JOSH MOORE: Well good morning, everybody. How's everybody doing? Terrible, yeah.

JUSTIN BENJAMIN: I wouldn't know if--

JOSH MOORE: Who had the idea to move the party to last night? I don't know. But thank you for coming. Thank you, thank you, thank you. We're excited. Obviously, this guy's excited. I'm super pumped, super pumped. How many of you guys are architects? Wow, a lot. Engineers? Awesome. And contractors? Other--

JUSTIN BENJAMIN: Programmers?

JOSH MOORE: Cool, so we've got a few. Awesome. We have a few people. Wow, yes, we're getting fist pumps in the back.

JUSTIN BENJAMIN: Still wait. Let's wait 60 seconds, 60 seconds.

JOSH MOORE: You tell some jokes. No, don't please.

JUSTIN BENJAMIN: What do you call it? What does is this anybody? What is this?

AUDIENCE: [INAUDIBLE]

JUSTIN BENJAMIN: Solid point, but no. Suppose I don't know either. But here comes another one. Do you get it? I started with it. That was hilarious. I got more. What is this? What is this everybody? What is it? It's a flock of these.

[LAUGHTER]

JOSH MOORE: So y'all really didn't come to hear his jokes.

JUSTIN BENJAMIN: I got more. Where are we?

JOSH MOORE: Last one.

JUSTIN BENJAMIN: Shoot, now I got to build it up. What is this? Anybody? Nothing? It's this doing push ups in a mirror.

JOSH MOORE: So we're done with jokes. But we're hoping to have a little bit of. Fun how many of you guys have gotten into Dynamo-- hang on a second-- but are a little iffy on feeling your way around? So cool, so got a lot of people that--

JUSTIN BENJAMIN: Back this up. Who has never opened up Dynamo ever?

JOSH MOORE: I was getting there. And then you feel like you're pretty good at Dynamo. You can write scripts, all that good stuff? So we have had the enormous challenge-- at least I feel like-- to try to hit the whole gamut of more beginner things, more advanced things. I also want to mention that we have a handout that's pretty thorough. We had probably two hours of content. We had to cut about 45 minutes out. All the extra stuff is in the handout. So keep that in mind. If you've already downloaded the handout, hopefully you've seen some of the things that we'll be talking about today.

JUSTIN BENJAMIN: And just to be clear, the handout has screenshots of our notes and some dictation to those screenshots. So it's super helpful, at least in our minds, hopefully for you guys as well. So please, if you can't absorb everything, you can't follow everything necessarily that we're talking about today, the handouts really, hopefully, are going to be helpful for you. So we encourage you to download.

JOSH MOORE: Excellent. So I'm Josh Moore, Perkins+Will. And--

JUSTIN BENJAMIN: I'm Justin Benjamin.

JOSH MOORE: This guys needs no introduction.

JUSTIN BENJAMIN: I love all of you, by the way. I already love you.

JOSH MOORE: And you are in the class Everyday Dynamos, Automating Simple Solutions That Bridge Workflow Gaps Within Revit. So today what we're going to be talking about is how we can use

visual scripting in Dynamo to solve simple challenges. I find myself, literally almost every day in Dynamo, just scripting really simple things most of the time. Sometimes they're more complex.

We're going to give a brief overview of Dynamo, some of the basic things, terminologies that we feel like those of you who are starting your journey in Dynamo may need a refresher or may need to know. We're going to give you some specific examples that help bridge workflow gaps inside of Revit. And then we're going to give you some tips and tricks you can use when setting up Dynamo, when you're inside of Dynamo, some things that I've found, that we've found, to be super helpful to remember when you're in there. So you don't waste a whole lot of time like I did when I was learning.

That pretty much says the same thing we just talked about, just in the format that Autodesk wanted. So we're going to move that.

JUSTIN BENJAMIN: Let's all take a quick second to gander at this cartoon. I think it's pretty hilarious, pretty awesome. Go ahead and digest and give yourself a giggle. Give yourself a giggle. I want to hear it. That was enough. I loved it. So this is, in our minds, a great representation of what Dynamo is doing for you. Again Revit's super capable, super functioning. You can definitely get from A to B. But Dynamo's within reach. And just, it's a matter of getting out there, taking time and effort just to swap out those square tires with these round wheels. So it's right there. It wants to be your help.

JOSH MOORE: And so I think what we want to do today is help those of you who are struggling with all the new things you have to learn in the AEC industry. It just seems overwhelming sometimes. If you were in the presentations, the innovation keynote, the first keynote on Tuesday, sometimes you just feel absolutely overwhelmed with all the things you have to remember. Let alone, as contractors designers, engineers, architects, just the daily stuff you have to know just to do that part of the job excluding software.

So what we're trying to do is we're trying to set up. Justin's going to come from a perspective of a newer user. I'm going to try to come from a perspective of definitely not an expert. Maybe I would consider myself maybe three or a four in Dynamo.

JUSTIN BENJAMIN: He's an expert.

JOSH MOORE: When you compare yourself to all those folks out there that have been doing it for a long time,

writing code all that good stuff. So I am not a coder. I come from an architecture background. We both do. And so keep that in mind. We totally relate. And we're going to try to relate to you guys.

