So You Want to be a Software Engineer

Transcript:

00;00;07;08 - 00;00;20;04
Kristen
Today we're here with **John** Crepezzi, principal software engineer, and we just wanted to take some time to find out what his job is and how he got into the profession. So, **John**, can you tell us a little bit about yourself?

00;00;20;29 - 00;00;45;09
John
My name is **John** Crepezzi and I am, like you said, a Principal Software Engineer. I've been a software engineer for,I guess, 15 years or so as a professional software engineer. I've worked in a variety of different companies I've worked at. If you've heard of the local website patch, I was early engineer. And then after that I worked at some other sites you might know, like Square and Tumblr.

00;00;46;01 - 00;00;56;00
John
And then most recently, I work at GitHub, which is a software company that makes essentially software for software developers, which I can talk more about.

00;00;56;10 - 00;01;00;02
Kristen
So you're helping with the software for the people who know software?

00;01;00;18 - 00;01;19;21
John
Yeah. The easiest way to describe GitHub is, developers, when they write code, they don't normally write it in isolation. They write it as part of a group of developers They need a place where they can collaborate and they can store the code and they can push their changes to a central place. And then other people can kind of look at them and review them.

00;01;20;09 - 00;01;25;04

John
Because we're working, like I said, as a team. And GitHub is that place for most companies.

00;01;25;24 - 00;01;36;07
Kristen
I think that's really interesting. Some people might think that you work with coding, you work with software. That's something that you do by yourself. But you're saying that this is a very collaborative profession

00;01;36;07 - 00;01;36;19
John
for sure.

00;01;36;20 - 00;01;43;06
John
Yeah, I think most movies have like sold us on the like image of like the hoodied Dark Room with the.

00;01;43;27 - 00;01;45;09
Kristen
Yeah, the hacker, like the.

00;01;45;10 - 00;02;06;20
John
Yeah, the hacker in the corner. So while there are definitely times like that, that's not how real software gets made. Real software gets made by groups of people that are constantly talking to each other, having meetings, a lot of which are over Zoom now because our workplace is fully remote and having planning sessions and figuring out where they want the business to go.

00;02;06;20 - 00;02;16;16
John
A lot of the efforts that we're doing are not things that can be accomplished in a few nights with cans of Red Bull. While that's super fun, these are things that sometimes take multiple years to do.

00;02;17;02 - 00;02;28;05
Kristen
I hear that you need to have some skills working with people, and you also need to have some patience, because if you think you're going to get it done in a night, that's probably not going to happen.

00;02;28;19 - 00;02;51;04
John
Yeah. Or you'll burn yourself out trying to get on a deadline and really like we need to make it so that our career is something that's sustainable for everybody. So typically when we're developing software, we're operating as part of three groups of people and those are typically Engineering, Product and

Design. Those three come together to make what you would normally see as a website on the Internet, something like Facebook or Twitter.

00;02;51;15 - 00;03;06;28
John
So Engineering are the people that are writing the code and making sure the systems are working properly. Product are the people that are making sure that the thing that we're building actually has a, you know, market fit and that we're building the right things. And then Design is making sure that, obviously, that the things that we're building look good.

00;03;07;12 - 00;03;35;02
John
All right. They're easy to use for people. So those three together really are what makes software. So there's a lot of collaboration there, but then there's also a lot of collaboration inside of just engineering by itself, because typically when you have that arrangement of engineering, product and design, the biggest group of people is the software developers. Because people that work in a bank, you probably need more people that are tellers than you need people that are closing mortgage loans or something just like that.

00;03;35;02 - 00;03;57;02
John
There's, just the the dynamics work out in such a way where you typically need more software developers than you need product people or design people. So inside of the engineering side, we have a whole way that things have to work. We have to not only write code, we have to write code that is maintainable in the future and that people looking at it backwards can kind of understand it when they read it.

00;03;58;01 - 00;04;10;13
John
So we spend a lot of time thinking not about just like, how do we make this thing work, but how do we make this thing work in a way that is not going to make our lives really, really bad or make the next feature that you want to write. Impossible to write.

00;04;10;28 - 00;04;16;21
Kristen
Right. So you need to think from a future backwards perspective, also.

00;04;16;21 - 00;04;42;06
John
Always. I guess another thing that we think about a lot when we write code is the stability and scalability of the thing. So that means it's very easy to write code maybe for something that is Facebook, but for only two people. But then you have Facebook for a billion people, and that's much, much harder to do, not only because there are more people using it, but also because there are more people using it at one time.

00;04;42;14 - 00;04;49;06
John
So you have to figure out all of the different things that could go wrong when multiple people are doing something at once right.

00;04;49;06 - 00;04;52;02
Kristen
And they're they're trying to do different things at the same time.

00;04;52;15 - 00;05;07;07
John
Yeah, all different things at the same time. And some of those things might conflict with each other or cause edge cases and so what we call them in the way that you wrote your software. Maybe I should step back and just say, like, what's writing software is?

00;05;07;07 - 00;05;07;26
Kristen
Sure.

00;05;07;26 - 00;05;10;29
John
Yeah. Maybe that would be helpful for people to lay a context.

00;05;10;29 - 00;05;38;08
John
So when you look at something like a website, something like Facebook or Twitter behind that, I think pretty much everyone knows that there are some code right? Someone had to write that thing. But typically the thing that you're actually interfacing with, the thing that you're using is actually multiple pieces of code that are coordinating with each other. So if you go to something like Facebook, there is code that's running on your computer, which is the thing that's happening inside of your web browser.

00;05;38;19 - 00;06;02;13
John
But then there are some things also happening far away in a computer farm somewhere, and that thing is called the server side. And then even that is talking to more layers of more code that typically that more code is something like the database software. So we have developers that are writing all different levels. So some developers are focused purely on the, the part you see, so to speak.

00;06;02;13 - 00;06;27;25
John
And some developers are focused on the, you know, the nitty gritty details of how the system works and scales. And when you're writing code, you're typically inside of a code editor. And while it looks like a bunch of random characters, when you sit down and read it, a lot of programing language is kind of read

almost like English, but the names of the there are definitely like curly braces all over the place and things like that.

00;06;27;25 - 00;06;45;17
John
But when you actually sit down and read it, for the most part, you can kind of read what it does as a sentence. I guess just to say that what can seem really daunting up front. Writing code actually is like a very, very organized system that has been built up over the past 60 years of people writing software.

00;06;45;17 - 00;07;04;21
John
And that's the most exciting thing for me, is that these things that we're building are just so complex and so many layers of, I guess, interactions between systems, but that in order to use them you only need the top part. It's like when you go on a carousel at the fair or whatever, you don't need to know how to carousel works.

00;07;04;21 - 00;07;05;02
Kristen
Right!

