And the worst thing is having to go back to a project manager or a scrum master and say, hey, I need to add a new user story for this new document that I need to write. I mean, you're almost like guaranteeing that you're gonna be micro managed if you do things like that. Why does it even matter if we write good code like a senior programmer or not? Well, first of all, it lets other people, understand our code. You've probably found yourself in a code base before reading it, and just being like, I don't understand what the hell going on in here and the temptation is often to just jump in and rewrite everything. But being able to write code where people can really easily read it is something that other developers on team will really appreciate and that often is the difference between them looking at you as a junior developer and a senior programmer. The second reason to even matters to write good code. Is it reduces the time it takes for you to support it and explain it to somebody else. I often work with other developed developers who will rush through and not follow some of the six things. I'm about to tell you towards the end of the video. And then their when they're constantly interrupted by other developers asking them questions about the code, and they're like, I don't have time for it. Well, if you do, freight follow some of the things I'm about to share with you today, that's actually gonna reduce that time quite a bit and let you just work on what you actually wanna do. And the third reason, if it's really good to learn to write code as a truly senior programmer, is that it reduces the chance that somebody's gonna rewrite your code. It's think one of the most frustrating things as a developers is when you work really hard to implement a feature. And at a very short period after that, somebody comes along and they don't like... Some aspect of it, or they don't understand it and they rewrite all of it. So if you file some of the things I'm gonna share with you today, I think it's really gonna increase the shelf life of your code. And that's actually gonna be much more valuable to your company because I think one of the biggest ways of time on many software projects is rewriting code that already works. It's just to suit someone's preferences when there's not really a ref reason that's actually adding value to the project. So what are some habits you can put in place to write code like a truly senior programmer. Well, I think the first habit you can put in place is completing your work before moving on. There's immense pressure on many projects through ashley, if you're doing scrum or something like that to say that you're done with work and know that, okay. I'm eighty five percent done, but there's a little bit of the code that's left and I'll just leave that to to a future time. I'll come back and fix it. But at least in my experience after many projects have I've been on. When I do that, I'm basically building up, tech to debt for myself. Now nobody else may see it. I may be the only person on the team that's aware of it. But it's ultimately there and somebody else will become aware of it. And a lot of times When I'm reporting status to project managers about what I'm working on, I have to be really honest if I'm gonna be truly profess and say no, I'm not done with this task. And if the rest of the team or the management is frustrated, I'd much rather have them be frustrated and have them know the real state of the project then basically lie to them to just say I'm done with something. So having the self discipline to really look at your code like this is my personal brand, Scott and n rod talked about that in the first interview I did with. I thought that was such a great point that every time we write code, somebody else is gonna look at it down the road and basic, they're gonna measure what they think of us based on that code. So having some pride and making sure that you really finish it before you move on is honestly one

of the first things you can do If you really wanna be taken seriously as a senior programmer. Where The second thing I see. Truly senior programmers do is the enforce coding standards. Luckily today, and we didn't have this fifteen plus years ago, you know, if you're using Vs code or something like that, you know, Ema max and Vi and some other tools will do this too. Most languages have auto formatting. So you go in and you write your code. If you decide to put your line breaks and your curly braces in different spots. Once you save the file, it'll format everything for you and it can assistant way. But, if you don't have something like that, finding a third party tool, if you're using Visual studio or, like, a thicker Ide that has a lot more features something like that there are plug for it, that will let you do code formatting and establish code formatting for your team, But some of the most frustrating code bases I've worked on have been where everybody has a completely different standard for formatting their code. And it's so easy especially as a consultant, sometimes when I come in on an existing project that I'm asked to audit to tell that very different developers broke the code, and it's really frustrating to have to read different styles of code when honest lee, we've got the technology today to do it. Now, I still think it makes sense to write a wiki topic or have a markdown file or something somewhere that's really easy for the team to get to that's says, these are our coding standards. And if you just use an off the shelf standard like that's been established by an open source community or something like that That's even easier because you can just say we're using this standard and give them a link to that standard to read about it. Another thing you can do that it will really help you write that truly senior level code, I think is to get much more disciplined about documenting patterns. I see many teams who will start a new software project they'll discuss amongst themselves pick a set of patterns, Some of them are just open source very well known patterns and some of them are not. But especially like on a react project or something like that where you're using react just for the rendering part of the page and there's all these other libraries and different patterns you can pick from, having least one wiki topic or markdown file or something that says, here's all the patterns that we've agreed we're gonna use on the project. I think is really valuable. Because that way, too, if you're doing code reviews or something like that. And you find the developer on your team is using some other pattern, It's not just arbitrary where you're like, well, we picked this pattern, we didn't tell you about it. You can be like, hey. Remember when you joined the project, We showed you this page with all the patterns. This is one we haven't agreed on. Now if you're using an off the shelf pattern something that's already part of a framework or a language that you're using, I think a really good idea can be just to include a link. Now to the documentation somewhere online that at least has like a tutorial or explains how to use that pattern just to make it easier on your teams that they're not googling to try to find some example that might not be the example that you want them to look at for a pattern that already exists. So Now if you have a pattern that's new, you've created some pattern to be dry or basically to make the code easier to write, I often see piece people that will introduce a pattern, they'll do a brown bag lunch or a demo or, like, a a meeting over Slack and they'll show it to the development team. And then that's the last time they talk about And I think every single new pattern that you introduced for which there is an existing documentation. You gotta create a wiki topic. Or a markdown article or something that says, okay, here's the purpose of this pattern. Here's when you would use it. Here's when you would not use it And here's