This next segment we're going to kind of get into is what we're calling Core dynamo. And like I said before, one of the things that we're going to try to do is relate some of these screenshots to you. In a few minutes, we're going to actually get into Dynamo, run a few scripts, and give you some examples of what you can do.

JUSTIN BENJAMIN: So again trying to be relatable, I"m going to compare everything to Revit. I think that, for the most part, we all have a foundation hopefully in Revit. If not Dynamo even farther of a reach, I think, for us all. So that said, Josh is going to use a lot of terms today. And for the most part, these terms are hopefully summed up in these seven bullets.

So relating to Revit, I'm going to relate each one of these terms to adding a parameter or parameters within Revit. So the first one, if you can think about when you click Add Parameter, you got type in the name, type versus instance, play that game. But one of the big things is you have to define as to whether or not it's text, a number, an integer, a yes/no, things of that nature. String is the equivalent of saying text in my mind. So when you say its a text parameter, that is what a string is doing. That's the kind of content or data within a string node.

A number is merely a parameter that has the ability to have decimals. So when you're creating or using a number node, you're giving yourself the ability to have a decimal. Integer, whole numbers, again relating to Revit and creating parameters, it's essentially you're going to do an integer or whole numbered parameter. A Boolean, it's a yes/no parameter. It's a yes/no. True/false, yes/no, same difference, so Boolean node.

The nulls, the way that I view nulls is just basically the lack of data. It's not saying it's dumb. It's not saying it's broken. It just doesn't know yet. So I always make a comparison to the yes/no parameter, where if you create a parameter assign it to content, and it's a yes/no, well sure by default, if you select geometry of that parameter type, it's grayed out. But it is checked. It's neither yes nor no. It doesn't know yet You got to tell it. So null node, a null piece of information is just lack of data. It wants to know. Won't you tell me?

Lists, in my mind, are schedules. You're just trying to sort data. And finally geometry sounds pretty literal, as far as the output, the manipulation or the generation of geometry. I want to use manipulation, again, in the sense that we're going to playing a lot with points in space in

Dynamo. So that would still be considered geometry in that matter, because those points are going to drive where the geometry goes. Disclaimer.

JOSH MOORE: Yeah, so I think for those of you are a little bit more advancement, been in Dynamo a lot, or you have some sort of coding or programming background, that probably sounds super simplified to you. But I want to, again, just make the point that when we're trying to get Dynamo usage, when you're trying to roll it out in your firm, maybe you're going to be the guy that is writing the scripts that's going to allow other people to use that on projects. You have to make people comfortable with opening this thing.

And if you can relate it to them and teach them that, hey, it's not really that much different Revit. When we're talking about strings, it's just text. When we're talking about all these things. So I just want to mention that. We totally realize that we oversimplified that a little bit. But hopefully that's a good way for you to help relate it to folks in your office.

So we've got a few examples we want to talk through, just some basic things before we get into Dynamo with some live examples that we want to kind of go over. One of the big things that you're going to have to do in Dynamo is convert data types. And this example on the left, converting a point to a string-- So one of the things that I did when I first started learning Dynamo was I didn't understand what functions are and all these more advanced mathematics things, or more advanced programming things. Probably most people don't understand that stuff.

I'm starting to get to that point. But he's over there now. So what we're doing here, basically, is just generating a few points. And then right here we've got a string from object node highlighted there in red. And all we're doing is converting the geometry to a string representation. And you can see, it looks about the same. But what we can then do after that is we can come here with a string.contains, and we can basically say, hey, does any of these things in this list contain that number, 0.3.

And certainly there's probably other ways to do this with functions. So you don't have to convert it to a string. But for those of us that-- maybe that just doesn't ring for you. You're not really sure how to do that. This is a really simple thing that I think most people can understand.

JUSTIN BENJAMIN: And it just came to me. As you see these nodes, especially the new users out there, what I've done in my office, is I've actually taken the bulk of these nodes that can do a specific command. And I made a library of these things. So if I want to do what Josh just mentioned, I

can just copy and paste, much like loading a family into the script in a building to utilize like.

So don't get so overwhelmed and see all these specific nodes and be, just man, I'm not going remember to do that one and then push that button. I really would take advantage of the packages that we're giving you, the scripts that we're giving you today and break them up and create your own library to do some of these conversions.

Now that said, what you're seeing over here is what I get really pumped about. And again, if we think back to Revit, you ever create a parameter that is text? And then you try to use Calculate Totals in a schedule or try to do some type of calculated or conditional statement with that parameter, and you find yourself lacking? Even if you put numbers in this parameter, but it's text. So it can't add it. It can't do it. I'm sorry, game over.

In Revit, you'd have to probably create another parameter. Make it a number or an integer, for that matter, or an area, and then copy that data from the old parameter to the new one. So in fact, you can use it for calculate statements, things of that nature.

Dynamo doesn't need to do that. Dynamo can convert it for you. It says, I got you. Let me help you out. And again, it's going to convert that text to a number so it can be played with. That's all that's really happening here, converting information so you don't have to create another parameter, for that matter.

