EG New Developers Working Group (2023-05-17 15:01 GMT-4) - Transcript

Attendees

Christine Morgan, Daniel Guarracino, Gina Monti, Jane Sandberg, Jane Sandberg's Presentation, Jessica Woolford, Josh Stompro, Marie Russell, Martha Driscoll, Michele Morgan, Mike Rylander, Scott Angel, Stephanie Leary, Susan Morrison, Terran McCanna, Tiffany Little

Transcript

This editable transcript was computer generated and might contain errors. People can also change the text after it was created.

Terran McCanna: Good afternoon everyone or morning to those of you on the West Coast. Well, I guess it's afternoon technically to you too. Just barely by one minute. So, welcome to the data developers meeting. I think everybody here has been here before, so I will just go ahead and launch into Jane's Tuck and she has consented to Join us to talk about test-driven development,…

Jane Sandberg: Here, great.

Terran McCanna: so I'll turn it over to you, Jane.

Jane Sandberg: Thanks Terran. It's so good to see everybody. Good to see folks. I know good to see folks that haven't met yet. I'm going to see if I can figure out the screen sharing.

Terran McCanna: To be a box on the bottom with an up arrow. Yeah, you got it.

Jane Sandberg: and I'm going to try slides even

Terran McCanna: Getting all fancy on us.

Jane Sandberg: Are the slides coming through? Oh great.

Terran McCanna: Yes, they're good.

Jane Sandberg: So yeah, thanks for your interest in this topic. I'm coming to you as, like somebody who's been trying this for the past year or so, not an expert but just something I've been playing with finding pretty helpful and so I wanted to share What I've learned so far. Today, I wanted to give a brief overview of test-driven development. Want to spend most of our time working together to take a small feature in the angular client and just like open vs code and try to work together on it. In test driven development, kind of way. And then and with an invitation to increase the test coverage in evergreen and ways you can help out with that. And interrupt me at any point during this session, please.

Jane Sandberg: So test driven development is all about unit tests. So I want to start by seeing the praises of unit tests. And so when I'm talking about unit unit test I'm talking about a test that's right. An automated test that tries to focus in on a very small amount of code and just confirm that. It's working as expected. And they're great for a number of reasons. They give you really fast feedback You don't have to like log into a browser. Make sure the right patch is installed. Maybe log out, log back in again click

through to a few, a few different screens. So like manual QA. I love it, but it is time consuming. And to end tests like Nightwatch JS or like the Pearl live tests, where you're testing a bunch of different system components, and making sure that they work well together. Those.

Jane Sandberg: Aren't as don't take as much time as manual, QA but it's still like enough time for you to like, pet your cat, make some tea, come back to your screen and then, like, forget what you were doing. Unit tests, they'll give you an answer in like milliseconds. They help you catch bugs. They help you. From introducing new bugs, generally give you kind of a safety net. They give you the confidence to know that you can make some interesting creative, dramatic changes, experimental changes. And as long as that test suite is still passing. You haven't really changed the behavior of the code and, and it's still good.

Jane Sandberg: An interesting thing about tests. Automated tests as well, is that they kind of have this role halfway in between documentation and code. Because they are describing how your code is used. Like what kind of variables you can throw into it? What they expected, return values are and Unlike documentation, I'm like, technical documentation, you can just like, go ahead and run it and it will tell you if the documentation the tests are accurate or not. And again, I love documentation. I love unit tests. I think they work well together. I would never say, we only need tests. Are we only need documentation? We need both. Also. If you are looking at it as chunk of code fit somebody else wrote and you don't understand it at all one technique that you can use to understand. It better is to just like write a test to see what it does.

Jane Sandberg: So all of that helps maintain the software long-term and also as a reviewer I find it helpful to if somebody includes a test with a patch, you can just like run the test and not have to like figure out all these logic games in your head of like wait what happens if I entered this value into that? Any questions about unit tests?