00;07;05;02 - 00;07;21;10
John
You just need to know that you get on it and then it rides you around. Software is kind of like that in that you can be on the surface and then you can get more interested and like no matter how deep you dig, you're never going to reach the bottom because there is just so much to software and it'll be not daunting.

00;07;21;10 - 00;07;27;10
John
And then you can dig deeper and you're like, Oh my God, I didn't know about all this stuff, and that you're actually still just like, you know, on the temp or whatever.

00;07;27;10 - 00;07;51;07
Kristen
Right? Right. So programing language, software, language. It sounds like when I first learned English grammar, it didn't make sense to me. So as a small child, you're just putting words together and it might not mean anything or people can figure it out. But when you first start to write sentences and you need to know where there's a comma go which is capitalized, where does the period go?

00;07;51;07 - 00;08;01;24
Kristen
How do I use a semicolon? And those are all rules that are different in each programing language, but are the basic component. How you do language stays the same.

00;08;02;15 - 00;08;18;26
John
Yeah. With the curly braces and the semicolons, for example. Like when I'm teaching people to program or showing people programing for the first time, a lot of times they're like, How can you remember, like, where this curly brace goes? And it's funny because like after a week doing it, they're like, there's nowhere else for this curly brace can go.

00;08;18;26 - 00;08;42;04
John
It's like asking someone like, How do you know where the period goes? In a sense, it's like goes at the end, like it just of you learn it so quickly and it is really like another language, sorry, grammar. And that's why they call it programing language. And I guess on the topic of programing languages, there are so many of them, there are so many programing languages and there are some programing languages that are only used for specific purposes.

00;08;42;14 - 00;08;47;14
John
And typically this is not for everyone, but like typically a developer kind of has to know one.

00;08;48;03 - 00;08;48;10
Kristen
Okay?

00;08;48;18 - 00;09;14;02
John
Or possibly two, you know, like it's not these are not things where you're expected to kind of know every language but then, maybe you go to a new job and you need to learn a second one. But then you'll also notice that kind of like romance languages in the real world in language that a lot of the programing languages share the same constructs, they share the same ideas, and that's because they're actually influenced by each other.

00;09;14;02 - 00;09;37;11
John
So maybe a new programing language comes out and you can actually go in Wikipedia and search for examples of Ruby programing language. And right at the top it says, Influenced by. And there's a list of writers that Ruby took ideas from. So these things are all intertwined. They're all interconnected, even to the point where, like the name of the method that you use to do certain operations is the same between all the programing languages.

00;09;37;21 - 00;09;55;26
John
Of course that makes sense because if there isn't a word that everyone already knows, that they know, there's a certain thing just to use that word, you don't have to invent a new thing and everyone gets

that. And I guess if a programing language comes around that doesn't do that, then it's probably not going to work out because it's like teaching everyone in the world something completely different.

00;09;56;04 - 00;10;05;05
Kristen
Right. What languages do you program in? And you said that you use different languages for different kinds of things. Can you give us some examples?

00;10;05;19 - 00;10;27;29
John
Sure. Yeah. I program typically and this is as of right now, JavaScript, Ruby and Go and I can talk about what those three are. But then in the past they're programed in PHP, programed in Java, C and C++, just like a bunch of different languages, Erlang. And okay, so I'm the three languages that I do.

00;10;28;13 - 00;10;50;24
John
It's interesting, these three languages because they're actually a kind of four, three different things. So I'll just talk through like what those three are. So JavaScript is the only language that can run inside of a web browser. When you're using a website and you maybe notice this more in the past ten years, that when you it used to be that when you click something, the whole page would reload and there'd be a different page that came in.

00;10;52;07 - 00;11;16;04
John
But then recently, all of a sudden you like click something and things like slide out of the way, or maybe new data comes in even without reloading the whole page. That's all JavaScript. JavaScript is in charge of all of that stuff, everything to do with that. So JavaScript is good for people that want to learn to make what we call the client side of web applications, which is the part you're actually clicking around in.

00;11;17;10 - 00;11;37;21
John
Ruby is made to do the other side. It's made to the server side of applications. Ruby is well known for programing language, but it is, I guess, most well-known because it's highly readable. It doesn't have any of the curly braces like those other languages have. There are no semicolons. There's not a bunch of extra syntax that you don't need.

00;11;38;03 - 00;12;03;22
John
And all the method names are really meant to read like English. Even the way that you write code is very much like a sentence. So Ruby is the server side microbes that is used at GitHub and then Go is a little bit we call it lower level. Okay. If you think of JavaScript as that most high level, Go is kind of the lowest level, which is there's not a lot to go, there's not a lot that you can do with it.

00;12;03;22 - 00;12;30;08
John
But the things you can do with that are very powerful concepts. But because there's not a lot of these higher level concepts in Go, it does turn out that your code ends up harder to read and a little bit more I guess, long form, would be the way to say it. And so you typically want to use Go because it is also faster and you typically would want to use it in a place where you have a very critical but very small piece of code that needs to be fast.

00;12;30;24 - 00;12;40;29
John
And then you want to use Ruby in the places where maybe it doesn't really matter how fast something needs to be. So those three really complement each other. And at GitHub, those are kind of the three that we use.

00;12;41;28 - 00;12;48;04
Kristen
And then if you change jobs and they use different languages, then now you're going to learn new languages.

00;12;48;07 - 00;13;12;08
John
Yeah, yeah. And you'll learn them in a couple weeks when you first start. And you might end up forgetting a lot about the old one or the old one might like evolve in the meantime to a point where you're like, you can hardly recognize that anymore. Oh, and also, if you particularly like the language that you're working in, most jobs do tell you upfront what they're building, things and they'll say, like, we're building a Ruby back end with a MySQL database.

00;13;12;14 - 00;13;22;04
John
And you know, if you know those two technologies and you're comfortable with them, you might choose to go there or you might think, I want to do something new and yeah, go, go find somewhere else.

00;13;22;21 - 00;13;25;01
Kristen
You mentioned that the languages are evolving.

00;13;25;01 - 00;13;25;12
John
Always.

00;13;25;12 - 00;13;34;17
Kristen

So, is this like comparable to the way that new slang enters English language? Or is this totally new rules that develop over time?

00;13;35;08 - 00;13;56;18
John
It's both. It's both. Yeah, it's both. Slang, is a lot of times invented for kind of a reason, right? It's to take, like, a common thing that you say and like shorten it. It's like, yes, just make it quicker. There are things like that. There are things where people are programing and everyone's kind of running into the same problem over and over again.

00;13;56;19 - 00;14;17;20
John
So then the language authors will just add a feature that makes it quicker to write that thing. And then there are some things in programing languages where they're trying to support a totally different use case than before, and now they're adding a brand new feature because at the end of the day, the languages themselves are code. They're more code.