JOSH MOORE: Exactly. And you notice right up here, we have that null value that Justin talked about. So this is a good example of Dynamo doesn't really take a string and try to multiply it by a number. It doesn't know what to do with that. So when we convert it to a number first, in the red box there, you can see that we actually get the result that we wanted.

JUSTIN BENJAMIN: A little feedback, did you want to move that down a little bit?

JOSH MOORE: I don't know what's happening.

JUSTIN BENJAMIN: You guys hearing that?

JOSH MOORE: Maybe we can turn it down in the back? You got me?

SOUND I'm trying to figure it out

ENGINEER:

JOSH MOORE: Excellent.

[INTERPOSING VOICES]

JOSH MOORE: So I'll talk a little quieter. How about that? So what we have over here in this next slide is talking about data structures. So one of the hardest things that I had to learn in Dynamo was specifically regarding how to structure the data in such a way that all of my information downstream was working properly. Super hard, at least in my mind at first, to get your mind around.

So what we want to do is talk to that. So that way, hopefully you can have a little bit better understanding of this. So on the left, you can see that we have a watch node. It's just showing you a list of four numbers, zero through three. And on the right, we have the same data but structured differently. And this is really important. In Dynamo, Dynamo generally will expect certain types of data structures with certain nodes. So it's super important that you're watching the data structure and how you're feeding that into your nodes to ensure that you get the results that you want.

JUSTIN BENJAMIN: Again, oversimplify this. It's like taking the first node is sorting information, Revit sorting it just by rooms. And the second node is sorting first by department, and then rooms. Just again, it's just sorting that information, same data just differently.

JOSH MOORE: So here's an example of that and why that's important, really basic example. But nonetheless, hopefully you can get the idea. So we have those same two lists, the single list and then what we call the list of lists here or the nested lists. And if we just focus up here on top right, up there on the screen, you can see that when we add these data together, we get zero and zero. And that's zero. We get one and one. And that's not one. That's two.

So at the top right, we're not really getting the data we expected. But at the bottom right, what we're doing is we're introducing this flatten node. So we're basically structuring our list in a way that now Dynamo can help us. Now the addition that we're doing is doing what we want. You can see we've got the numbers there in blue than we expected.

JUSTIN BENJAMIN: Josh just real quick--

SOUND ENGINEER: I'm going to switch you off mic. Seems like you have bit of--

JOSH MOORE: I'm OK.

JUSTIN BENJAMIN: As he's switching him out, I'm going to talk to a couple of the bullets that I was going to address on this slide. And you'll go to his. My favorite ones out of this particular slide is the place face hosted light based fixtures on ceiling hosts. The bullet's a little special. But all you're doing is we're swapping out ceiling hosts of light fixtures with face-based light fixtures. We know we can't really do this simply inside of Revit. We can't go ahead and go into the family and convert it's hosting ability.

So we have to either rebuild it or create some type of routine that essentially takes geometry from one host, one family template to another. What Josh has scripted together-- so awesome-- is the ability, basically, to address or acknowledge all these hosted fixtures versus face-based ones and essentially push the fixtures in. Your point, essentially, another just little families, and they go, and they swap themselves out. It's, again, just way applicable as it relates, at least, to Perkins+Will and hopefully you guys as well.

JOSH MOORE: So the scripts in blue here are scripts that we're just going to talk to and not get into. But in the data set, as well as there's a link in the handout that can get you there, you are able to have access to these scripts. They are all commented for the most part, at least telling you what they do. And in the handout, it talks about them.

So you're welcome to use these. Tinker with them. Hopefully give you some ideas, generate some ideas that can help you do these kinds of things at your office. We call it, in my office, we call Dynamo hero. You've heard that before. There's been presentations about it. But it's so true.

There's a joke in our office. I'll go into meetings with BIM teams and stuff like, BIM meetings. And they'll present a problem. And they'll ask me, Josh, is there a script for that? And I'll be like, well, I think I can write a script to solve whatever that problem is. So a couple of other examples up here, going back to this first one, keyplan setter.

How many of you guys have had either yourself or somebody on your team literally go through hundreds of sheets checking a box to assign the key plan to the right sheet? You've done that before. And it takes hours. It's just super monotonous, laborsome stuff. So what script can do

is it can basically read your sheet number, because most of the time, you're putting that information in your sheet anyways. Like A101A, that's area A. A101B, that's area B.

You can tell Dynamo, hey, map this guy to a parameter that you're checking a box on the sheet. And now you rip through, 200, 300, 400 sheets in about 30 seconds. And you're done. So you're already putting that information. This is really great.

JUSTIN BENJAMIN: It removes, also, think about the user error also that it removes. I've had staff go through, and that's exactly what I meant to do. But they're also listening to music or some comedy album. And they're giggling. They're like, ha, ha, check box, whatever. And they move on. This removes that user error.

But that's it. By the way, I did see a hand raised. We're going to really-- we so much want to take questions. We're going to definitely have to wait until the end. We have a couple of email addresses, or a single email address as well, some Twitter accounts that we can point you to. I'm going to encourage myDynamojourney@gmail.com. We're going to reference call to touch on that at the very end. But you want to make notes of your questions, at the end, we would love the questions. And we will respond.

So anyway, again, we want to answer the questions. We got so much content.

