

One Knight in Product - E98 - Irene Yu

Mon, 12/20 10:19PM 36:29

SUMMARY KEYWORDS

product managers, technical, coding, programme, people, skip, dev, students, questions, tech, technical skills, product, software, understand, dev team, interviews, concepts, developers, technology, database

SPEAKERS

Jason Knight, Irene Yu



Jason Knight 00:00

Hello, and welcome to the show. I'm your host, Jason Knight. And on each episode of this podcast, I'll be having inspiring conversations with passionate product people. I'll be talking to thought leaders and practitioners in and around product management to help you build the right products and build them right. If that sounds like the sort of thing you're into, let's develop that relationship, you can head over to OneKnightInProduct.com. After this where you can sign up to the mailing list, subscribe on your favourite podcast app or follow the podcast on your favourite social media platform and guarantee you never miss another episode again. On tonight's episode, we get technical and wonder if learning out of that hello world in Python is really going to help product managers bond with their engineering counterparts. We reflect about what being technical really means, why it's important how far you should go, and what you should focus on to give you the greatest chance of success. We also ponder the glorious future where hiring managers don't ask people who will never have to code in their day jobs to do a coding exercise in an interview and wonder why they even started doing that in the first place. For all this and much more, please join us on One Knight In Product. So my guest tonight is Irene Yu. Irene's a former graphic designer turns software developer turned company founder and educator. Irene started coding at 10 and has worked at a glittering array of tech companies including up and coming Everything Store Amazon, before getting bored explaining basic architectural diagrams to confused product managers and setting up skip level, which is aiming to help product managers and non tech startup founders become more technical, so they can understand the trade offs of technical decisions before presumably ignoring them. Anyway. Hi, Irene. How are you tonight?




Irene Yu 01:36

Hey, Jason, I'm so happy to be here. Thank you for having me.



Jason Knight 01:40


No problem. So first things first, you are the founder and lead mentor at Skiplevel. So for the record, what problem does Skiplevel solve for me?

 Irene Yu 01:50

Oh boy, where do I start? Well, I created Skiplevel as the go to place for non engineering tech professionals learn the technical skills and knowledge that they need to succeed in tech. And to feel more confident most importantly, to feel more confident. Yeah. So you know, first, lots of professionals that work in tech who aren't those don't actually have a technical background. So many of them come from a business or a marketing background. And they're kind of just thrown into the pit of fire, to kind of figure it out themselves. And frankly, a lot of people struggle with that. And it's not that people don't want to learn or that they're lazy or something, right, it's just that there isn't really a good and easy place to actually learn technical skills and knowledge that's actually useful for the non engineering tech role. So you know, there's plenty of people who take a coding class, which I talk about all the time, in order to be more technical. And that's actually the least effective way to become more technical. So most technical education that's available today were created for engineers. That means most of the technical education either go too deep into coding into building, that it becomes confusing, or it doesn't provide enough context on fundamental tech concepts, or they don't explain technical terms and lingo well enough, or just doesn't provide information that is actually useful for the non engineering tech role that works with dev teams, but is not necessarily a dev themselves. So skip level is meant to be the safe place that helps professionals increase their technical chops in a way that's comprehensive, and easy to follow.

 Jason Knight 03:31

Yeah, that sounds good. And obviously, it's interesting. Yeah, I know, we've both come from development backgrounds, ourselves. And I'm thinking like, as you're talking about technical books of things like clean code, and all these other types of books that you have, they're very much focusing on like the best way to write classes. And the best way to just structure your code. And I guess, what you're saying is that that's not really the best way for people to learn how to be technical at all. But do you think that those are kind of follow on books from this type of thing? Or do you believe that really, the goal is to get them to a certain point, and then they kind of stop and do other product management stuff instead?