00;14;18;01 - 00;14;18;10
Kristen
Yeah.

00;14;18;10 - 00;14;27;28
John
And that's actually a fun thing too, because this is like really cool thing and programing is like a lot of the languages that you use are actually written in themselves.

00;14;28;19 - 00;14;29;00
Kristen
Okay.

00;14;29;15 - 00;14;43;01
John
They write the first version of the language and then when they want to write the next version, they essentially re-implement the base version. Meaning the code for the language is a lot of times written in the language.

00;14;43;22 - 00;14;54;27
Kristen
So what do you do day to day? Can you take us through? I'm sure this changes if you're working in-person versus working remote or traveling for your job. But give us a typical workday.

00;14;55;16 - 00;15;17;15
John

Yeah. I am a principal software engineer. The reason that the principal title is in there is because we have just like any other job there are levels as you grow in your career. So you start as a software engineer and then you're a senior software engineer, and then you're what we call a staff software engineer, which is I don't know, it's like you're a Boy Scout, your Cub Scout or Wolf Scout like,

00;15;17;20 - 00;15;45;05
John
You know, it's like that's the same thing. So a principal software engineer is one of the highest levels of at least our engineering hierarchy at GitHub. So that means that basically I spend probably like 30% of my time coding and most of my time now figuring out how to direct technical projects or to mentor people or to spend time with people talking about what they want to do with their career, make sure that they're working on interesting projects.

00;15;45;15 - 00;16;17;24
John
So I, I'm not right now at least a people manager meeting like no one reports to me. Most of my job is actually like, people, you know, it's mostly how can I make these developers more productive and more, I guess, like happy with what they're doing? Because you also have to consider tech is a very competitive landscape and you're only going to deliver good software if people that are working at a certain place are happy with what they're doing because what happens otherwise is like somebody works at the grocery store and they don't want to work at the grocery store.

00;16;17;25 - 00;16;28;11
John
So they just like take the bananas, move them over there and then take the bananas and move them over there. And they're just like moving back and forth and not making any progress. the same thing happens in software. Like if someone's not motivated and they're not going to.

00;16;28;17 - 00;16;31;16
Kristen
They're going to do the least amount of effort to keep the job.

00;16;31;17 - 00;16;54;02
John
Yeah, exactly. Yeah. So a lot of being a principal software engineer is finding those places and trying to put people in positions where they'll succeed. Okay. So I work directly with the engineering managers. I actually work, at the principal level, you end up working directly with the engineering leadership team to which the vice presidents and senior vice president of the companies that we kind of work directly with them as well.

00;16;54;24 - 00;17;01;12
Kristen
Is that based on the amount of time that you're at an organization or these are all promotional opportunities as we go along?

00;17;01;18 - 00;17;28;21
John
They're promotional opportunities. It's expected that everyone will get these promotions over time, at least up until the senior level. And then it's about your performance and are you doing the right kind of work? So we have a very defined career ladder document which basically lays out the expectations of each of these roles. And I think it roughly goes when you're a software engineer, someone can kind of hand you a problem.

00;17;29;01 - 00;17;46;09
John
They can come to you and they can say, Here's a problem and here's the rough outline of how you would solve it. And now I need you to go implement this thing. As a senior software engineer, someone can kind of come to you with a problem and then you're kind of expected to figure out the solution. So there's less guardrails and you can do that independently.

00;17;47;03 - 00;18;16;09
John
And then, as a staff software engineer, you're identifying the problems and you're working to fix them. And then, as a principal software engineer, the types of problems you're identifying and fixing are ones that are across multiple teams, not just on your one team. That's the big thing that changed when I went from being a staff engineer to a principal engineer is that now I'm thinking about the whole engineering organization and like, what do we need to build a company that can, by the next ten years rather than thinking about maybe the concerns of my team?

00;18;16;25 - 00;18;21;18
Kristen
So you do a lot more of the people work and that that's.

00;18;21;18 - 00;18;22;14
John
That's a funny thing...

00;18;22;14 - 00;18;33;24
Kristen
Things change as you move like you're getting less of the actual programing is happening and as part of your job day to day but you're, you have a bigger picture of what's happening.

00;18;34;04 - 00;18;59;29
John
Right and you can impact things more and you can make more change through, I guess, the influence and unblocking other people in conversations. Then you can by just writing code. And that goes back to kind of what we were talking about at the beginning is that the guy with the hood in the corner of the room, he can't do much by himself or she can't do much by herself like you need the group of people

and once you have a group of people, then part of the work becomes like, how do you organize and motivate the people?

00;19;01;04 - 00;19;13;16
John
And it is funny, like you're saying, like I always make this joke where it's like I spent 15 years becoming really, really good at writing code just for someday one person to be like, don't write code anymore.

00;19;14;06 - 00;19;14;12
Kristen
Right?

00;19;14;16 - 00;19;37;09
John
No, like that. And it feels it feels weird. But actually that happens in every industry. If you look at it like the person that's really good at the grocery store is a manager and he's not working at the grocery store anymore. And the person who yeah, everywhere happens everywhere. Yeah. At one point I also did in a previous job I was the engineering manager.

00;19;38;07 - 00;20;04;23
John
That is the role where like people report to you and you're having like regular managerial one on ones of people. So you're like setting them up for career success. And I would say my role is also like a little or maybe like a lot different than that. So even though my role is still about people, it's like people, but with a focus on the technical side, while in engineering managers job as people, but with a focus on the career.

00;20;05;06 - 00;20;05;17
John
Yes.

00;20;05;26 - 00;20;06;07
Kristen
Yeah.

00;20;07;04 - 00;20;29;07
John
I love my job also. Yeah. I went to go be an engineering manager and I think I like took a lot of really great things from that. And then I decided it wasn't for me. And then I went back to being what we call an IC or individual contributor, and I found this other path which I kind of knew existed but never had seen before, which is that in programing you can either follow this managerial path or you can follow this individual contributor path.

00;20;29;07 - 00;20;43;21

John
And they're both. I think I thought for a long time that the only way, because this is how it works in a lot of other industries, the only way to grow your career is to go into management in a lot of places, but actually in software you can grow your career as not a manager.

00;20;44;02 - 00;20;44;07
Kristen
Okay?

00;20;44;12 - 00;20;46;27
John
And that's super exciting and that's what I'm doing now.

00;20;46;27 - 00;21;16;21
Kristen
That is very exciting because I think that while management is a good option for many people to move up, it's not going to be the strength of everyone, just like every job is not for every person. And so having an alternate path to move your career forward that isn't necessarily now becoming a people manager, that's good. And also part of why we're doing this career series is so that people can see that there are different options than what you necessarily thought a profession would provide.