JOSH MOORE: So let's keep going with this. We've got a couple of others on here I want to talk to. So replace carriage returns in room names. Who knows what a carriage return is? A couple of you. Carriage return as basically programmer language for somebody going into a Revit parameter and holding Control and hitting Enter. Have you ever done that trick, where you can force a return in a parameter name?

So get this. I had a project team. Somebody didn't like how-- you know in Revit how the room tags will just automatically adjust wherever they want. And sometimes they look really stupid where they adjust. So somebody was like, I really don't like the way they did it. So he went through 1,000 rooms and Control Entered. Well guess what problem that caused? In a room schedule, it didn't show the whole name. It just showed the first word.

Big problems, we got 1,000 rooms. And Talking to the staff, and we're like, they're going to have to go through 1,000 rooms and manually remove that carriage return. No, we're going to use Dynamo for that, because it only takes-- I don't know, I can't remember exactly-- eight nodes or something, five minutes to write it. Run through it in about 10 seconds, 1,000 rooms'

carriage return gone. Really simple things that you can do to solve everyday problems inside of Revit.

This next one here, push object level two parameter value--

JUSTIN BENJAMIN: Such a good one, good one.

JOSH MOORE: So how many of you guys have tried to schedule a linked model for doors or anything else? There's something you can't use when you schedule a linked model. The level-- what? Come on, Autodesk. Really? Come on, help us out.

So this is where Dynamo helps us out. So what we can do is we can say, hey, Dynamo. Go tell me where all my doors live, what level they're on. Grab that level name. And push the level name inside of some parameter value. And we can rip through 100 doors, 1,000 doors, 10,000 doors, 30 seconds. Now we've got the value in comments or whatever parameter you want. And all of a sudden, we can start sorting, filtering whatever you want by that level name. Fantastic.

JUSTIN BENJAMIN: Huge deal.

JOSH MOORE: Really good.

JUSTIN BENJAMIN: The last one-- I know it's not blue. By the way, blue just means we're going to talk to it today. Black means we can't really address it. But again, we're giving you the scripts anyway. But I got to talk about transferring a project from one file to another.

I don't know how many times, whether or not I want to use the Perkins+Will from our firm template. And then all of a sudden, we're pushed. Be it, maybe there's another version of that Perkins+Will template that we want to upgrade to. Or maybe that the client is pushing their own template. And if I have this geometry, phasing, works, that's all pre-established in the Perkins+Will template. And I need that content to migrate over to this other template environment.

This script saved my life. This was able to take it, make it so that it retains works sets. It retains phasing. That's existing, demo, and new. And it moves that geometry from one project environment to another. This is otherwise impossible, as far as I'm concerned. Copy and

Paste does not work. Losing phasing settings, you're losing work sets. You can't link in and bind, again, losing phasing work sets. This is such an awesome, glorious reason why Dynamo's super duper.

JOSH MOORE: So just to briefly expand upon that, because we didn't do a write up on that one, but it's basically two scripts. The first script goes and finds all the objects phase values, phase demolished, phase created. Turns it into a text value. Pushes it into a dummy parameter. We call it dummy work set, dummy phase created, dummy phase demolished.

And then there's a second script. So then you take the model. You bind it in. Well guess what? Those particular text values remain. You bind it into the new model. You run a second script that goes and reads those dummy values and reassigns phase created, phase demolished, and work set value based off of all that.

JUSTIN BENJAMIN: So basic, but so glorious. It's really just-- think about them interpreting that workflow. That's all that's happening. We're not geniuses.

JOSH MOORE: So we've obviously got several others over here that we put on there. Some of them you may have seen before. But definitely all these things are possible. We've got scripts that do them. And I'd be happy to send you some screenshots or something about how those are set up.

So we're going to move right into the good stuff. And we're going to level this up. So first script is basic. Second script's going to be a little bit more advanced. And the third script, hopefully, will blow your mind. It's a bigger workflow that we've been working on. But it is more advanced. it is more advanced. So I'm going to let Justin-- he's going to do the intros for this. And you want this?

And then after that, I'm actually going to open up the script. And I'm going to open up the script. And we're actually going to run it live. Hopefully it's going to work. You know what they say about doing things live in a presentation? How many of you guys have failed completely, because something didn't work, Revit didn't open, whatever?

JUSTIN BENJAMIN: This is going to work, though.

JOSH MOORE: It's going to work.

JUSTIN Come on. We got it.

BENJAMIN:

JOSH MOORE: You have faith in us. So we're going to open this up. And we're going to run it. And then I'm going to briefly walk you through the overall logic of these scripts. Just so you know, also a part of the data set, all three of these-- well technically four scripts, because the third one is two scripts-- but all of these scripts have a video that I've literally walked through very specifically what I've done at each step with the logic, and what the nodes are doing. So those videos are in the data set for those.

So we are going to go through these a little faster, because we only have an hour and a half. Took me like half an hour to explain each one of these scripts in depth. No way we have time for that. You have a video. So I'm going to give you the overall logic ideas. And then you guys can continue to research on your own.