00:05:00

Jane Sandberg: Zoom to my next screen. So, When I first heard about test driven development, I was thinking like Okay so it's just you write the tests first. Just so that you make sure that you do them, keep yourself honest.

Jane Sandberg: that's kind of how I thought of test driven development and that's true but over the past year or so, as I've been watching my colleagues at Princeton do a lot of test driven development I've become like really impressed with two other ways test driven development can help. so, first of all, if you write the test first, that kind of, by necessity means that the code that you're writing is easier to test than if you weren't if you didn't have testing in mind. And so that seems like, kind of obvious, but what is nice about that is that that's easy to test. Also, tends to be a little easier to read as well. Because if it's easy to test, that means you're like not relying on a bunch of external dependencies.

Jane Sandberg: You're maybe sticking to single responsibility principle and only trying to do one thing with the chunk of code, you're keeping your code in smaller chunks. Both of those help both testability and readability, so that's a nice benefit. But for me like the top benefit of using test driven development has been, I get like so distracted. I get like I if I'm like looking at a computer screen for like an hour and a half, I'm going to like go down a rabbit hole, which is why I've put a picture of a rabbit staring out at us from a little hole in this slide. All like, spend 20 minutes, like trying to investigate something that ends up being totally irrelevant. And then like coming back to my code and not even remembering what I was trying to do.

Jane Sandberg: And I've well pair programming with my colleagues who use test driven development. I've been just like really inspired by how they always like know. Like What is the next step? What is the next step in the process? And so that's why I wanted to type test driven development myself.

Jane Sandberg: so, Here's are the steps that I take when I'm doing, I'm trying to approach a problem through test driven development. First of all, I run the test suite, make sure that everything passes if any thing, if any of those tests aren't passing,

Jane Sandberg: if they're red. So oftentimes when you run the tests and they don't pass, the console will give you a message in red, so I'll use that language interchangeably. If you're test, suite is red, it's not passing. If your test suite, it's green. It is passing. But anyway, if you my first step is usually run the test suite, make sure it's all green. If there are any errors, address those first. Because we want to make sure there are no holes in our safety. Net. Step two is just like trying to break the the problem down into small, easily testable requirements. Since we're trying to test units trying to test individual little chunks,

Jane Sandberg: if we want to define what those chunks might be, And then I choose the easiest requirement and write a failing test for that requirement. So basically write a test like, imagining a world in, which you'd already written the code, right? Right. The test that asserts that that already has happened. It will fail. So the step is called red because the test suite will be red. Then as soon as you have a test that's failing, your next job is always try to get that test passing, because any test that's failing is a hole in your safety. Net. Even if we just added the whole ourselves, it's still, we need to patch that that hole in the safety net. So the next step is to write enough code, just enough code to make your unit test. Pass, this does not have to be good code, just has to be just enough to get your test to pass and this step is called green.

Jane Sandberg: And then the next step is to improve the code. So instead in the green step we were writing just enough code to make the unit test pass, it might have been not our best work, it might have been goofy little code. We really want to once. Once the tests are passing, once our safety, net is patched we want to improve the code, make it more readable more maintainable, all that stuff. And so that step is called refactor. So anytime you read about test-driven development online, you'll see this cycle called Red Green refactor. The idea is you just keep going down your list of requirements and each requirement. You just write that failing test read. You write the code to make that test pass green. And then you improve that code, so it's up to how you want it to be and that's the refactor.

00:10:00

Jane Sandberg: so, That was the part where we get to actually do these steps ourselves. So for the angular unit tests, here are the steps to do that. Let me find my terminal really quick.

Jane Sandberg: Oh great. My slides now.

Jane Sandberg: So, Step One, get an evergreen development box. Step two is install Firefox and chromium. Since these are javascripts, they're they're running through the browser. Then go into the eg2 directory. Install the node modules, and then the first time you want to run this test suite, you run NPM, run test. All the future times you can just run ngtest which is a little bit faster. To make terminal.