 Irene Yu 04:09

Yeah, I think that the ultimate goal is for them to be technical enough to be able to ask the right questions. Yeah. And I know that sounds a little bit fluffy. But let me explain. So every team and company does technology a little bit differently. You know, they architect their software a little bit differently. They set up the infrastructure a little bit differently, their shipping schedules are a little bit different, right? So the sign or the yardstick by which we measure someone who has technical is that they're able to go to any company, talk to anyone about a product, and that they are able to ask the right questions to then understand how technology is being done on that specific team or that product. So it's kind of like teaching someone to fish versus giving them the fish to feed them once right. The second thing is that, you know, technology changes all the time, it changes very quickly. There's always something

new coming out onto the market, there's always a new tool, there's always a new service. I mean, my head is spinning, right? But if you have enough tech fundamentals, and can understand tech lingo well enough, you can know what to Google? And what questions to ask to understand what is this new tech, and what problems it solves? And how to speak about it. So, you know, learning technology, getting technical skills is actually a long game. Yeah, there's no magic bullet that will immediately make you a super senior engineer, right? It's really a combination of active learning. And just by doing it, you know, through every single day working and building technology and working with web scenes, and skip level is a really great precursor to kind of getting a head start and that learning process. So it may take someone three years to figure out what an API is, and what are the components, right, some directors of product management that I know, never learn it. And that's something that you can actually learn in the Skiplevel programme in like, 15 minutes.



Jason Knight 06:07

Oh, there you go. I'll sign up tomorrow. But it's just then aimed primarily at product managers. I mean, we've talked about product managers already. And I know that we said an intro about startup founders as well. So is it really focused at that kind of person? Or do you think that this is just anyone in an organisation that needs to have some kind of interaction with developers like salespeople and other types that maybe you wouldn't necessarily consider being the target for this sort of thing?



Irene Yu 06:33

Well, yes, and no, the product management role works the closest with dev teams without having to code themselves. So the programme was created with the product management role in mind. But it's really for anyone and all people that wants to feel more confident in their technical chops. So that means being able to speak. So think, to write to communicate more technically, and have a better understanding of what goes into building software. Right? What's the proper way of building software? There are a lot of students in the skipable programme that come from different backgrounds. So there's the non technical side of founder that's very popular, I had one who was looking to hire a CTO, and really didn't know what to look for when they were interviewing, right. Yeah. So even taking the first module on infrastructure and applications was really helpful for the person to be able to ask the right questions and also follow along when they actually get an answer. And then ask more probing know deep dive questions. There's students in marketing people who are copywriters, people who are in a more client facing role, people in sales, people who are interviewing for a tech role, who decided that they could really use better technical chops.



Jason Knight 07:48

Yeah, makes sense. And I can see that working for things like pre sales with particularly technical products as well kind of given them that head start, like you say, but you started out as a developer yourself as the day you started programming at 10. As did I. So we're basically the same person already. We're exactly the same. Yes, exactly the same in all ways, shapes and forms, but you obviously live with this stuff. But what was it that got you into programming in the first place? What got you sort of started at age 10 down that development path?



Irene Yu 08:17

Oh, man, what a blast from the past. Actually, it was because I was obsessed with my blog when I was a kid. And I actually started out my background in graphic and web design. And then web development before I moved into software engineering. So before I wrote a single line of code, I actually learned how to use this design programme called JASC Paint Shop. Not sure if you've heard of it.



Jason Knight 08:41

Paintshop Pro?



Irene Yu 08:42

Yep. PaintShop Pro!



Jason Knight 08:43

Yeah, I remember it fondly



Irene Yu 08:45

Yeah. So this was in like, the very early days of the web, like in the 1990s, before Photoshop really took the lion's share of the market. So I started doing web development in my teenage years, because, you know, my blog just had to look amazing. And it just had to be the best. Yeah. And it turned out to be a really great childhood hobby, because I started off my dev career doing web design, and web development, and just a lot of creative technology for ad agencies before I moved into software engineering at larger companies



Jason Knight 09:16

by and then you moved into those larger companies and into the world of software engineering, and you've worked at a bunch, but obviously, the most notable that you've worked at is Amazon. Yeah, we've all heard of Amazon. What sort of stuff were you working on Amazon then as a developer there?