JUSTIN BENJAMIN: And for after the video, I further dumbed down each one of the scripts, just again because I need to simplify it for myself. So the handout has the images I'm about to run through, as far as the overview of the script. But it even goes down in detail, in the handout, each node-- a series of nodes rather-- and what they're doing in layman's terms. No offense, he's just way too smart. So that's accessible too.

That said, let's do it, script one, exporting views to Excel containing a category. I like to see this really more for the BIM managers, the model managers. You have a situation where you, maybe, have hundreds of text styles hundreds of dimension styles, things of this nature. And really you would like to simplify this. Whether this is in a template or your project file, you don't need your staff going through and having to select from 20 different dimensions styles, 30, 40, different text styles. It's annoying. It's frustrating. There's no consistency.

So what this allows you to do is query that data, really understand which text styles are being used the most. And then hopefully you're able to then go inside of Revit it and simplify it. Delete. Select the ones that aren't used that often, rather. Swap them out for ones that are used often. And then delete. So that's what this is going to ultimately do for you. It's a very simple script that allows you to query data.

So when you launch one of Josh's amazing, super-duper, glorious scripts, you are in an environment where he will start to here. This is a brief summary of what's going on inside the script. And then this is essentially what you, the user, have to do. Everything else is back end stuff.

So going into what's happening over here, all we're doing is pointing essentially-- I may actually go back a second. To run or utilize Dynamo, we do have to open up Revit. We have to open up the project file that we would care to query. And then go to the Add Ins button. Activate Dynamo. Dynamo pop up shows up. And then you browse to the script that you'd like to utilize. So hopefully that's a pretty simple workflow to gets you right here.

Once you open up one of Josh's scripts, this particular one says OK, where do I want the Excel file that's going to have the data that's coming outside the Revit file, the that I'm querying? Where do I want it to fall? That's the first one, is where do you want-- where's location?

The second one is merely saying, inside of Excel, you have the worksheets. So sheet one, sheet two, which you can of course name. That's just saying what name you want that worksheet to be. And then finally, what do you want to query? This is a drop down menu. You can query text, dimension styles. You can find them and do a couple.

So the image over here is the initial output that script's going to give you. I know it doesn't look super attractive. We're going to show you in a second how to make it more attractive. So this particular script, when it was run, it's pushing out this information. We see the views that these dimension styles are in. And of course, you may imagine this Excel file gets pretty long.

The next image here is just saying the exact same thing. I've just swapped out the dimension. I swapped to say text notes. And so now we see the views and the text notes that are in those views. Now again, this raw data, not super duper great. It's not exactly something I could read easily.

So that's why we encourage you to use pivot tables inside of Excel. We're outside of Revit. We're outside of Dynamo. We're just using basic Excel tools, a pivot table within the Excel environment. You can then sort this a bit more constructively.

The first image here has the information sorted first by view, and then by what's inside that view as far as text styles go. So we see that there's only, in this one view here, floor plan, code compliance, [INAUDIBLE] legend. We have this single text style inside of it and then all the views according, not exactly super constructive in my mind just yet.

To the right, the next image in the center, rather, is sorting first now by the dimension style--

I'm sorry-- text style, and then all the views it's in. This is a little bit better. Hopefully your template or your project file has only the amount of views that one needs to start a project. So you can see where this text styles lives.

You can take this another step farther and again, just using simple Excel to query it, you can find out how many instances of this text style live inside these views. Again now we can see that this particular text style is only used x amount of times 129, 119 times or whatever. So we can, again, hopefully start maybe purging some out at this point, again just querying data.

And why is this particular script especially impressive? Because it queries legends. So if you're thinking that maybe I could go inside of Revit, select a single bit of dimension, a dimension style or type for that matter, right click and Select All Instances and find out how many times it's been used, that particular workflow does not include legends. So you're really giving misleading information. This script does. So again, a BIM manager kind of tool, just querying data. Let's go live.

JOSH MOORE: Just to add to that, I can't tell you how many hours I spent manually going through legends trying to find where text styles and dimensions styles. Anybody else done that? Seriously it's-- and this will tell you where that stuff lives.

So here we go, going live, going dangerous. Export styles. I don't think I need that. So we have a model open here. This is actually one of our project models. It's got a lot of stuff in it. And you'll see that when I export this to Excel.

So what I'm going to do, you can see I've got a lot of legends here. And I'm sorry this is so small, guys. Who works on one screen in Revit? Probably nobody. And it's really helpful, when you're in Dynamo, to throw the Dynamo screen on your second screen. So bear with me here when I'm trying to do all this on one. But you can see we've got all kinds of sheets and views and stuff like that. So I'm going to pull back up Dynamo or attempt to here. Which one is it? That one, I think. Nope, it's not. There we go.

So what I'm going to do first-- so again to what Justin was saying, I've got these-- I'm just going to maximize this. I've got these three nodes here. And I'm just going to leave them the way they are. But you can definitely change this path. Change the category like Justin showed. I'm just going to come over here and hit Run. And it's going to query this file.

And there is my Excel file. It's writing that data out. Give it a second. You can see at the

bottom left there it says, run started still. We're going to wait until that says run completed. So I got a few warnings. That's probably OK.

And you can see, there is my script. I've got text nodes tab at the bottom left. And it's entering every instance of every text note in the whole project. And I can scroll down through here. And I've got probably 1,000 or more of these things.