Terran McCanna: Jane, would it be possible for you to make the

Jane Sandberg: Yeah, sorry I didn't hear Terran.

Jane Sandberg: Is how is this size?

Jane Sandberg: Yeah, if it is not working out for anyone, let me know.

Jane Sandberg: so, in the EDT directory, The NPM install. There aren't any additional dependencies. But yep, here we go. And then it's just MG test.

Jane Sandberg: No angular will like compile our code, which, of course, always takes a while, but then after it does that, it will go through the test suite really fast.

Jane Sandberg: All right, so it ran our 92 tests and after everything was compiled it took two and a half seconds.

Jane Sandberg: So our first step is good, the Tests suite is all green, of course. All this text is green so you can't really see that. But It would definitely tell us if it was red.

Jane Sandberg: Let me go back to the slides. I also want since that, that compile step took a long time. But I often do is I cheat and don't run this in an in an evergreen development box. I just run this locally and most of the tests are such that they'll just go ahead and pass even if Evergreen's not running locally which is nicer because I have a little bit more ram, it goes a little faster.

Jane Sandberg: So Step Two was to divide divide our feature into small requirements and so I've done taking the liberty of doing that today. So, the bug is called Bug 1754. 364 constraint inputs for lib time zone settings.

Jane Sandberg: And so this is for the library settings editor. The idea is instead of just like a blank input box for administrators to type in the name of a time zone actually give them a list of valid time zones so that they they know what what What values are actual valid time zones? We noticed we have a comment for Mike telling us exactly how to get that list of time zones, which is nice. Remington's done some documentation work. I figured out another way to get a list of time zones, but it doesn't actually work so we'll definitely use Mike's way.

00:15:00

Jane Sandberg: and so now we get to choose which of these smaller requirements, we think is going to be the easiest I'll say this. Last one is not the easiest, but other than that, Any places people want to start, which of these three you want to start with.

Terran McCanna: I would say just retrieving the list in the first place.

Jane Sandberg: Sounds good. So, I'll head over to VS code and we can start writing that test.

Jane Sandberg: So the first question is, Where do we want this? Do we want like a separate time zone service? Do we want to add it to an existing service? Do we want to put it in the component? We're going to use it.

Terran McCanna: Since we all since Pines only works in one time zone, I'm not sure where all of the different places. Are that evergreen would select time zone.

Jane Sandberg: It's a good point. I also don't know. I don't even work in an evergreen library anymore so I definitely no.

Terran McCanna: so, maybe put it in, in the kind of generic set of like, utilities That can be used from a number of places.

Jane Sandberg: I like that. Let's do that. Yeah, you tell, here we go. This seems great.

Jane Sandberg: We've already got one for date. So it seems like this might be a really good place to put it.

Jane Sandberg: So to write our test. Oh first, let me check. How is the font size here?

Jane Sandberg: Cool. So, let's just Call it say Time zone. That's that. Yes. All the test files. All have this dot spec in them. So you know, that it's a test.

Jane Sandberg: And so the format for tests in angular, is it uses this library called jasmine? So, You define you use this keyword describe. To describe your test suite. and so let's just call this Time Zone Service, maybe Because we think we'll write it and service in here or times when utility, I'd like that Terran.

Terran McCanna: So is the jasmine library already included? Or is that something we are going to need to install before we do this? Okay.

Jane Sandberg: That is already included, that's just comes with with angular.

Jane Sandberg: And then we write individual test cases. So,

Jane Sandberg: You can say it includes valid. Zones.

Jane Sandberg: What's to say? Have like maybe a class called Time zone. Let's keep it simple.

Jane Sandberg: And maybe a method called values. and then, we can say, To. Contain, I always struggle with, if it's to include or to contain but it's to contain. And then what's the name of a time zone? That we would expect it to have?

Gina Monti: Can we go to Central?

Jane Sandberg: Sure. Uses evergreens. Expecting.