Irene Yu 09:30

Yeah. When I moved into software engineering, I actually started working at a personal finance company that helps people manage their money and get access to financial advice. So I was working on their front end application. I was mostly fixing bugs and working on continuous improvement tasks. Then I moved into the advertising org. And this was a really big jump

because I went from working on simple front end applications to my team fully owning databases, infrastructure and dealing with scaling. And, you know, we're talking about hundreds of 1000s of people visiting our software at any given moment. So scaling was definitely a big priority. Yeah. So I started building out the front end application of an app that allows UI UX designers to build their own landing pages without the help of it, though. And then 10 months into my time, I actually moved into a back end development. So I owned a product that lets customers schedule test drives, with our car advertising partners, like Audi and Mercedes for free. And actually the first test drive campaign launched in the UK.



Jason Knight 10:37

There you go, maybe I saw it.



Irene Yu 10:40

Yeah, too bad you didn't get to take advantage of it, because I thought it was an awesome campaign. So then, I was working on advertising. And then I moved into the internal documentation team. And I built a lot of work on both the front end and on the back end.



Jason Knight 10:57

Yeah, and that's obviously really interesting kind of traversing all of the different parts of the stack as well. So again, like back in my day, when I was still developing stuff, I think having an understanding of all the different parts of the of the stack, and the the front end database, the architecture, the back end, is obviously really helpful, and maybe something that not all developers get to do. And I think it's something they all should do. But it also feels like that's something that's really helpful for you from a kind of programme perspective, from a development of a training course type perspective, if you're trying to bring someone up into this the you've kind of got that wide ranging experience. Did you find that that was really helpful in informing what should go into these training programmes?



Irene Yu 11:40


Yeah, definitely. I mean, my experience definitely helped me a lot with the content and the course at skip level. I mean, I just learned a time from some of the smartest engineers I've ever met. And I, I 10X-ed as a dev just like breathing the same air as them.




Jason Knight 11:59

Yeah, that's fabulous. And I guess one question, then that I have to ask, and I think that maybe some of those brilliant devs that you've worked with, and some of the people that you've learned from and some of the teams that you've been in? I mean, I think it is fair to say that not all developers have a reputation of being particularly patient with non developers. Like some developers really look down on non developers, to be honest, yeah, like, based on some of the people that I've worked with in the past, they get kind of a little bit annoyed when people don't

understand what they're talking about or can't talk to them on their terms. And you don't have to file on say, Twitter or other social media platforms to see product managers digging at developers and developers digging at product managers, and everyone kind of just so comments. Yeah, turning that guns on each other. And, you know, ultimately, they should be uniting against their real enemy, the salespeople, right. But the point is that you've got this kind of dynamic. And obviously, that's not a dynamic that you subscribe to, because you're trying to help. Yeah, people that are in that situation to become more technical. But what specifically then gave you the urge to step away from development and to cross that line? And try to educate these people in such a constructive way? Because not all developers would have done that, right?


 Irene Yu 13:11

Yeah, right. I started skip level when I was a dev. And when I was an engineer, I started tech mentoring a product manager that had approached me and asked for help, because she was really struggling with just working with us, like the engineering team. Yeah. And she couldn't keep up with discussions, and she just felt very unconfident at work. And frankly, I can tell because some of the questions that she asked during technical meetings really show that she lacked a very basic understanding of software. Right. And you're totally right, you know, members on the team that was on the team would get a little bit frustrated. So I started working with her to get her kind of up to speed on the software fundamentals. So I was meeting with her about twice a month. And sitting with her during product ideation, she's coming up with ideas to kind of talk through the technical feasibility of what she's thinking about, like the product and the feature. And after working with her for a while, I distinctly remember that she was signed to ask much better questions during product engineering meetings. And through that, you know, the whole mentoring experience with her, I realised just how many people in tech struggle silently with low confidence because they don't understand how technology or software works. And they don't really know how to communicate with that. Right. And I think that that's totally understandable. Because even if you're not a dev, you're still building software. Right? So you're still a very important part of the team. Yeah. Now imagine how much better communication and collaboration would be if everyone was actually on the same page and speaking. And it was around the time that I realised that there weren't really resources out there to teach technical skills and concepts to non engineer professionals, so product managers, or designers or non technical sort of founders, most people who want to try to become more technical usually go for a coding pass. And after the coding class, they would tell me, you know, I just took this coding class on Python. And then they have no clue what to actually do with that information that they just learned, right? So coding isn't really, it's not a very effective way to become more technical, because it focuses on going very deep into a narrow vertical and software, when really, we should be focusing on having a breadth of knowledge throughout the software development lifecycle. So once I came to this realisation, that there weren't resources out there, for people like this product manager that I had mentored, I knew I wanted to start working on building skip level. And also because it was really fun, mentoring her.