So anyway, to show what Justin was talking about, what I can do is I can say I want to go to Insert Pivot Table. Excel usually does a pretty good job with basic data, just to generate it like this. I could absolutely add a header in there. It'd take 30 seconds or less. But I'm just going to leave it the way it is. I'm going to click OK. And now I've got this pivot table. And I can actually just drag these guys down here like this. And there you go. There's one methodology. Or I can flip-flop these.

And now I'm seeing, I've got this text note in all of those views. And if I want I can come in here and count those in values. And now it's telling me how many of those text notes I have in each one of those views. And we could do the same thing with dimension styles. And in a few minutes, those of you who are not aware of this-- well actually, I take that back. We were going to spend some time plugging. So I'll plug it right now.

With Dynamo Player, imagine you can just run this on the fly without opening anything up. And for those of you who haven't heard already, if you are working with IMAGINiT Clarity-- anybody using Clarity at their office? They were supposed to be announcing Dynamo automation in Clarity here at AU. This is fantastic, because now you can actually set up these scripts to run overnight with nobody doing it, outputting data somewhere and literally start querying this data however you can imagine.

Imagine running Power BI or something like that on Excel files to give you reports of how, maybe as a BIM manager, how optimized your projects are. That's it for the first one.

JUSTIN BENJAMIN: Did you want to go into the script itself.

JOSH MOORE: I did.

JUSTIN BENJAMIN: Like actually inside of it?

JOSH MOORE: Oh yeah, sorry. Thank you. See, he keeps me straight. So what I wanted to do is just talk through the script itself. So obviously here is the first part of it. And basically what we're doing with this script is-- and there are other ways to do this-- this script will only work on a work shared file. I don't know many firms or many people that don't work in a work shared environment. So that's generally a safe thing.

What it does is, those of you who've ever seen this before, if you click on any text note or annotation element in a work shared project, what's the work set of that element? Anybody throw it out there. The view name.

JUSTIN BENJAMIN: That was impressive. Well done.

JOSH MOORE: Amazing. So all we got to do is just get the work set of the element. Pass that right through. And that's literally all we're doing. So first of all, we're getting all the text notes. You can see-- where's our little, thank you sir-- you can see we're getting categories there. Text note, and we're going to get all those elements in the project, every text note element, every instance of that element. And then we're just saying get the type name.

And then that's what this node is doing. It's just getting the element.getparametervalue by name. And what that's doing is just getting the type name. So then the only other thing we're doing over here is you can see this awesome node right here, element.workset. And that's getting the work set name. And if I hover over that, you can-- actually that's getting work set element, excuse me. So hover over that. There you go. Autodesk.revit.dv.workset, does that help anybody? Anybody know what that is? That doesn't really help anybody.

So we're going to use this next node next to it, element.nameuniversal. These are both Clockwork nodes I believe. And look at that. Now we've got the name, fantastic. So then all we're going to do is we're going to create a list of both of those. So moving over next, we've got a list of the type names and a list of the work sets, or really the view name at that point. And then we're going to have to transpose that list.

Now the reason we transposed that list is because if we don't, in Excel it's going to push everything to thousands of columns instead of two columns and a bunch of rows. List.transpose is really important. I don't really understand why list.transpose just isn't built into the Excel node, because I think I use it every time before I pass information to Excel.

So then this last note here, Excel.writetofile, all we're doing, that's just the basic Excel node where you're passing in the file path, coming from the beginning, passing in the sheet name, and then the data. Start row and end row, all that is, usually, like zero in Dynamo is just that first A1 spot. And then this is the data we're passing in. And That's how we're getting it.

JUSTIN BENJAMIN: Again to reiterate, the handout has every one of these, be it series of nodes, with little call outs, so again trying to simplify. The video also that he's providing is quite literally what he just said. So between the two of them, we hope that you guys can really follow along and retain this after you leave here.

JOSH MOORE: How many of you guys think you can use that in your office?

JUSTIN BENJAMIN: That's 100%. I see it. It's 100%. I'm really good, really good with numbers. Script example number two, I honestly didn't even know this existed. Josh, when you showed it to me, I was blown away. This is super cool, guys. Basically try to imagine for a moment that you have a topography that no doubt was generated probably from a CAD file provided to you by the civil engineer. or you did it manually, however you did it with points.

Point is you have this topography. And maybe you want to put roads on there. Maybe you want to put it so that there is a parking lot. Maybe you want to acknowledge the sidewalks, things of that nature. Now granted, there are tools inside of Revit in topography, subregions, what services. But those are not glorious. We all know that. Not to mention that they're not glorious, they also don't provide hosting environments, like floors, if I want to host content to that bit of geometry.

This tool, basically what it does is it looks at topography, looks at a floor surface. And if you're familiar about when selecting a floor surface inside of Revit, you have that option. So you modify sub elements. And you can add points to that floor and give it heights. This often is used for drainage on a roof or in a parking garage inside of a building. You're just trying to make it so that slab has divots. This basically, it creates those points inside these slabs and maps them to the topo. Wow, that's crazy. That's crazy.