some example snippets of code that will show you how to actually apply it in some common scenarios. As a consultant, I'll tell you one of the things my clients really like that I do for them is I often any code that I write in any project On am really anal to be honest about my documentation. But I just find the great thing about that is the the other developers on my team, especially when I come in and remember, I'm a consultant, and I'm not part of their company. They love working with me because they have a all these topics to go to, I'm really disciplined about that. So if you really wanna get looked at as a senior programmer. Unfortunately, a lot of these things that we've convinced ourselves aren't really important, like self documenting code, you really just need to reject some of that and actually have just the maturity and the professionalism to really just step back and take the time and actually document the patterns and the decisions that you've made on your project. Another thing that will really help you work with other developers like a truly senior programmer would write code is to review any new pattern you're introducing to the project as soon as you've introduced it. I've been on projects before where let's say there's one developer that's come up with a new way they wanna access the database. That's different than how everybody's been doing it. It's a completely different library. And they just introduce it to the code in over several Sprints or just months of development. They just start to slowly roll it out and introduce it to every feature across the code base, and they never told anybody about it. And then they find out later that somebody already evaluated that pattern and they had really good reasons for what the either they didn't wanna use it. And now it makes the person look kinda like an idiot because they realized that it's not actually gonna meet the requirements of something that maybe another person on the team, either already knows is coming later in the code base. So I would just suggest if you're going to do something like that, change a pattern or introduce a new pattern, put just enough code into the project to demonstrate it, and then get with your team, show it to them, ask for feedback. They're probably gonna have some opinions on ways you could do it even better, and then you can get buy in from everyone and they'll actually be supportive of it, and you can actually enlist all the other developers on your team, to help you incrementally add that pattern to the code instead of putting it all on yourself. Another thing you can do that will really help you write code like truly senior programmers that I've worked with this. Never create a user story or a ticket or if you're using Jira something like that, a task, that represents ref. Why would I say this? Well, anything that you put in is an individual item that your management can see is something they can take out. And I just find that building quality into code is something that we as developers, I think we happen to take responsibility for it. And as professionals, we have to actually prevent our management from making bad decisions that are gonna tank their project, whether it's through technical debt or just not delivering stuff that actually can be extended as the project evolves. So when I'm on a project, and I realized factoring needs to be done, I never create a ticket or a user story or estimate a ref effort. What I do instead is, I discuss it with the rest of the developers and we come up with a plan on how to spread that effort across the team so we can incrementally ref ref. Now incremental ref ref... Factoring is a really valuable skill and I meet a lot of developers who don't know how to do this. They look at it like, okay if we're gonna change, let's say the database pattern we're using or we're gonna change some way that we do state management react. We're gonna use redux or something like that. We

have to now take a two week period or a month, and ref all the code to do that. It's a lot smarter to just decide on the development team. Okay. Every time we get a feature, we're gonna basically add some tough to it so that we build that new feature using the new pattern, and we also go back to one or two other features and update it to use this new pattern that we've agreed upon. The key though is don't call that out as a separate item because again, management will be tempted to pull that out. And I think as professionals we can't let that happen. And the final thing that I think will really help you write code, like truly senior programmers, and I love when people do this on my teams is anytime you're asked to estimate work, you gotta add extra time for unexpected design meetings and documentation that you might have to write. It's so common when I'm on projects. I'm going along writing code, starting to implement a feature, and I find out, oh, there's some uncertainty in here. And to meet the requirements, now, I've gotta add some new documentation. And the worst thing is having to go back to a project manager or a scrum master and say, hey, I need to add a new user story for this new document that I need to write. I mean, you're almost like guaranteeing that you're gonna be micro managed if you do things like. That. So I always tell people, give yourself extra time, not just for meetings and uncertainty and all that, but writing documentation. If you really wanna keep the quality of the code high, and you wanna keep the team as productive as possible, you have to protect your commitments with the extra time that you know it takes to write really excellent professional code. Now I know a lot of these are my opinions today. There's probably things that you agree with. You don't agree with. Leave me some comment. Below. I'd be really interested to note if you don't agree with me why and what would you suggest instead. And if you've also got some other suggestions for what makes someone a truly senior programmer, let me know about it. Until next time, thanks.