00;21;17;22 - 00;21;34;02
John
This is a little bit more rare, but if you wanted to be an individual contributor and not do the people part that I'm, I guess, interested in as part of my job, there are people that do that. There are some developers that are principle engineers that are we call them specialists. They're essentially focused very tightly on one problem.

00;21;34;16 - 00;21;54;05
John
So they're probably not seeing as many people problems. They're kind of like really focused on like the nitty gritty of a single problem. And that's totally an option too, and you can still grow in that role. I guess what's different about software development and maybe like other other industries could learn from this? If you work somewhere you work at like a bank or something, you have like a certain amount of value that you can give the business.

00;21;54;12 - 00;22;15;17
John
And normally in most businesses that value that you're giving is more than you're being paid, probably a lot more than you're being paid. You know, like you're at a bank potentially bringing in like millions of dollars to the bank and you're getting your your normal regular salary. And I guess what happened with software engineers is that they started giving software engineers equity in the company.

00;22;16;04 - 00;22;40;24

John
And this is a pretty typical thing in software. So when you get a software job, you get it's small, a small, very, very small portion of the company. That's part of what we call total compensation. So you go to equity and then you get your salary next to it. What's good about that is that you're motivated to kind of grow this company and feel a part owner and I think that's really great for people, you know, to feel like they're super involved.

00;22;40;24 - 00;22;52;13
John
And I guess that has taken software developers out of the mode of just being, you know, I am at this company maybe a dispensable commodity like some people may feel at their jobs all of a sudden to like, Oh, I'm part owner, I need to make this thing work.

00;22;53;00 - 00;23;02;03
Kristen
Right, right. So there's, there's incentive to make the company successful because you're going to see some of the benefits of that directly.

00;23;02;19 - 00;23;19;20
John
Yeah. Yeah. And, and you feel like you want to stay and that. If you do better, you'll get more. Like, incentives are aligned, right? Mm hmm. Where not a lot of jobs, like we were saying before, about to the grocery store. It's like not necessarily aligned incentives for someone to do, like, ten times better at their job, right?

00;23;20;06 - 00;23;20;15
John
Yeah.

00;23;20;28 - 00;23;54;03
Kristen
Right. Selling more soda isn't going to affect my take home pay. Yeah, that's not going to change. But if I had equity in my company in some way at any amount, then what I do does change. Or some places they'll have other incentives, even if it's not equity in the company. There's other incentives. Like if we meet this goal as a team, then everyone will get a bonus and that makes you feel more, one, that you have more control over some aspect of your compensation.

00;23;54;18 - 00;23;57;12
Kristen
But it also motivates you to hit those goals.

00;23;57;17 - 00;24;19;11
John

Right? I think I just think software is so great. And every time that I talk about it or like I did this, like bootcamp thing last year and like I probably spent a fourth of the bootcamps being like, This is so great, everything's so great. Like, it's just amazing that this is a job. You know, I grew up really interested in computers.

00;24;19;11 - 00;24;39;17
John
Like, I spent a lot of time on the computer and a lot of time like in my room, sitting at a computer and like reading books about computers. And back then I wasn't doing it because I thought it was a job. And I was like kind of startled the first time that I worked for a company, I was like, Oh, look, these people are just as excited about this thing as I am.

00;24;39;29 - 00;25;03;04
John
I think a lot of people think that software is not creative in any way, like re-imagine software, and we kind of like equate it to, I don't know, like our perception of accounting or something like that, you're just going to sit down and like there's one way to write this thing, but software is so expressive, I can read code that someone wrote and get a feel for how they solved the problem or their general experience level.

00;25;03;15 - 00;25;28;07
John
There are many ways to write the same piece of code. We have things called design patterns that essentially are like, I guess, templates for solving or thinking about different problems and code. And the way you lay out systems can be totally different. Different companies work completely different. So I think it is, I think a creative outlet and one that is just absolutely never ending.

00;25;28;13 - 00;25;35;18
John
Like there's more types of paints in software, you know, like you ever touch.

00;25;36;05 - 00;25;46;12
Kristen
Yeah, it's the dichotomy between left brain, right brain or creative versus more analytical. I think it sets people up to feel like, Oh, well, there's more...

00;25;46;12 - 00;25;48;05
John
there are numbers here.

00;25;48;05 - 00;26;03;16
Kristen

Yeah, there are numbers here. So this is math. And math also can be very creative, but there's a set up dichotomy that says your creative stuff happens in art and literature and places like that, and that's not necessarily always true.

00;26;03;16 - 00;26;30;08
John
It's like creative, but it may be in a different language, like it's like creative, but in a world where you can only see the creativity once you understand how the thing works, right? And then maybe someone says something and you're like, Oh my God. It's like, that's the smartest thing I've ever seen. And even aside from the expressiveness of the code, there's creativity in the programing itself because like, you can think of an idea and then you can just make that thing.

00;26;30;13 - 00;26;48;29
John
What else is like that? Where you can just think of something and make it like, I want to make a giant dinosaur sculpture out here. It's like, I don't have all the materials, I don't have like the stuff to do it. But in the, in the computer, you can do anything I can just make anything I want and I can make a system that does, you know, some complex thing or something that's very specific to something that I need or that my friend needs.

00;26;49;10 - 00;27;05;24
John
And you can just do it. You can just make anything. That's the promise of the computer is like the computer is originally built as essentially a generic machine. All right. That's just so cool. There's the generic machine. There's a machine you can tell it to do anything. Yeah, and it'll do it and.

00;27;05;24 - 00;27;06;26
Kristen
It'll do it. Yeah.

00;27;07;07 - 00;27;13;19
John
And if you make a mistake, it'll make a mistake. And that's cool, too. The machine just does what you say.

00;27;14;04 - 00;27;14;12
Kristen
Yeah.

00;27;14;28 - 00;27;17;12
John
Forever. You know, like, that's. That's fun.

00;27;18;19 - 00;27;26;13

Kristen
Are there any specific tools that you use to do your job that someone that might be interested would start exploring?

00;27;27;10 - 00;27;47;24
John
Sure. Yeah. So GitHub is before I worked at GitHub, I used GitHub for probably ten years. GitHub, in addition to being the place where multiple people can work together on code, also supports a thing called open source software development, which is software development that happens between people, I guess, on the Internet that typically don't know each other, right?

00;27;47;24 - 00;28;08;20
John
So there might be like a piece of software that essentially is pushed into the public sphere and then people from all over the world are working on it together. So that's a thing that happens on GitHub. Then I also use my editor where I actually write the code. Then there's a variety of tools to support that code. So there's everything from how does the code get tested?

00;28;08;27 - 00;28;24;25
John
And then once the code is out in the world, how do I observe the code? Which is I guess that's like you can't just put your code out there if it's a really complex piece of code and trust that it's working correctly, you need to be able to kind of monitor it, make sure that it's doing the things that you expect it to do.