 Jason Knight 15:50

So that's not a very developer attitude. But I think that the point there is really valid, like I've worked with product managers, myself and my time and, you know, seen product managers out and about in the community that maybe lack some of that background, and like you say,


it's no problem for people to lack a background, it's completely understandable if they come through a different path, like if they've come from marketing, or if they've come from customer success, or they've come via some other path, which hasn't really intersected with development that much. And, obviously, it's very common, if you think of big tech, and you think of maybe some of the big Fang companies or whatever Fang is called now all the companies are changing their names. Like they, that's kind of like the archetype right? That you get these developers that then become product people or these developers that come founders? I guess the question then arises off of that is you've obviously finished with that one person back then you taught that person what they needed to know. And then you started skip level. But as part of that, did you start a mini Skiplevel programme within Amazon and start training other people there as well?

 Irene Yu 16:56


No, I actually ended up mentoring one on one to product managers and seeing a similar results in both of them. And the seeing that they both started to be more confident during technical discussions. But actually, after a tech mentoring the product managers, I started teaching the skippable programme live on the weekend. So the early version of the skip level programme. I taught it to professionals who didn't work on my team. And I taught them on the weekends, it was about five weeks, two or three hours per weekend. And once I saw Yeah, that there was a real appetite to learn a lot of people want to sign up, and that people needed resources to help them learn. That's when I decided to go full steam ahead on skip level.

 Jason Knight 17:39

Yeah, that makes sense. And it's obviously a little bit of kind of emergency market validation there as well, like, so you're starting to become a product manager yourself back at that point to start. So doing that research and doing almost like an MVP of the of the approach, which I think is fabulous.

 Irene Yu 17:55

Yeah, definitely learned a lot about product management, and just, you know, working on skip level, but also talking to a lot of product managers.

 Jason Knight 18:02

Yeah, absolutely. And I think that's fantastic. You know, but I guess one question I do have, and we've kind of touched on a couple of times this idea that coding isn't the way to learn to be technical, which is completely fair enough. And I have to learn a lot of other stuff, which again, is completely fair enough. But what kind of stuff that we specifically talking about, like, like, How deep did they go, they kind of just get an idea of different types of databases and want to see ICD pipeline is or are they going like mega ultra deep into some really hairy concepts and almost getting to the stage where they could become a entry level DevOps engineer, like how deep do they go into that?



Irene Yu 18:40

That's such an awesome question, Jason. So I talked about going for breadth of knowledge over depth of knowledge, right? So going deep, is going deep into coding, which is one vertical in the software development life process. But it's actually important to see actual demos of technical tools and concepts, and even do some of that work yourself. But so what's the actual philosophy of skip level? So if coding isn't going to actually make someone technical? What should they focus on instead? So I teach that they should be focusing on improving for specific technical skills. So what are the skills? So the first one is having a broad awareness of different technologies that are available in industry, right? So for example, API's, message queues and job schedulers. And this kind of helps you with understanding the availabilities, but also the limits of technology. The second skill is understanding technical trade offs. So as soon as a dev gets a feature or a requirement, they start immediately thinking about how to build it. So the things that are going through a dev's mind is how do I build for maintainability? How do I make sure that this is secure, right? How do I make sure that this is reliable and flexible? How do I make sure that the system is fault tolerant? How would I build the system? So being able to think through these types of technical trade offs will go a really long way? Being able to empathise with them. Yeah. And then there's the third one, which is being able to speak, communicate and understand technical jargon. So this one's pretty obvious. The last skill is having a familiarity with the process of building software, also known as the SDLC, the software development lifecycle. So understanding this process, and all of the tools and concepts that are involved, will help you understand how long it takes to build a feature and what could actually go wrong and how to speak about it. So these are the four skills that skippable aims to improve for students, none of them involve coding. So I often use you know, real life product scenarios in the video lessons to kind of explain how different technologies are used to tackle what types of product problems by also do demos throughout the course to kind of show concepts like spinning up a MySQL database, querying a database, and then also other hands on activities. So those hands on activities are actually really popular with students because it allows them to actually do some of the work themselves along with questions that allow them to critically think. So for example, in a module on data, students are given a product scenario and then asked to design the database schema, and then create the actual database along with some mock data. You know, so going back to your original question, the course does go deeper, where we need to in order to build more empathy, and be able to visualise these technical concepts and be able to have a detailed conversation. But it never goes so deep that it becomes too confusing, and really, completely useless the role.