Jane Sandberg: With them in this format.

Jane Sandberg: Oh, okay. I have no idea where the chat is. Oh yeah. America Chicago.

00:20:00

Jane Sandberg: Say America Chicago. I also saw America slash New York in there.

Jane Sandberg: and so now we've got a test, which when we run it

Jane Sandberg: Well, let me do one more thing in this test. One thing we can do to make make this a little bit faster. Is not run, all 92 tests but we only want to earn the test that we're working on right now. So you can focus on a test and to do that you Do an f in front of that. So for focus. So focus describe time zone utility and it will just run this test. I already see red squiggly lines. I know it's going to fail so so we're on the right track here.

Jane Sandberg: So I'll do MG test. I'm on my local machine, which doesn't have angular installed globally. So I'm going to add npx in front. I'm just to get it rolling. Okay. And it says I can't. I've never heard of time zone. I don't know what this is. So we've got a failing test. See a lot of red, we're on the right track.

Jane Sandberg: So now is the step where we actually have to turn this green. So first of all, we have to add something called time zone so that it knows what to do.

Jane Sandberg: So we can just export class time zone.

Jane Sandberg: No. Do I know how to do JavaScript? We'll find out, please, at any point, if you see me about the big, like some horrible mistakes, please, just dump it.

Jane Sandberg: Okay. Okay, so it's still mad. It still can't find time zone. That is because we haven't imported it from that other file yet.

Jane Sandberg: Wonder if it gets code, will be smart enough to find it.

Jane Sandberg: Okay.

Jane Sandberg: So, our previous failure was cannot find time zone. Our new failure is property values does not exist on timezone. So we're making progress searching, fresh new errors.

Jane Sandberg: So I would just find a function. I'll use.

Jane Sandberg: Let's return an empty array for now.

Jane Sandberg: Is it just without that?

Jane Sandberg: Okay, we get a new and exciting error. Expected. This blank array to contain America Chicago. It didn't contain America Chicago. SO We're. Which makes sense because we returned an empty array.

Jane Sandberg: So now we're looking for the quickest way. Easiest way to get this test passing copying and pasting is always a quick way to get a test passing. So, let's just grab Mike's code.

Jane Sandberg: You can import moments.

Jane Sandberg: I'm always so bad with the export the important exports and always hope that VS code will do it for me. oh, he did.

Jane Sandberg: Yay, we've got one success here and it's in green. We have done. The red, we've done the green and now we have to do the refactor. So, is there anything that we would like to change about this code?

Jane Sandberg: Word this code.

00:25:00

Jane Sandberg: I will get rid of this extra line. and I thought one, maybe improvement on this test would be just to also add a test for Making sure that it doesn't include invalid time zones. Just to be sure.

Jane Sandberg: so, Can anyone think about like a fake time zone that we would not want to be in here?

Terran McCanna: Margaritas margarita time.

Jane Sandberg: What? Oh well, that is red because the test was wrong. Because we need, we want it not to contain. Good time. I've been like sucked it in Margarita the time.

Jane Sandberg: Okay, two success. We've got two tests. This is great.

Jane Sandberg: Anything else that we need to do with this particular functionality?

Jane Sandberg: Any questions so far?

Terran McCanna: Mike had a comment in Chat that says, If we want the list to be magic and presented in the library settings, editor, the list would need to be wrapped in an ideal class that we could tell the Yao US about So I'm does that mean the list that is it's being actually being pulled from in the first place.

Jane Sandberg: Let's see. Mike could you just like if you're able to could you unmute and say more

Mike Rylander: I am, I've pardon my voice. I'm getting scratchy coming down something but So I I'm my assumption was that. but the assumption behind that comment was that we wanted this to be in the, In the Library Settings editor when you go and choose when you, when you're going to set the Yes for the local time zone but I can't recall. There may be any.

Jane Sandberg: Yeah, I was thinking of building one as part of this work. So certainly here is there's this input type Over here.