00;28;24;27 - 00;28;51;17
John
So when you're writing code, we actually put kind of like breadcrumbs throughout the code that call out to another system that lets us essentially look inside of the box, so to speak, the thing that is running. Just to talk about testing really quick, so developers sometimes write, I guess for the most part write pretty complex and large pieces of software, things that span thousands of files that have to all interact with each other in a certain way.

00;28;52;14 - 00;29;14;12
John
And when you're writing a piece of software like that, it can really quickly become difficult to verify that when you change one thing, you haven't broken something else. The way that we solved that is the way that I guess we solve everything, which is just more code. So we actually write code that checks our code to make sure that it's behaving in the right way.

00;29;14;12 - 00;29;32;18
John
And that's called testing. So yeah, we write, we write a regular code and then we write code that uses our regular code. And I guess in the simplest case, like you could imagine, a piece of software that could

go to a page on Facebook and could click around and do a certain behavior and make sure it works properly.

00;29;32;20 - 00;29;36;14
John
That's the test. We actually write them at the same time that we write the code.

00;29;36;28 - 00;29;47;08
Kristen
So you're sort of making user bots to test the code, like some like what I might do. You're making the code do what I might do to test it.

00;29;47;16 - 00;30;00;01
John
Yes, because it comes very difficult to think of all the things you can do on Facebook. There's so many buttons you can click and there's so many interactions and weird states that you might be able to get into.

00;30;00;01 - 00;30;00;19
Kristen
Right.

00;30;00;19 - 00;30;11;18
John
That these tests essentially can do what might take a human thousands of hours of manual verification. They can do in 5 minutes because it's, you know, their computers, they can do anything.

00;30;12;17 - 00;30;22;07
Kristen
So how did you come to want to be a software engineer? Like, how did you decide that that was something that you were interested in? And then what steps did you take to make that happen?

00;30;22;11 - 00;30;29;29
John
I was really interested in computers and I always tell this story, but am I allowed to say that, our Dad, that is the same Dad?

00;30;29;29 - 00;30;30;09
Kristen
Yes.

00;30;30;09 - 00;30;31;21
John
that. Okay, you're my sister.

00;30;33;08 - 00;30;33;23
Kristen
That is correct.

00;30;33;23 - 00;31;25;08
John
And and we had a computer downstairs and I think it was like a DOS computer. Like DOS is like an old operating system before Windows. That's just like the work of black and white text on a screen. And Dad did not know, I think, anything about code, but he knew enough to start this program called Q Basic. It's an editor and programing language in one. you type something and then you hit F5 and then it just runs the code and then when it's done running your code it just puts you right back in the editor and you can keep typing. That's like Q basic does. And it's actually an implementation of a more generic programing language called Basic. So Q Basic is just the Microsoft made version of Basic, which is why if you've ever heard of a programing language called Visual Basic, it's also just based on Basic,

00;31;25;08 - 00;31;25;18
Kristen
Right.

00;31;25;24 - 00;31;28;07
John
It's kind of like we were saying before, language is like this end for me.

00;31;28;07 - 00;31;29;16
Kristen
Yeah.

00;31;29;16 - 00;31;46;12
John
So he opened up that program and he wrote print **John** and then he wrote go to ten, go to ten and go back to the previous line. And then he ran it and then it just wrote Hello **John** or whatever. Just like a million times on the screen.

00;31;47;11 - 00;32;15;23
John
And I was just like, okay, like, how do we stop this thing? What do we do now? And he's like, I don't know. And he actually had to restart the computer, so we didn't know how to get out of it because he had created in programing called the infinite loop. It's something that will never exit. So then, he restarted It, brought me back in to Q Basic and he opened the help menu, which I think is one F1 back then, like all the help menus and in the help menu there's documentation you can read and I just like started reading it.

00;32;16;04 - 00;32;39;10

John
I don't know why it was interesting to me like I was watching this movie the other day, The Imitation Game, did you ever watch that? Yeah.. At some point, the boy hands on turning this like cryptanalysis,book. And he's just like, I think you'll like this. And like, who knows why? I don't know. I just, like, open the thing. And I guess it worked, you know, just like it worked with my brain, and I just, like, wanted to do all of it.

00;32;39;10 - 00;33;01;14
John
And I think part of it was like, Oh, this computer, like, we had to restart it because my dad didn't know what he was doing. Right now I want to understand what it's doing, and I want to make it so that it's like Legos. It's like my dad showed me these two building blocks and like now he showed me that there's actually like a thousand different building blocks, and I want to put those together and see what they do.

00;33;01;29 - 00;33;22;02
John
Yeah, I was really interested back then in like making something that looked like a professional program. Like, I wanted to make DOS, I wanted to make that! Which is kind of silly, like the in retrospect, like most people wanted to make games and I just like, I want to make the terminal thing. So that's what I would do all the time.

00;33;22;02 - 00;33;38;22
John
And then from there I did more programing books. I had this like memory in my head. This is a very odd thing, but I had bought this Python book and I think we went to like IHOP or something and I like sat in a corner and was just like reading this programing book and it's just, it's just the mom in particular was very good.

00;33;39;00 - 00;33;56;04
John
We went to the library pretty often and we went to the bookstore pretty often and like when we said we wanted a book, she would get it for us, you know? So there was always something new to learn. I didn't actually take any programing classes in high school. I was like just programing on the side and learning for myself.

00;33;56;26 - 00;34;25;11
John
At some point I like thought I was a hacker and then I realized that I was just a programmer, which is is good, probably like a better trigger anyway. So I didn't program through high school but I had friends that programmed and we would talk about it at lunch and things. And then when I went to college at the New Jersey Institute of Technology, which is in Newark, New Jersey, I went there as a computer science major, and it was nice to there's a convenience of I just never thought I was going to do anything with this.

00;34;25;15 - 00;34;47;27
John
That's so it was like, what should your major be in? So, it's like, I just wanted to be in computer science, I just wanted to do that all the time. I know not everybody has that, but that's how it worked for me. So I went to college and I took the computer science classes. I think for the most part, for the first year or two, you don't really get into those real computer science classes for the first two years.

00;34;47;27 - 00;34;55;23
John
You kind of break in that either the intro to computer science or you're working through the college level like the classes that you have to do, like the common classes.

00;34;55;23 - 00;34;57;00
Kristen
The general education requirements.

00;34;57;00 - 00;35;20;02
John
Yeah. So you're like working through that. I'm like sitting in algebra. I'm just like, this feels like high school. And it wasn't until junior year that I was like, Oh, wait, these people do have something to teach me. I don't think that college is necessarily required. Actually, I know that it's not a requirement because we see a lot of people enter, enter programing and enter, you know, that work right alongside me.