Jason Knight 21:34

But do product managers in your opinion, really need to be technical. Like if we go back to some of these interview guides that you see for product management jobs, and you see all these, again, Fang or whatever it's called companies sitting there and saying that you have to do either estimation questions, which isn't coding or actual pseudo coding or coding exercises within product management interviews to kind of get in and get the job. Now, what we've been speaking about so far has been very much about well, that's not a good measure of a product manager. So do you really think that these people that are interviewing people like this are doing a good job in asking product managers to be pseudo coders?



Irene Yu 22:10

11:00 PM 22:10

Well, the short answer is no. I don't think it's fair at all to do coding interviews for product managers, because there is simply because product managers aren't expected to code in the role, right? I mean, I mean, dev teams certainly do not expect product managers to ever code probably be their worst nightmare, right? Yeah, absolutely. I mean, it's already a nightmare for them. To have product managers do it is just terrible. But you know, I will just talk about why there are technical questions for PMS during interviews in the first place. Companies, especially big companies want product managers that are technically skilled, because they know what a difference that makes. So work with a product manager that can communicate technically, and understand what goes into actually building an app. And you know, knowing that building an app is actually really hard. And sometimes it's just not as easy as you know, just build it, you know, and having very hard and flexible deadlines. dev teams work better with product managers are technical, because it cuts down on a lot of inefficient or frustrating communication. And technical product managers tend to write better requirements because they understand what devs have to go through in order to build something. So that's number one, why there are technical interviews for PMS at all. But I'll tell you why big tech companies do coding interviews for product managers. It's because the tech industry hasn't figured out a better way to test for technical skills during interviews. Yeah, so testing for whether a product manager can code is really just a proxy for whether this product manager is technically skilled. And this is a fallacy because coding and being technical. They are not synonymous. So a technical product manager that I know who doesn't have a background in engineering, but is very technical. He uses a saying that goes, you don't have to be a basketball player to know what a free throw is. Yeah, you know, you can still know the rules of basketball without being a basketball player yourself. And the same goes for software, you can still understand how software is built and what goes into it without being a dev, right? So it makes sense. So instead of testing for coding ability, a better way to test product managers for technical skills is to focus on the four technical skills that I talked about. So broad awareness of available technologies, being able to speak, communicate and understand common terms and concepts, being able to think through technical trade offs and understanding the software development process. So basically, you know, product managers do not need to code and shouldn't ever be expected to code and knowing how to code is not the same as being technical



Jason Knight 24:42

I'm detecting a theme. But I've been in situations before with very non technical people trying to specify database technologies because they've seen some articles on a website or they've just heard something been said in passing and it sounds good to them. And there's obviously a spectrum of light Being technical or not technical, and you can be any point on there. And I guess what we're saying is that when people way, way down one end of that spectrum, they, they probably shouldn't really be trying to make any technical decisions at all, because they don't really have any knowledge to back that up. Are you worried that there's this possibility that people could take your programme become more technical, and then go into their baseball bat bouldering into the standard one morning or into the sprint planning and say, Hey, I know enough technical stuff now. And you should do this and start looping over specific solutions? Or do you think that's not really a problem?



Irene Yu 25:37

The whole? Gosh, I hope not. I hope not. I mean, you know, that's something that does happen

sometimes, especially when the product manager was a dev themselves, and a part of them maybe misses coding, and so they want to make the decisions. Yep. But this is not something that I suggest, you know, students do, right like to go into one on one combat with a dev team like that is not helpful at all. But with the knowledge that students get from the course, they will be able to better come up with suggestions for technical solutions. Now, I do encourage product managers to voice their opinions, right? Not just product managers, but really anyone working with a dev team. Yeah, they should always voice their opinions, of course, humbly and respectfully. And most importantly, they should know what they're talking about. Right? Otherwise, that's just not listen. But But ultimately, the Dev has the most context about the application. Right? So they own the house. Yeah. So product managers should be part of the conversation, they should contribute their ideas, the suggestions, but ultimately, let the dev team take ownership of their expertise.