Mike Rylander: Mm-hmm.

Jane Sandberg: Which just returns like what data type that the library setting is. And then that's used in this big old ng, if Or ng switch case where if it's an integer you show this display. It's a currency you show this display. I was thinking we could just add like another one for time zone and then Modify this function. So that it recognizes some that there are some special cases where We want to present a selector of form input. That's not. Exactly tied to the data type. That was how I was hoping to approach it today.

Mike Rylander: I well let's let's see if let's see how far we get Yes, let's go for it and see if we can do it that way.

Jane Sandberg: So let's since we're already talking about it, let's jump into this third one. This input type function, needs to return time zone. If the setting name is Lib Dot time zone.

Jane Sandberg: so,

Jane Sandberg: let's make our test.

Jane Sandberg: The component, it's going to be a little bit more complicated because you can see instead of our like unlike our Time Zone Object or Time Zone class which didn't have any dependencies, we do have a dependency here in the constructor that will have to keep in mind. And also, we've already got some behavior existing and we want. Maybe don't want to mess up that. So I think we're all start actually is write a test for input type just to make sure that we've got that behavior locked in before we start messing with that method.

00:30:00

Jane Sandberg: So I'll do f describe.

Jane Sandberg: A long name.

Jane Sandberg: I'm just going to copy and paste.

Jane Sandberg: And specifically, we're going to be you can nest these describes as much as you want. So, We're specifically going to be describing that function, called input type.

Terran McCanna: So I noticed that you didn't use up describe on the second one. Is that?

Jane Sandberg: Yeah, and if it's wrapped within an F, describe it will like focus on the whole whole area.

Terran McCanna: Okay.

Jane Sandberg: and to set up this test, we'll need one of these.

Jane Sandberg: So we can do component equal new one of those.

Jane Sandberg: Import it. And then it's mad because we didn't give it the model that it wanted.

Jane Sandberg: so, we can try the simple thing and see if we can just do new NDB, modal here, and make it happy.

Jane Sandberg: But then it's getting to the point where like, okay but this this constructor for ngb modal wants all these arguments. I Don't want to like get into the internals of energy being modal. They might change their API in a future time and I don't want to be locked to their implementation. so instead, I'm going to get rid of that and what we can do is make a fake one so we'll do is

Jane Sandberg: Asked. Mock the modal. Pulls Jasmine. This is another thing that Jasmine provides which is really nice. Which is this ability to create these fake ones for fake dependencies. That will keep typescript happy. We'll keep your test happy. And you don't have to worry about what the NGB modal team is going to be doing. It's, you want to figure an NDB model?

Jane Sandberg: It takes this array of what functions you want to have available in your test. I, Don't really know what we'll want in our test so I I'm just going to go with open because I know that that's a function on Ngb modal.

Jane Sandberg: and then,

Jane Sandberg: Through our mock in there. and then, Happy so far.

Jane Sandberg: And then we just want to say expect components. And put types.

Jane Sandberg: To equal. Then. Oh, because it's right here.

Jane Sandberg: Run our tests.

Jane Sandberg: It's mad.

Jane Sandberg: Okay, it's this is my classic error that I always do. Expect was used when there was no current spec, which means I forgot to like, actually write. And it block which says, what the best actually is doing. so, how about this test could be saying? It gets its value from. The.

Jane Sandberg: What was that called? It's called the data type. The Entries data type.

00:35:00

Jane Sandberg: that again, now that we have the right syntax,

Jane Sandberg: okay. And We are getting this error expected, undefined equal. When you get like these really nasty errors in angular. So especially in tests sometimes, if you scroll up there's additional information. But I'm not finding anything here.

Jane Sandberg: so, And also these line numbers don't really mean anything because they're looking at the compiled version of the code. Which is a huge annoyance that I have with angular. But I do happen to know the issue here which is that we never told it what the entry was. So we can give it a fake entry and make it happy that way.