00;35;20;02 - 00;35;40;12
John
Some are like actually principal engineers at GitHub that didn't go to college for computer science or some that didn't go to college at all. So there are we can talk about that. There are like totally different paths you can go to like a boot camp is a pretty popular one. Self taught is also another really popular one. Like I could have probably just went to being a software developer instead of going to college.

00;35;41;05 - 00;35;43;01
Kristen
Can you talk about boot camps, too?

00;35;43;22 - 00;36;09;07
John
Yeah, boot camps are they become really popular recently and what they are is all of these things you need to learn to become a fairly proficient beginner programmer crammed into a tight space in kind of like an intensive way. Sometimes these are things you pay for in the case of General Assembly. Or you can look up different schools that do this.

00;36;09;07 - 00;36;28;23
John

They're schools just like colleges, but they don't have accreditation. So they're not colleges and you pay them and then they teach you. And a lot of times it's in person. And then there are also free boot camps like I ran recently, a free boot camp called the All Aboard Boot Camp, that you can find on the Internet at allaboardbootcamp.com.

00;36;28;23 - 00;36;41;25
John
That boot camp was particularly for people from underrepresented backgrounds that were looking to get into tech. So that boot camp was for people that have never programmed before but wanted to learn the basics of Ruby and JavaScript.

00;36;42;00 - 00;36;42;09
Kristen
Right.

00;36;43;08 - 00;37;01;28
John
Boot camps are a great option and actually like a really good option. Even if you're pre college and you're kind of looking and saying like, Is this for me? A boot camp could be like a very quick way to determine, is this something that I want to do? Yeah. You know, before you commit to a major and four years of college.

00;37;02;02 - 00;37;09;00
Kristen
Yeah. And you said that you can also be self taught. Are there things that it would be helpful for someone that wants to teach themselves?

00;37;09;15 - 00;37;35;18
John
Self taught is definitely harder and probably something like a free boot camp is helpful. One thing that's really hard about self taught is that, oh sorry, one thing that's really good about college is not necessarily that they're teaching you, but they have given you the path that have to follow in order to get to the destination, right? So like you go into a computer science degree and they've essentially 101, 102, 103.

00;37;35;18 - 00;38;09;08
John
They've laid out like the path that you have to go down that culminates in you being a viable software engineer out in the world. And that's what bootcamps do the same thing. They basically have laid out here is a set of steps that you can follow that ends with you being a software engineer, right? So when you're teaching yourself, those steps don't exist anymore and you have to find them and a lot of people try to find them and then they get lost because there are just so many different directions you could go in that it's not clear which are the ones that you should actually follow to get essentially to the destination as quickly as you possibly can.

00;38;09;08 - 00;38;24;24
John
So it might take more time or it might get lost or you might get demotivated because you head down the wrong path. Or you might spend two years learning a programing language that no one uses anymore. So instead of doing that go, I guess I would start with like a free boot camp just to get your bearings right.

00;38;24;25 - 00;38;43;26
John
So then after college, I got a job at Sun Microsystems. I did a couple of internships in-between, but I got a job at Sun, which was like a I don't know, it was like my dream job at the time. And there are two things that happened. One, there is this, and I hope he's not listening, but terrible, terrible man.

00;38;44;13 - 00;39;10;04
John
I don't know. He wasn't he was a more senior developer, but he wasn't taking like the mindset of a mentor. He was more like in the mindset of like, oh, this is a person that's like going to displace me or something. And so he was never helpful with that. And then Oracle, another company, actually purchased Sun Microsystems, and that's when I was like, This guy's not helping me and I'm not learning anything.

00;39;10;11 - 00;39;19;26
John
And now I also work for this company that I definitely no intention to work for. I'm sure Oracle's a great company, but at the time I was like, I'm just not what I wanted to do,

00;39;19;26 - 00;39;20;22
Kristen
right.

00;39;20;22 - 00;39;27;25
John
Because Oracle's an enterprise software company, so you're mostly working for like big businesses and stuff. I wanted to work on like things that people use.

00;39;28;05 - 00;39;28;17
Kristen
Right.

00;39;29;29 - 00;39;54;05
John
So I left and then a series of jobs between then and now. But that job of Sun Microsystems was enough and long enough and enough experience to head me into the next opportunity. And then everything since then has just been like building the previous thing. I like to think somewhere in the back of my

head though, that man being kind of terrible to me has like motivated me and helped me to like not be him to other people.

00;39;55;10 - 00;39;56;24
John
I think about him all the time.

00;39;57;00 - 00;40;12;21
Kristen
I hear like there's a lot of movement. Is that typical of someone that's in software? Like, do you move companies that on a regular basis or do people stay at the same company for 30 years?

00;40;12;21 - 00;40;29;22
John
Yeah, that's a great question. And it's kind of a tough one to nail down because there are definitely some people that go to a company and just stay there. Like I know people from college that went to somewhere like Bloomberg or AT&T and just are there and they probably will never leave. That's a totally fine thing to do.

00;40;30;13 - 00;40;50;20
John
I guess the way that I went was more the startup route. Like after I left Sun Microsystems, I went to a bunch of smaller companies, and those smaller companies have more problems than just code. They have problems of like, is our business viable? Is the thing that we're building actually have a product fit. So early in my career I did move fairly often.

00;40;51;03 - 00;41;16;26
John
Sometimes because the company was failing and I was going somewhere else that wasn't failing. Or maybe sometimes because a company had seen that I was a new guy, I was fairly underpaid and then I was given an opportunity to go somewhere else and earn a more like, I guess, market level wage, right? So like early in my career I moved more, but then kind of like the further I got, I noticed that I stayed at places a little bit longer and longer.

00;41;17;15 - 00;41;30;02
John
I think this is not great, maybe advice, but I do say to people pretty often that when you're starting in your career, a lot of times the easiest way to move up is actually to move over.

00;41;30;18 - 00;41;31;09
Kristen
Okay.

00;41;31;09 - 00;41;53;20

John
And I think I think that's probably also true for other industries because like you work at one bank and they look at you and they're like, you are teller **John** Right. And you get stuck in Teller **John**'s and it's like until you go interview another bank that they're like, Oh, here you're going to be desk **John**. And then you're like, Oh, okay, like I want to be desk **John**

00;41;53;20 - 00;42;13;02
John
And it's actually the moving over has kind of freed from the typecast you've been stuck in at your job. So a lot of people earlier in the career move like that. And I guess as you get further in your career the amount of time that you would have to spend in a certain level to move up actually gets a lot longer.

00;42;13;22 - 00;42;32;05
John
So it's more in your interest to stay somewhere and kind of figure out those more difficult problems. Also, I guess another thing connected to that is that a lot of the problems that I work on now you start them and it may not be done for multiple years. I want to see them through, like, so that, that makes you stay longer too.

