuff, I want to create things, that's just my - what drives me and even as a manager, I want to make teams that make stuff and so that's what I love to do the most. Back there, yes? Can you talk about the decision-making process at Facebook and how that fits into the corporate culture? Talk about the decision-making process at Facebook and how it fits into the corporate culture. It really depends on the decision. I think that if you code, you know that you make decisions with every line of code that you write. And so Facebook is an environment that has the expectation that everybody is going to be making a lot of decisions and so our job as a management team is actually to plug every individual into as much information as possible about what's going on so that everybody can make the very best decisions. It's a very distributed decision-making environment. And Facebook takes that a little bit - that sounds like pretty good, that sounds like mom and apple pie, right? But Facebook takes it a little bit to an extreme in the sense of we've worked very, very hard to avoid - to give teams, small teams, a lot of autonomy to pursue their own destiny and so we have kind of eliminated a few of the checks and balances that are more standard at other software companies. So you generally don't go to other teams - like as you get larger, more teams have a stake in what you're doing, but you generally don't go to other teams and ask their permission. You generally build stuff and test it and then if that manifests as problems for someone else, then you work it out and fix it. And so I like to say it's a "try catch" model of decision-making, not an "if then".

And so that has for the most part served us very, very well because it means that just people operate a lot faster, we move a lot faster and even if we're heading in the wrong direction, we figure that out and course correct faster. Can I just ask you how involved is the senior management, like Mark, in decisions that are very granular? We know of companies that have a tremendous amount of top down control - if a button gets changed its color, the most senior person is looking and saying I would rather have it blue than yellow. How involved is Mark in that - all of those decisions? Mark will definitely give you advice about pixels. And when I say advice, I mean make decisions. So Mark has organized the company so he can spend the bulk of his time on product and product strategy and he has very able lieutenants that allow him to do that. And he basically has set up his calendar so that each day of the week has a theme and the theme is one of the departments. So one day it might be mobile, one day it might be platform, one day it might be whatever, and then there will be a block of four hours and just teams rotate through that block and just present stuff to him and talk stuff through with him. And it's amazing because he is probably the most gifted product thinker in the company and maybe in the Valley, maybe in the world. That's probably stretching it, but certainly in our space. And he - and so like having a half hour of his time is just like amazing and, but it's - you do have to learn because he can operate at so many layers of abstraction at once, like sometimes he will say like, yeah, that's not going to work, I think we should try it this way and you have to kind of unpack and figure out like is it your CEO, is he wearing his CEO hat when he is giving you that feedback and that's because of strategy and how you're going to impinge on some other part of the product, or is that - or is he wearing his designer hat or his PM hat, which is about how users are going to receive the feature or how it looks.

And he will jump between those layers of abstraction and you just have to try to follow him. I mean, and just make the most of it, make the most of every second of his time that you get. So yes, he will be very involved and he has structured his time to be involved. But he can only pay attention to so many things at a time too and so there are - if he is not paying attention, like he doesn't expect you to sit around and wait for him. He expects you to run forward actually while he is not looking, and if he sort of comes back to you a month later and finds that you haven't moved from where he left you, he will be pretty disappointed. Great. More questions back there. Thanks. I think Facebook's culture of kind of move fast and break things is really admirable, but at the same time I think for the average user, Facebook hasn't really changed that much in the past few years, its photos, its chat, its status. So I'm wondering how much the kind of culture of building things is focused around building new things at Facebook versus changing and improving what's already there? That's a good question.