Let's get into this. Let's do this. Are you pumped? I'm pumped. Let's make it happen. I get really pumped for so many things. So as I mentioned last time, when you open up one of Josh's scripts, he's gone ahead and provided you with a brief overview of what's happening. How awesome it is, created by Josh Moore. Always want more of this guy. Get it, Josh Moore? Hey oh!

Then we have, again, the start of the script. We want to just input a series of bits of information. So starting from the top working down, I look a number of divisions as essentially pixels. How accurate, how clear, essentially, does the floor match with the topo? And by the way, the next slide is going to further embellish what I mean by this, so as far as a graphic goes.

So the first one is saying 20 pixels per inch versus just the accuracy. Now you may be enticed to increase that number from 20 to 100 or 1,000. And yeah, it gets more and more accurate. But then try running the script. That's going to add a lot of time. There's going to be quite the lag. You'll find in this case, 20 is plenty.

The next element here is the edge point spacing. All this is really trying to say is how often-- well, this is accuracy again. But instead of the body of the surface of the floor element, it's the edge of the floor element. And how many points or how accurate do you want that to be? Again, so you could go from 12 to 20 to whatever. But again, that's unnecessarily accurate.

Finally, offset above topo, love it, love, it, love it. If we want to, essentially, create a curb that's four inches off of the road, let's just say the road is matching perfectly, aligning flush without the topo. Maybe I want to jump up a bit for the curb, four inches to give that depth, something you can't do with the other tools out of the box inside of Revit, or topo, rather. So those three things essentially are helping with accuracy and whether the offset above.

The next thing here select first in model element. So select the floor that you want to warp. And then select the topo you want that floor to warp to. Boom, that's pretty awesome. So accuracy, and then what what do you want to play with?

So the next image gives you a sense here. We have the topography right there and right there, selected in blue. We have a floor service above it. A little tip and trick. You're going to want that floor service to be above the topo. You don't want it to somehow be intersecting that topo at any given time. It will potentially error out a little bit. So even if it's 400 feet above the topo service, that would be better than if even at one point it intersects it. So that's the floor surface. That's topo.

Over here, we ran the script. And on the the floor, you can see now number of divisions are those center points, giving you that level of accuracy. You can see it's pretty accurate. And then the edge point spacing, so that's just the 12 inches. Every 12 inches, there's another

point. So again, just helps with accuracy.

You do have to run this on each piece of a slab, essentially, that you have. So you'd first run it on, say, the roads. And then you'd have to run it individually, this curb, and that curb, then that curb. But I'm pretty sure the output is worth it. how? Do I say it? The juice is worth a squeeze. Because the juice-- do you get it? Love it.

Let's keep going. This further explains the image. I've gone ahead and isolated the floor object. You can how accurate it is. The one thing you might not love about this is after you run this routine, that triangulation, it's going to show up. It's going to show up both in plan and perspective. And that's gross. I wish Revit knew how to handle that. It does.

So if you just go to any one of the viewers that you don't want to see, that of the triangulation, just go to Visibility Graphics. Expand Floors, and uncheck Interior Edges. And you're golden. Looks smooth, it's cool. It's glorious. You can see, at this point, we've ran the script on those other floor elements, so the curb elements. And it played out well. Maybe can we go to--

JOSH MOORE: So we're going to go live. We're going to do it.

JUSTIN BENJAMIN: Let's do it. Let's play this game.

JOSH MOORE: So I have to ask, because I've done it. How many of you have actually drawn a floor and tried to manually warp it to the topo? Come on. Yes. This is for you. You've done this before. Somebody came here. And they were like, we need a rendering. But we need to show the curbs. And you're like, crap. Like That's going to take forever. And you're manually pulling points down. And it doesn't really look that good most the time, unless you spend a whole lot of time on it.

Well, enter Dynamo. Here we go.

JUSTIN BENJAMIN: I say dyna, you say mo. Dyna--

AUDIENCE: Mo.

JUSTIN BENJAMIN: I love it, love it, love it, love it.

JOSH MOORE: So here we go. So I'm going to minimize-- I'm going to pull this down for just a second, because the last time I minimized it, it was annoying. So we should have the same exact thing that Justin just showed. You can see there, there's our floors. And our curbs are modeled in. And all we're going to do is I'm going to come over here to this part here. And I'm going to change what I want.

So we're going to leave 20 alone. That's fine. 12 is fine. But what I do need to do for the main floor, for the road, is change that back to the zero. And then all I'm going to do is use this Select Model Element Floor and say Select, and come over here to Revit and pick on the floor. And then say Select Model Element Topo. Select, pick on that topo.

Now this one here takes about 20 seconds, I believe to run. Let's try it out and see. So again to reiterate what Justin was talking about, it's mapping all those points, doing all the geometric calculations, if you will, on that floor object over here. And in just a second, as long as technology cooperates, we should see that floor jump down to the topo surface.

Now all the little blue points you are seeing, those don't come through into Revit. That's Dynamo's geometric representation of what you just did. JOSH MOORE:

JUSTIN BENJAMIN: Wow!

JOSH MOORE: There you go. Now you can see it.

JUSTIN BENJAMIN: Oh my heavens. No way, no way.