00;42;32;18 - 00;42;44;06
Kristen
Early in your career when you're moving to different jobs are also picking up all the different things that maybe one company uses one language, one company is a different one, and you're picking up all the different skill sets for those positions.

00;42;44;06 - 00;42;58;24
John
And you're finding your fit right as you go. You're like, I didn't like that. I'm going to go somewhere else. Oh, I like this, ok, I'm going to keep this, but I don't like that I'm going to go somewhere else. And yeah, and then you find that good fit, I guess the length of project thing also can expect what you're saying before about the different levels.

00;42;59;02 - 00;43;15;07
John
If you're that software engineer, that base software engineer, a lot of times things are being given to you and that may be like a week long project that is fully specked out and fully described, so you can leave at any point because like you're going to finish that project and then there's going to be a next project.

00;43;15;07 - 00;43;15;11
Kristen
Right.

00;43;15;12 - 00;43;23;09

John
As you get further, the more you're like, Oh, I'm really like kind of attached to this project, you know, because now I'm really in it.

00;43;23;19 - 00;43;25;06
Kristen
Right? You're more invested. Yeah.

00;43;25;09 - 00;43;25;16
John
Yeah.

00;43;26;12 - 00;43;46;20
Kristen
So if somehow software engineering computer programing was as specific as that, now tomorrow, it's not a career anymore. What could you do with the experience that you have in a different industry? What kinds of skills is better? Yeah, translatable.

00;43;46;29 - 00;44;09;04
John
I think on one hand it's like, that's not going to happen. Other hand, it's like this happened in the late nineties, like the dot com bubble essentially was this, like people that were high paid software engineers had to go find different things to do. So there are a lot of other things that you can do. You have this critical thinking ability and this ability to communicate about pretty intense topics.

00;44;09;12 - 00;44;31;25
John
So that is valuable in all different industries. And then another thing is that you, I guess, are like a problem solver, right? Well, there's not another industry you can go to that would be like computers, in a world where computers don't exist, there are plenty of other industries where a skilled, thoughtful worker is going to be relevant, right?

00;44;32;06 - 00;44;47;04
John
I can't say for sure, but I think I could jump into something else. And while I might not be as interested in it probably would be effective at it because I'm like, I know what it's like to work on a team. I know what it's like to plan projects, and I know what it's like to solve difficult problems.

00;44;47;04 - 00;45;07;28
Kristen
Yeah, and those are valuable skills in multiple professions. And when you graduated, when you left and you had multiple options, you chose software engineer, but you could have chose among what other things where people that were graduating with you choosing.

00;45;08;11 - 00;45;29;22
John
Yeah. I mean, you could have gone the I.T. side, you know, the people that work in an office and make sure everyone's computers are working properly. You could have gone the database administrator route, which is a role where you're making sure that where the data is stored is working correctly and that it's performing correctly. You could have gone into Information Design, which is like how the data is actually stored.

00;45;30;18 - 00;45;50;16
John
There's a new field, Human Computer Interaction, which is like a little hand-waving, but it's a little bit more of an academic field next to computer science. Yeah, I think those are kind of the core ones. And then I guess if you're interested more in like the actual physical computers, you could always go a computer engineering, which is, you know, more of like the on chip design with the actual hardware.

00;45;50;28 - 00;45;59;26
John
As a software developer, you know a lot about, at least as you get more involved in the lower level stuff, you know a lot about what happens in the computer,

00;45;59;26 - 00;46;00;15
Kristen
Right.

00;46;00;15 - 00;46;12;05
John
And how, for example, pieces of information move around inside the computer. But you you don't typically know about like how the circuit boards, for example, are designed, and that's computer engineering.

00;46;12;27 - 00;46;19;27
Kristen
So as a software engineer in your daily life, do you still get asked by family and friends to fix their computers?

00;46;19;27 - 00;46;20;06
John
Yes!

00;46;20;06 - 00;46;23;22
Kristen
Even though that's not necessarily your...

00;46;24;24 - 00;46;28;10

John
All the time. And then there's this website and it's called Let Me Google It for You.

00;46;28;22 - 00;46;34;17
Kristen
Right. And how can you explain? Yes, I work with computers, but I don't do the thing that you're asking me to do.

00;46;34;17 - 00;46;47;02
John
Yeah. Yeah. There's a good chance that you know more about that than I do, you know why, because I most of my time on a computer is spent inside of either in Zoom or inside of the code editor.

00;46;47;02 - 00;46;47;10
Kristen
Right.

00;46;47;10 - 00;46;56;08
John
I really, actually, you know, a lot of times don't do other things because like at the end of the day, I'm like, I want to go home and do something else.

00;46;57;00 - 00;47;04;01
Kristen
Yeah, right. That's an interesting situation because you get all of those questions.

00;47;04;01 - 00;47;04;29
John
Yeah, people come over for Thanksgiving.

00;47;04;29 - 00;47;07;06
Kristen
My phone doesn't work. Can you fix it? Yes.

00;47;07;12 - 00;47;11;10
John
How did you get logged out of your email again? Like you're logging out on purpose.

00;47;11;19 - 00;47;15;06
Kristen
I don't know what you're did. Yes.

00;47;15;20 - 00;47;22;01
John

Yeah. And then they say, what's your password? And they say, I don't know a password. So like, okay, so you want me to like hack into your email or something? And,

00;47;23;17 - 00;47;24;16
Kristen
Yes, yes, yes..

00;47;24;16 - 00;47;25;07
John
Give me something.

00;47;25;18 - 00;47;36;01
Kristen
So give me something. Yes, absolutely. Do you have advice for people that are wanting to enter into this profession and or advice on how to be successful in the profession?

00;47;36;20 - 00;47;59;18
John
Well, first off, welcome, we're excited to have you. And next, I would say take a look at a boot camp and try to get like that base understanding. And then pretty much as soon as you can start applying to jobs, because there is so much more need for software developers in the world, and over the next 20 years, then we have software developers.

00;47;59;25 - 00;48;23;11
John
And that's why that on the previous question, I was like, that's probably not going to happen. It's just because everything is on the computer. We're able to control money movement in the Swiss banking system, all computerized. You have how things get to your house from Amazon, all computers like the amount of industries that are built on computers or that need to be built on computers, it's just growing, I would say go apply somewhere.

00;48;23;11 - 00;48;48;03
John
A lot of larger companies, especially mid to large sized companies, have junior engineer positions, they call them. They're for engineers that are either like entry level or they are essentially intern level, but interns in software get paid quite a bit. Like I think people would be surprised to hear that some of the amounts that interns are making in software development, so go get that job because the only way that you're going to start learning is by doing.