The question is move fast and break things is admirable, but it seems like Facebook hasn't actually changed that much in the last couple of years, so how much is it incremental versus sort of big new disruptive things? We certainly feel like we have changed a lot in the last few years. I definitely think we have sort of drastically improved on things like photos and messaging and we have tried in a lot of ways to drastically improve. We tried two new UIs for the newsfeed, which is one of the bolder things that we do, because changing the homepage always makes people mad. Actually, one of the biggest places where we have innovated a lot is in our mobile technology stack and how we're delivering mobile products. And I think actually the - I don't know if you had a chance to try it yet, because it only runs on a few Android phones so far. But we delivered a product called Facebook Home, which is a really social take on the phone experience and it sort of takes the phone and it turns on its head and puts humans at the center of the phone experience and tasks, makes them more secondary. I think we feel like that was pretty innovative and it's still very much 1.0 and it has a lot of work that needs to be done. But I think we feel like we tried something big there, and we feel good about it. So I do think it's a good question because we have got to always sort of push to do some of both, right? We are serving a billion users and so on the one hand, we don't want to neglect them, we want to keep sort of - and we don't want to take anything for granted, we don't want to feel like okay, we can ignore photos now, we have been there and done that because a photos upstart could come out of nowhere and eat our lunch if we're not kind of constantly striving to make the photos experience better. So, I don't know, yeah, I think we just try to do both.

Great. Another question? Yes. So what are the next things - big things in mobile that you just talked about? What are the next big things in mobile? Well, Facebook Home I think will be the next big thing. We haven't yet done it. I mean, we've shipped the first iteration and from here we have to iterate until it's really awesome. The other interesting thing about mobile is that mobile growth's enormous. A lot of it is happening in the developing world. And so users are flooding onto smartphones; in the next three years, we're going to have orders of magnitude more smartphone users than we've had in the past. But the interesting thing is the smartphone that those folks in Latin America, in Asia, in Africa, the smartphone they're going to be using is going to be a Gingerbread phone. It's like a three year old version of Android.

And so I actually think the race there is not who can do like the snazziest, richest, craziest feature on mobile, it's actually who can deliver the most streamlined, speedy, lightweight, light touch, smooth experience on a very low-end, low power device. And you've got to be - in this country data is basically - I mean, you pay for your data plan, but then once you've paid for your plan, you don't think hard about how much data you're using. When you pay by the minute, when like the - when data is as large a part of the spend as it is in parts of the developing world, you're pretty - like if we do over-fetching in the newsfeed in the U.S., it just doesn't matter too much. Like we send a few extra stories, so that your feed loads faster, it's slightly sub-optimal, you don't really care. That happens in Africa like we're costing some - our users money when we send too much data. So thinking about how to do that and then we just think about phones as the mechanism by which a whole next set of billions of people are going to get online and are going to get on Facebook, because frankly lots of people are going to have smartphones and feature phones that have never and will never have a computer. And so what is an entirely mobile centric Facebook experience look like that isn't a second screen device, a second device in addition to your laptop. So those are the things we're thinking really hard about. Can I ask you a question about privacy? Yes. I'm sure nobody in this room cares about privacy, right? I mean, this is always an issue about what people put on Facebook and how other - who gets to see it and how that information gets used.

I think a lot of people - I've had many, many conversations with people who are extremely conflicted because they so want to have the social experience, it's - they're drawn to it and then they're also afraid of it. Yes. They're afraid about when I put something out there that feels really personal, who is going to see it, how is it someone going to monetize that? Yes, can it be re-shared beyond that... Exactly. Yep. I think that's - I don't want to say that's - I think it's really hard. I think there are no textbook answers. I actually think it's one of the most fascinating product challenges that Facebook has and I think most other companies have not bellied up to that challenge fundamentally, right? Most of the products that we think of as social networking like a Twitter or a Pinterest or even an Instagram have an all public model. There is no privacy model. You go in knowing that whatever you put on there is public for all to see.

And I think Facebook is different in that we're trying to create a private or a semi-private sharing model where you can control the audience. And that's like a really difficult thing to do and it's very hard to hit the sweet spot between what will people understand, what's usable, what's easy. Everytime we sort of try to dial up and give lots of privacy settings, we find ourselves in this quagmire where we've given users controls that they don't want or can't use or that paralyze them. And so I think that it's just a space where we need to continue to innovate and I think others have tried innovating the space, I think, and have figured out that like oh, it's not that Facebook is dumb or ill-intentioned, it's actually that this is really hard. So it's one of like a very sort of fundamental hard design problem. How to help people understand the difference between audience and distribution, how to help people understand who sees what, and I think we've just got to keep trying and I think to the extent we are successful, we create value for our users that they will see in the world. I will say I think sometimes it's over blown. We do surveys like a very large proportion of our users use the privacy settings. The idea that somehow like the unwashed masses just don't know how to use Facebook and don't know how to use the privacy controls, that's actually meaningfully not true. Like how many of you in here have never changed the privacy defaults on Facebook? Right.