Jason Knight 26:43

Yeah, I guess as long as they keep on the right side of the tracks, and I want to be happy and no one's going to be arguing too much for anyone else. But if you consider, like the golden journey of skip level, so someone starts out, they enrol on the programme, they go through the process, they come out the other side, hopefully with some nice certificate. And they've kind of got to level one of what is you've got to offer? What do they then go and do like is that them done? And they're just kind of fine. And they just kind of pick everything else up as they go? Was there like a future path for them? Via you, or via some follow on service that you'd recommend to them?




Irene Yu 27:19

Yeah, well, I want to first say that success looks a little bit different from person to person, success isn't gonna be the same for everyone. And it depends on their skill level and the expectations of their job or what their goal is, right. So for example, success might look very different for technical writers versus someone who's in a client facing role. And that just needs to be able to coherently speak and explain technical concepts, versus someone who has a non technical sort of founder that just needs to be able to interview a CTO, right? Yeah. So ultimately, again, my goal for students is to come out and be able to fish for themselves, right be able to ask the right questions, and then comprehend what's been said to them or what they're reading or what they're looking at. So that no matter where they move in their career, whether that's looking for a new job or working with a third team. And if a new technology comes out, they can feel confident in their ability to figure things out.



Jason Knight 28:17

Makes a lot of sense. But I guess you've had a few students going through this already, like you've been running this for a bit now. And I'm assuming that you've had a few people come through and finish the programme and go on to do whatever it is that they do in their jobs that they've got, or the jobs that they get afterwards. Are there any testimonials or kind of stories that stick out from the people that have come through the programme that that really make you think this is all been worthwhile?


 Irene Yu 28:43

Oh my gosh, yeah, I've actually been blown away from some of the unsolicited messages that I've gotten in the testimonials I've gotten. So I just had a student in the skippable programme. He's a product manager. And last week, he sent me a direct message after he had finished a course. And he told me that he used the knowledge he got about software architecture and software diagrams to create one for his team for his product, because one didn't already exist well, and that the engineers loved it. And this was something that he wouldn't have been able to do before skip level. So it wasn't just for the team, but he wanted to use a softer diagram because he was having trouble kind of visualising the system. So it was also just, you know, for him, but it's kind of useful for the entire team. I had another student who was a business development manager before Amazon Web Services. And after she finished the first section on cloud computing, she sent me an email and said, I finally understand the difference between EC two and ECS. And that it was so much easier to understand than the internal Amazon training. By the way, I have taken the internal Amazon training, it is very verbose, it is very difficult to actually understand. So for her to say that she actually now understands easy to an ECS, which is, you know, pretty high level stuff is absolutely priceless. And she said it's absolutely priceless. So this was information that was directly related to her work, right, her job. I had another student who has a background in strategy. And he has zero and I mean zero experience in tech. So Facebook suffered an outage, and I had written a post about it on LinkedIn. And it had a lot of technical concepts or some technical jargon, the student comments and said, I cannot believe that I actually understood the majority of what you brought. And before skip level, I would have absolutely been scratching my head. So you know, this student just needed a place to start. Yeah, he needed general context and a foundation. And so that's what he got. Right? That's success for him. The last student was one of the students that I taught live, and he was a newer product manager. And one day, and this was after he took the course, he texted me. And in all caps, he said, one of the devs just asked me if I can merge into live, and I actually understood him. I cannot believe it. So the same student didn't say no. Yeah, I hope he said, Yes, yeah. And you know, so that made him really excited that he actually understood and that he didn't have to make something up, right. The same student also started interviewing for a new PM role. And one of the questions that he got was, how much do you know about the software development lifecycle? Yeah. And guess what, that's an entire module in this couple of programmes. So he was able to confidently answer I know everything about the software development lifecycle.