Jane Sandberg: And that is looking in the data type. So we can just make sure our fake entry has a data type of wool.

Jane Sandberg: Okay, and we have success here. So now we have the safety net knowing that we can work with this data type function or this input type function. And we can make our changes and we know we'll know when we're changing the behavior.

Jane Sandberg: So we can. To work on what we were actually wanting to do, which I kind of forget. So I'll go back to the list.

Jane Sandberg: So we want this function to return time zone. If the settings name is Lib Time Zone,

Jane Sandberg: So then we can set up our kind of test data differently here. So, data type would be a string. The name would be Red Dot time zone. And then we expect it to return time zone.

Jane Sandberg: We're in the red part of red green refactor. So, we'll expect this to be red because we haven't yet implemented that and we get the current behavior which is just to say string which is just to take the input type just from this data type I'm going to get input type and data type confuse so much while we're talking about this.

Jane Sandberg: But this should be pretty easy to to implement and say, if this stuff entry name. Is. Lived time zone. That is to say if, if the library setting that we're trying to edit in the editor is named Lib.com Zone. Then we want the special input type. Called time zone.

Jane Sandberg: I think that should be enough to get it to Green.

Jane Sandberg: And it is. So now we've done red, we've done green now, we have to do Refactor. Are there any changes that we'd like to see in the test or in this code?

Jane Sandberg: I see one thing that I'd like to change. It'll give it another minute for other folks to say,

00:40:00

Jane Sandberg: so the thing that I'd like to change is this line seven and eight and lines, 15 and 16 are just like duplicates of each other. And so just to keep this a little shorter, keep it more concise. I'm just going to move this out of the test and put it at the top.

Jane Sandberg: My test again. See how it's looking.

Jane Sandberg: And then, we can also. For this. Where we're setting up our component. We can also do a jasmine has this before each block that you can use.

Jane Sandberg: And that's code that runs before. Each of your tests within this describe,

Jane Sandberg: so, Think.

Jane Sandberg: is that and then say Say in advance that this is going to be One of those kinds of components.

Jane Sandberg: Maybe I don't even need let anymore.

Jane Sandberg: Tonight, that's a little bit cleaner.

Jane Sandberg: So we have done two of the things on our checklist. I don't think we're gonna have time to do the other two. Especially Kind of the last one. This one. So I'll say that I did make a branch that does this. And these first three took me half an hour total to do and the next one took me two hours. So, but I can't hear interested.

Terran McCanna: I just want I just I just want to let you know, we don't have to we don't have a hard stop at four so however, you will long, you want to go is fine. If people need to step out then You know, it is being recorded. So

Jane Sandberg: Okay. Do a couple, I'll go through my slides and then I do have a have something that's at four Eastern so I'll have to jet at that point.

Jane Sandberg: so, Here is the part where I'm going to invite you to participate in, automated tests with evergreen. And so, my first invitation is that when you make a pull request, just include a test. Even if it's just for like a small part of your changes, if you're adding like 20 functions. But you only tested one still helps. Even if like some angry people on the Internet, think you're not doing testing, right? It's still helps even if you copy it and pasted it from another test in the project, it still helps. In fact that's like A Good. If there are enough automated test examples in evergreen that you can can get started that way, I think that's, that's just great.

Jane Sandberg: And of course, like we did, we wrote our tests before we wrote our code, in this case, but like all tests are great. Great additions to evergreen. So it doesn't matter if you wrote it before or after the code. um,

Terran McCanna: So, I have a very, very general question. So when you include tests,…

Jane Sandberg: yeah.

Terran McCanna: is the practice to keep that in evergreen and definitely or or Does that get removed at some point?

Jane Sandberg: And Terran. That's like amazing question. So yeah, they're just like any other code. They might outlive their usefulness and then we can just delete them. one example that I've been doing a lot in my day job lately is Writing Tests for Our We use the rails a lot. So they call them database migrations and evergreen terminology more like upgrade scripts. Because those are things where we want to make sure that we're not losing any data when we run those, those database changes. We want to make sure we're not like locking the database so that there's performance issues. But then like after we've run the database, migration like there's a really no point in keeping that test around because we've already done it, it's not giving us any additional information. So yeah, you can.