Alright, so that was like one person. So how many have changed the privacy defaults on Facebook? Everyone. Alright, like how many of you have never used Facebook? Okay, one, two, three, alright. It's good. I'm counting. So I think there's your answer, right? Like I think actually people are quite savvy, they do care very much about privacy and that means that it's one of the most important features that we have, and it's one of the most important products that we have, and it's something where I think we're just not done innovating, but it's not that we aren't trying, it's that it's hard. Great. Back there. So how has the culture evolved or changed post IPO? The question is how is Facebook's culture changed since the IPO. And the answer I think surprisingly is that it has not very much.

This is actually my third IPO that - the company I worked for in Texas went IPO; VMware, of course, went IPO while I was there, sort of twice actually, and now Facebook. And I think the difference - and the IPO caused real change at those earlier two companies and not so much at Facebook. I think part of it is that Facebook was simply larger and more mature. I think Facebook is really pretty unique in how long we waited to go public and so we were pretty gelled as a culture, as an organization, as an executive team. I think that honestly there had been, because of that long wait, there had also been a number of opportunities for employees to sell equity on the private equity market and so there were a set of people who already had great wealth from Facebook equity and so it wasn't like sort of overnight people started phoning in. I also think actually we have optimized for hiring people who are not there for the paycheck. I mean, we - it's a company that handsomely compensates people, but we're trying to hire people who really care about what they're building and who are at Facebook for the opportunity to touch a billion people and to make stuff for them. And we get to keep doing that and that hasn't changed. I want to say the milestone of hitting a billion users actually got as big or bigger hoopla than the IPO itself. So I think we were really worried about that and actually it's been kind of not that big a deal.

Facebook's engineering is very centered in Silicon Valley, especially compared to other companies that have local engineering centers around the world. So what's Facebook's approach to kind of Silicon Valley or central engineering versus having local development centers around the world, especially regarding proliferation of Facebook's culture which is a strong component? What's Facebook's - the question is what's Facebook's approach to regional development centers, we are very Silicon Valley centric especially in contrast to other companies. This is true. We have only three other engineering offices outside of the Bay area. One in Seattle, one in New York and one in London. The New York and London ones are pretty new. And we're looking to grow those - all four of those sites, but I would definitely say that we would rather have a few large sites than many small sites. It is very clear to us that working across long distances, geographic distances is just hard because making software is fundamentally a team sport. There is no software of significance ever that has been built by a single person acting alone. Facebook itself, like Mark enlisted his roommates within a week, right? And so - and this is one of the areas in which college will kind of mislead you actually because you're taking a lot of these yes classes, where it's actually cheating to get help from somebody else and then in real life what we expect you to do is get help from other people.

And it's really hard to collaborate with someone who you can't look in the eye, to resolve a dispute or a - just a miscommunication. And it is really, really hard to communicate with someone who is in a distant time zone from you where you're overlapping hours of awakeness are not that many. So Seattle was our first remote office and we deliberately - it's the training wheels of remote offices, alright? It's like the exact same time zone and it's like a two hour flight, so you can get up there for a day trip if you have to and that's - our strategy is basically to go slow, to travel a lot between offices, to throw lots of money at all the parts you can throw money at, like having really good VC equipment that's readily available. But we just want to see it be super successful first in the places we are at. What we don't ever want to do is have an office in a region for the sake of saving money. We will just never hire an engineer - like that just won't happen. So, the regions, I mean, Seattle, New York and London are obviously not low cost locations. We are there so we can bring more talent to the company. But I think we are aware that like working in - at a distance from headquarters is hard and we need to do more to support the folks in those offices. Is it important to have offices in other places for just regional specialization so that you understand your customers in different parts of the world? Or do you find that you can develop a product that...? I feel like I should say yes to that, but honestly, no.