Jason Knight 31:43

Excellent. So it sounds like you're already fighting the good fight and bringing everyone up and making them more effective at their jobs, which is fantastic. Hiding the good fight, it's worth fighting the good fight. But imagine that someone hasn't started the skip level yet. Or maybe they can't afford it, or, you know, they just need a little bit of advice to get them started before they come along and find you. So some product managers sitting there struggling a bit to build that relationship with their dev team, maybe not so confident in their technical skills, aside from advising them to sign up to skip level, what's like one piece of advice, you'd give these people to get them started and maybe help them build a little bit of their own confidence.

 Irene Yu 32:21

I do have a few actionable tips that I share, one of these tips here, and this is one that I really

love. And this tip goes for everyone, anyone can use it, even if you're just interviewing or you're just starting out, or, you know, you already have a job in tech. So instead of taking a coding class, you know, because I rail against, you know, taking coding class, in order to be more technical.



Jason Knight 32:43

I remember that!



Irene Yu 32:45

Yeah. Right. So instead of taking a coding class, take a few tutorials from popular cloud computing services. So cloud computing platforms, offer many services and tools, and each of them come with their own tutorials, right. So some popular cloud computing platforms, I suggest checking out Amazon Web Services. Firebase, as are some of the tutorials can be a little bit advanced. So I do suggest starting off with the simpler ones. And as you go along, if there are any concepts or terminologies that are used in a tutorial, that you don't understand, to do just a little bit of research on them. And this is a lot better than taking coding class because it exposes you to more parts of the software development lifecycle. So for example, how do I set up infrastructure? How do I pull down a code base? How do I edit the code base using a virtual control tool? How do I commit the code? How do I ship the code? How do I use a database? Right? I do have a few recommended tutorials to start out with on the skip level website. And it's on the resources page. So skip Loboda co slash resources.



Jason Knight 33:54

Very good. Excellent advice. And what's next for skip level then? I mean, have you got any plans for world domination? Or maybe the first taking over the Amazon training programme?



Irene Yu 34:03

Yeah, world domination? No, I think Jeff Bezos and Elon Musk are working overtime in in that department. I don't think they'd be happy if I joined. Yeah, so a few things skip level is actually working directly with teams and companies to train the employees. So we've recently been training product management teams at different companies. There are also bigger plans for the skip level community. Currently, only those who enrol in the programme can join, but we will be opening up a smaller section of the community to everyone who wants to become more confident in their tech skills. There are plans for technical workshops and live q&a sessions with me other tech mentors and product managers and technical sort of professionals. So yeah, stay tuned. It's gonna be a wild ride next year, but I'm super excited. And I guess that's my version of world domination.



Jason Knight 34:55

Okay, well, you can only dominate the world if you start dominating bits of it first. Guess start

small. So where can people find you after this if they want to chat about technical stuff or talk about skip level or maybe even sign up.



Irene Yu 35:07

So the best way is to find me are on social media, Twitter and LinkedIn. My handle is I am hiring you. That's @iamireneyu, my DMs are always open and I love connecting with and hearing about people's stories. So definitely reach out. The other way to reach me is through the scope of a website at Skiplevel.co. If you click on the talk to the instructor button, then you can book a time with me to chat about your career goals or find out more about Skiplevel.



Jason Knight 35:36

Sounds good. I'll make sure to link that in the show notes. And hopefully you'll get a few more technically incompetent, but soon to be technically competent product managers coming your way. Well, it's been a fantastic chat. So obviously really appreciate you taking the time and sharing your thoughts, opinions, and maybe some of the ways that we can all be better being technical. Obviously, we'll stay in touch but yeah, that's for now. Thanks for taking the time.



Irene Yu 35:57

Thank you so much, Jason, this was super fun.



Jason Knight 36:02

As always, thanks for listening. I hope you found the episode inspiring and insightful. If you do again, I can only encourage you to hop over to OneKnightInProduct.com, check out some of our other fantastic guests, sign up to the mailing list or subscribe on your favourite podcast app. Make sure you share with your friends so you and they can never miss another episode again. I'll be back soon with another inspiring guest. But as for now, thanks and good night.