00:45:00

Jane Sandberg: Some test will like live the test test of time and others will be useful for a little while and then stop being useful and we can get rid of them. Does that answer the question?

Terran McCanna: Absolutely, thank you.

Jane Sandberg: and then, Even if you're not writing a test, you can also use that needs test label and launchpad to mark, pull requests that you think could benefit from some, some automated tests. And also if you're in a position to fund development, consider adding those automated tests to your requirements and also be willing to like get a little bit more money for them because it will take some more work. Realize that you're like, maybe not going to be able to stretch your development dollars as far. Initially, but that your development that you're finding will will stay reliable for longer.

Jane Sandberg: Um also please feel free anytime to just reach out to me on. If you have any questions I'm always happy to just like hop on Zoom for 15 minutes and look at something with you. want to say thank you to all my colleagues at Princeton who got me up and running with Test driven development and then also, If this was helpful for you and you are in a position to do some advocacy work. If this session was helpful for you the most meaningful, thank you. I can imagine is support for my local community colleges librarians and computer science faculty The administration at that college decided that the college doesn't need any librarians anymore. And also doesn't need a computer science program anymore and…

Terran McCanna: Oh no.

Jane Sandberg: they're all cutting half of their faculty of color as part of those cuts and it's a huge loss to our community and also it

Terran McCanna: That's insane.

Jane Sandberg: Yeah, it's ridiculous. And also I'll point out that like pretty much everyone in that library has contributed to evergreen in some way. A bunch of the computer science students have contributed to evergreen as well. So if you're in a position to help reach out to me and I can show you how to plug into those advocacy efforts,

Jane Sandberg: Oh, and yes, this is a library thing. I have to give you like books to read at the end. Just like first video, I like have it playing anytime I'm writing angular tests honestly, this guy just like read my mind somehow and knows exactly what errors. I always run into when I'm writing tests in angular. I, I swear like half of the video, the views of this YouTube video are probably me, it's great, highly recommended. if you have access to LinkedIn learning this test driven development course is in Java, but I thought it was really good with the concepts and and also at helping you like divide Large features into like small individual chunks.

Jane Sandberg: Working effectively with legacy code is like always just like a classic book it's great for this is not like a statement on like whether evergreens legacy code or more legacy code than any other project or anything like that. It's just like when you're looking at a complex code base which was written by lots of people. And you're trying to wrap your head around it and trying to make changes to it. Without breaking all the rest of it, this book is kind of like a choose your own adventure of like how to actually make those changes safely. And then, I just love everything by sending Metz. And she's a big proponent Protestant development as well. Any questions?

Terran McCanna: Feel free to unmute yourself and ask if you have any questions.

Terran McCanna: I think it was just so clear that nobody has any

Jane Sandberg: That's because we didn't do that last one, you think?

Jane Sandberg: Because Stephanie, I thought your comment of, like, figuring out which imports you need. And I literally always just use via's code anymore.

Stephanie Leary: I've got a question for you Jane. Um, are there any good tests that are already in evergreens code base that we should go look at?

Jane Sandberg: Yeah.

Jane Sandberg: The one that's coming to mind is a test that Josh you just wrote and it's not an angular one. It's a Pearl Life test but I thought that was just like a great example of Taking a small feature and then just like, exercising it with, like all sorts of like, pretty much any way you possibly could. And giving like a lot of confidence that the feature is working. As expected.

00:50:00

Josh Stompro: That took me forever to write because it was the first time I've ever done that, but it was, it was fun.

Josh Stompro: You find a link to it, I think.

Jane Sandberg: Yeah. Please feel free to drop that. Drop that link.