We actually did throw a few engineers into Japan. We found Japan a really hard market to crack and people just who were using their phones in very different ways and reasons - wanted to use social networks in very different ways and so we did establish a small Japanese engineering team and product team around really cracking into Japan. That's the one country where we have ever had to do that. I think for whatever reason, Facebook, just connecting and sharing with other human beings has ended up being kind of a universal value proposition, like if I didn't see it myself, I would have said it's impossible to make one piece of software that is used by so many people from so many walks of life. But no, we have not needed to sort of physically visit every region of the world to develop product for the whole world. Great. Another question, yes? The early stage measures of success of Facebook were quantified by number of users. You, as head of R&Dat Facebook, how can you do a deeper quantification of success? How do we measure success? This is an excellent, excellent question. We are a very metrics driven company. It's actually quite interesting because we're sort of a vision led company, but also a metrics led company.

We just mix both. And as a product team, you always have two sets of goals. You will have some that are just milestone goals that are just like we're going to do X because we believe in X and we're going to deliver it, we're going to launch it. And those are usually goals about launching a new product and then you have a whole bunch of metrics that you're tracking and that you're trying to drive up and to the right. And it very much depends on the product. So Facebook as a whole used to measure monthly active users, now we're measuring - now we pay more close attention to things like daily active users or a metric that's even harder, which is L 6 of 7, which is a term we made up, which measures how many people are coming back to Facebook at least six out of seven days of the week. And that's an actually even more difficult number to reach than daily active users, it's a smaller number than daily active users, but still a large absolute number. And then it breaks up team by team, so the photos team will have goals, we've had different types of goals; we've had goals around volume of photo tags or volume of photos. We have goals around the number of people, participation rate, the number of daily active or weekly active people uploading photos, photos per uploaders. Messaging similarly has goals along those lines, participation rate.

We are probably more interested in reaching spread than volume per user. I'm more interested in 40 people uploading a photo each than I am in one person uploading 40 photos. But it differs at different points in time. Newsfeed, we spend a lot of time thinking about clicks, likes and comments as measures of engagement and we also measure time spent. That's a big one for us. And on mobile, we're looking at more quality measures, we're looking at things like crash rate, memory consumption, star ratings in the Play store, in the App Store and - but I will say that a lot of the time when it comes back to decision making about product features, we're very often more metrics led than I think outsiders realize. A lot of the times when we were trying to decide to go one way or another and you sort of sit out there and scratch your head and go Facebook, why did you do that? It's because we tested both options on our users and we figured out which one drove the metrics up more. And we have this very interesting philosophical debate inside Facebook often, which is how do we know we're really doing the right thing for users? And I think that one thing we do deeply believe is that ultimately we can't be too didactic. We have to be a little humble about how people use to choose the site and if people use the site to share funny cat videos, then it's not quite our place to sit in judgment on that and to decide like that, that's bad, but baby announcements are good or like definitely - probably people in this room, probably people would say baby announcements are bad and funny cat videos are good. College students maybe have different values.

But I think that at the end of the day, we look ourselves in the eyes and say more people spending more time on the site means we're creating more value for our users. And if we're not doing that, then we are failing our users and if we are doing that, then we're probably doing right by them. And that makes certain things tricky, especially when we get really conflicting - like when on the one hand like your gut or your intuition or even what people say with their words is telling you go this way and then where people are spending time on the site is going a different way. And I think you've got to kind of let go of your ego and say the users actually are not children, they're operating in their own best interests. And if what we're doing were sort of really bad or is like a guilty pleasure and sucking them in, like that doesn't keep - that doesn't retain users over time, right? Like if you create an addictive experience that really sucks a lot of time out of people, but it's ultimately like bad for them, they leave; they won't stay with you. I really think people are adults. And so I think if we were out to sort of create experiences - when we create experiences that are bad for our users, it's very visible. They stop coming back, they spend less time, they leave. And so we really - I think we really have every incentive in the world to just deliver a lot of value to our users and at the end of the day, the fact that we can measure it, the fact that we can say yeah, I shipped this feature or I fixed these crashes and now users are able to spend more time. Like you get that very - it's like a hit, it's like you did something great for them and you can see it in the numbers.

I'm sure you will agree that this was absolutely fascinating. Please join me in thanking Jocelyn.