00;48;48;24 - 00;49;08;15
John
And then probably at that job, a lot of the companies that will hire you at that level are going to assign you a mentor or onboarding buddy, if they know that you're somewhat entry level and that person can

be like, they're like your mentor, like the person that you'll go to when you have questions about how to do things or when you want to get better, they'll work a lot.

00;49;08;28 - 00;49;22;18
John
In the beginning and it'll be hard. But you'll learn how to communicate with people, how to work on a team and how to be a software developer. Yeah, I think the most successful software developers are the ones that realize that code's only half of the job.

00;49;22;28 - 00;49;31;14
Kristen
Yeah, you said go apply for jobs. What do I do when I get to the interview? What kinds of things can I do to set myself up to get the job.

00;49;32;03 - 00;49;58;25
John
I'd say have someone look at your resume and make sure that it is as tech relevant as you can make it. A lot of times if people don't have a lot of tech experience, they can do something like go get involved in an open source project and then actually include that on the resume. So it's like something you did that you didn't get paid for, but it is something that can be a resume builder and a skill builder in the meantime, while you're interviewing for jobs and then when you get the interview, they're normally going to do a technical interview.

00;49;59;00 - 00;50;20;24
John
So you'll need to know at least enough to pass that technical interview. For a long time, companies were very focused on this thing. It's called Leet code, LEET code, and it's short for elite code. But the idea of Leet code is that these are like very involved and very complicated riddles. I Guess is the best way to say it.

00;50;20;24 - 00;50;41;12
John
Like, what people don't like about these Leet code questions is that you kind of need to know the trick to how they work in order to solve them properly. And for a long time, companies like Google pushed these Leet code style interviews in the industry for a long time, was doing these types of interviews. But that's changed quite a bit recently, and a lot of companies will say appropriately, we don't do these Leet code interviews.

00;50;41;22 - 00;51;04;09
John
So those are the companies you really want to find, because the same companies that are not looking, I guess, Leet code in general, selects a very particular type of engineer, particularly one that went to college for computer science, you know, and knows about computer science fundamentals. And so if you're not that person, you want to find a place that is not only picking people like that.

00;51;04;09 - 00;51;04;15
Kristen
Right.

00;51;04;15 - 00;51;15;02
John
So you find a place that is not having these Leet code interviews and then just do as many interviews as you possibly can. They will be hard. Sometimes they won't go, well, you know, that's how it is.

00;51;15;09 - 00;51;16;07
Kristen
Yeah, it happens.

00;51;16;10 - 00;51;16;17
John
Yeah.

00;51;17;10 - 00;51;23;19
Kristen
So just keep applying and just because one place said no doesn't mean that all places will say no, right?

00;51;23;21 - 00;51;41;02
John
Yeah. And if someone gives you an interview that you think isn't fair for your position, then that's probably not somewhere you want to work anyway, because that means that they don't have the ability to support someone at your level. So I think a lot of times people are looking for their first job and they actually want to like be at a higher level kind of from the start.

00;51;41;02 - 00;51;59;23
John
Maybe they don't want to take an entry level position. They want to have the next position up. If you want the support, I would actually say it's better to take the entry level position and then try to move up as fast as you can than to overreach and feel like you're lost. Because once you're coming in as maybe a senior software engineer, no one's there to help you.

00;52;00;05 - 00;52;15;09
John
You know, people are helping you and giving you resources, but you don't have a mentor from day one. You don't have a group of people you can, you know, just instantly ask for basic questions and things. So if you don't have a good foundation, you know how to program, find a company that can actually support you.

00;52;15;17 - 00;52;28;09

Kristen
Yeah. You talked a little about like fit and being at a place that matches what you want in your work environment or your work style. What things did you learn that are differences among companies?

00;52;28;09 - 00;52;45;12
John
A big one that we talked about is work life balance, which is, that's in every industry probably. But work life balance is when you work at, for example, a start up, a lot of the startups are having employees work 60 hours a week. So that'd be an example of bad work life balance because a typical work week is 40 hours.

00;52;46;01 - 00;53;02;00
John
Yes. And on the opposite side of the spectrum, a really good work life balance is somewhere, and I think GitHub pretty much has this, where like my kids are having a dance recital tonight and like I need to leave at 4 p.m. to make that. And that's totally fine. I don't even need I don't need to ask for time off.

00;53;02;00 - 00;53;30;15
John
I don't need to figure out when, I just do it. Or like, hey, I woke up late this morning, I'm not going to be in until 10:30. That's fine. I don't have to tell anyone. I don't have to do anything. And no one does, nowhere, anywhere in the company. That's a good example of a good work life balance where like the job is kind of molding itself around your life to the point where it's not an impediment on the things that you have to do in the hope that, that good work life balance will essentially make you a more productive member of the team.

00;53;30;18 - 00;53;30;28
Kristen
Yeah.

00;53;31;20 - 00;53;38;22
Kristen
As then you can be fully dedicated to the place where you are when you're there because you were able to take care of your life life.

00;53;38;25 - 00;54;04;16
John
So work life balance is a really big one. Another big one is like whether or not to have support and time spent for more junior developers. I guess another one is some companies are big enough that they have engineering, onboarding. They'll actually have in some cases like a four week engineering onboarding where they're walking you through, here's how our systems work and here's how to use this thing and here's how to develop this piece of code and here's how everything's laid out.

00;54;04;16 - 00;54;15;17

John
So companies like that, that's a really good sign that they're kind of like invested in making sure that everyone's on the same page, but typically only like mid to large size companies can, or do, take the time to do that.

00;54;15;20 - 00;54;23;28
Kristen
We're kind of at the end here. Is there anything else that you wanted to mention to people that might be interested in becoming a software engineer?

00;54;23;28 - 00;54;27;27
John
I mean, hopefully I've gotten across the idea that this is the best thing you can ever do.

00;54;28;06 - 00;54;29;08
Kristen
Yes, obviously.

00;54;29;20 - 00;54;50;15
John
And software development is super exciting and always changing. And software is in every industry. So no matter what your passion is, if your passion is clean energy or your passion is fintech or banking, or your passion is, I don't know, fishing or something, like, software you can do all of those things. You can find a place that, that does it.

00;54;51;01 - 00;55;10;13
John
And on top of that, the fact that a lot of software companies let you work remote and have things like flexible hours or good benefits, software pays really well. There's a lot of room for expansion in your career, like hopefully, I have communicated. I think it's like the greatest thing and I'm so happy that I found it.

00;55;10;13 - 00;55;11;03
Kristen
Yeah.

00;55;11;03 - 00;55;12;21
John
Yeah, I hope you'll enjoy it too.

00;55;12;21 - 00;55;18;15
Kristen
We're really glad that you took the time to talk to us a little bit about your career. Thank you so much for being here.

00;55;18;23 - 00;55;21;29
John
Thank you. And that's it.