Terran McCanna: I'm going to go ahead and stop the recording, but I want to say thank you very much Jane. That was fascinating and I already know, I'll be watching it again as I learned more.

Jane Sandberg: I am glad.

Terran McCanna: And does anyone, we have officially another eight minutes in the session. Gina did you have something you said you wanted to talk about today? If we had time?

Gina Monti: I did, but I feel like it might be better for us to take a look at it after the translation stuff was occurring on.

Terran McCanna: Okay.

Gina Monti: Yeah, so we're looking at notices right now, but We're in vodacom.

Terran McCanna: Okay. Sounds good. Oh, Josh has the link in chat. If anybody didn't see it.

Jane Sandberg: If you want, we can take a look at what I did earlier. For those other two items. In this feature. Because I'll share my screen again.

Terran McCanna: Oh sure.

Terran McCanna: Actually. I should have kept the recording going.

Jane Sandberg: let me open the link first so I can

Terran McCanna: Josh, is it all right? If I add that link to the the wiki page, we're all post the video. As it as a pro example.

Josh Stompro: Sure. Yeah.

Josh Stompro: I mostly just copied somebody else. So,

Jane Sandberg: Okay.

Jane Sandberg: So this actually this first test looks really similar to what we did the first time.

Jane Sandberg: We chose like a different function name this time around and then file a time since, but other. And we also have margarita time in this one rather than I don't know, I don't know. Whatever that test is. Oh, does not include duplicate time zones.

Jane Sandberg: So then this was like the one that took me forever to write a test for which was the template itself. So angular gives you these tools for writing tests where you're actually like looking at the template and seeing like what ends up in the DOM

Jane Sandberg: Especially since this was in a modal. It took me quite some time. So I had to make this mock, modal component to like, give it a home and an NG container for the output to actually exist in

Jane Sandberg: this is pretty similar to our other test of the lib time zone. But then this one you have to use this thing called the angular testbed to actually like render the output and and check it that way. And that a certain at the end ended up just being we looked for the editor in this const edit element.

Jane Sandberg: This fixture detect changes. You just have to sprinkle? Yeah, around any time you change something to tell. Angular. Okay. I changed something in my test. Please just like go ahead and re-render everything and then just check to make sure that it was displaying this, this new component that I made the eg time zone, select

Jane Sandberg: so, that's the part that took me two hours and then This part.

Jane Sandberg: This eg this this was the the new component that I made that's just like a time zone selector. And the template ended up being very simple. It was just like literally just a label and a combo box. Test was pretty simple as well. I made like kind of another one of those those jasmine fake objects for the time zone service and just gave it one time zone in it so that I didn't have to worry about like hundreds of time zones and wanted to keep this simple and focused on the on the On the tumble box itself. And then the

00:55:00

Jane Sandberg: The typescript for that component. Was honestly, just like a lot of boilerplate angular stuff that's required. Not much actual. Actual code that. I wrote just like all that. Pages and pages and pages of boilerplate that angular makes you right.

Terran McCanna: Does anybody else have any questions?

Terran McCanna: Well, thanks again. Jane and next month which will be June 21 if Jessica if you're still up to it we'll talk about the whole whole list full holds, pull this filtering issues that you wanted to try tackling and I will do my best to try looking at that code in advance to familiarize myself with it. but,

Jessica Woolford: You need to.

Terran McCanna: Oh, so yeah.

Jessica Woolford: Make you test for it.

Terran McCanna: Yeah, Jane and Mike if you want to come back and help, that would be great.

Jane Sandberg: Sounds fun.

Terran McCanna: Okay, well, thanks everybody. I'll get the, I posted the two links to Jane's and Josh's code on the meeting page. And I will post the video as soon as it's available which will probably be tomorrow. I usually takes a while for it to process. So, thanks again. See y'all everybody.

Jessica Woolford: Bye.

Josh Stompro: Thank you.

Stephanie Leary: Did everyone?

Meeting ended after 00:57:30 👋