JOSH MOORE: So this is really cool. So this saves you a lot of time. And now when you're ready to run the curb, all you got to do is come back over here and say Select Floor. Pick on the floor object. And hit Run again. And this one goes a little bit faster. I forgot to set the offset. But can I just come here and do that? I made a mistake. Can I just come here? Six inches, run.

JUSTIN BENJAMIN: Was it a mistake? Or was intentional oopsie-daisy?

JOSH MOORE: Will it work? Let's see.

JUSTIN I'm excited. Yes it did. Oh man, again.

BENJAMIN:

JOSH MOORE: There you go. So now your curb is mapped to that topo surface. Now I'm not going to do the other four curbs. But you can see it goes pretty quick. It's not that bad. Now obviously if you had a humongous floor object, the number of divisions you're seeing in the body here would need to increase. And the time would go up. And obviously it would depend on the speed of your machine and stuff like that. Just like anything else, it's churning through all those points and doing all the math and stuff like that.

So let's walk through the script.

JUSTIN BENJAMIN: Let's do it. Let's play this game.

JOSH MOORE: So what I'm going to do is I'm going to maximize Dynamo. You've already seen this part. So I'm not going to talk to that again. You can see the craziness that is points and things like that in the background. That's just all the things that Dynamo's doing to make this thing happen.

So this one is going to be talking more to geometry. And to be honest, this is one area that I haven't gotten into as much. So when I start throwing terms out there like polycurve and all these kinds of things, I'm going to try to explain what I think they are. But to be honest with you, I'm just making my way through it like some of you guys.

So some of our math whizzes and folks out there that know all that geometric terminology, you probably could give a better explanation than me. But at the end of the day-- sorry-- who cares, because it's working. So here we go.

So we're going to move over here. And what we've really got going on at the beginning is two parts. So first, we're feeding in the floor object on the top. And obviously Dynamo's doing this simultaneously up here. And we're using this node called collector.elementsketch. And we're basically getting the outline of the floor. That's what that node is doing. It's changing the floor object and getting that sketch outline that we created. That's all that's doing there.

And then it's outputting Dynamo curves. The curves are just those line representations, arcs and lines stuff like that. And then down here, what we're doing is we're getting the mesh of the topography. And remember when I talked about converting data? So the node I need next is creating a surface. And I'm getting the surface and projecting it onto another surface. So if I tried to pass the mesh in here, those of you who know Dynamo, will it work? No, because it

wants a surface, not a mesh.

So we've got this awesome springs node here that will convert a mesh to a polysurface. Bam, done. So then I pass a polysurface in there. And up here what I'm doing is I'm getting the curves of the floor. And I'm passing that in there.

So what's going to happen is I've got my topo that I've just gotten the polysurface of. I've got my floor outline. And what that project node is doing is it's saying take this stuff and just map it onto that curve thing. That's all it's doing.

So we're going to keep moving over. And basically we're generating another polycurve about the curves that we generated here. And then we're basically patching a new surface. So what's happening is that the output here is just the curves that are mapped onto the topo, now we need to create a surface out of those curves. And that's what this surface by patch is doing.

So we keep going over. Like I told you, this is getting a little bit more in depth. We've got three things that's happening. At the top, we're generating the body of points. I call this the field points. So we're basically just feeding in the surface and doing a little UV division. To be honest with you, I don't even really know what UV is. But that's what it's doing. It's a local coordinate system, as I understand it. But that's basically the extent of how I understand it.

The second step here, what we're doing is we're getting the edge points. We're determining the edge points. And then this last one here, this one's tricky. I had to add this in at the end, because I was getting unexpected results.

So here's what happens. This, you would think, would cover the corners. But because I'm just evenly spacing along the perimeter of the surface, I don't always get a point right on every corner. So what was happening is some of my corners were just sticking up in space. So I had to add this third section, basically, that more or less just gets all the corners of every line segment.

So I've got three sets of points. And I move over. And I'm going to put those points into one master list. I'm going to flatten that list, because again the data structure that I'm expecting downstream is most likely not a list of lists, but a flattened list. And actually it doesn't look like, in this case, I even needed to flatten it. But I did just in case.

And then what this node is doing is it's basically saying, OK, what if I have two points on top of each other just by happenstance? Who knows? The field point ended up on an edge point. Or an edge point ended up on a corner point. Well what I wanted to ensure was that the output that's happening wasn't going to do anything weird. I don't want points on top of points when I'm feeding that into the output.

And then the last piece here, before we actually feed in the output, all it's saying is do I want to move those points up a distance? So remember the offset value at the beginning? So I'm basically saying, do I want to move that up a distance? If it feeds in zero, it doesn't do anything. If I feed in six inches, it's going to take 6 divided by 12, because everything's in feet in this case, and convert that to inches. And then basically go in there and give us the point moved up whatever distance that I'd specified.

And then the magical node right here, floor.slabshape by points. So this is great. You pass in the floor object. You pass in all the points. Dynamo does the rest. Pretty fantastic, so that's pretty much it for this one.

JUSTIN BENJAMIN: And I want to just reiterate one more time. I'll do probably a handful of times, is that the handout blows up each one of these nodes, talks to it. Video further elaborates upon it. The terminology we used way back when, the string, the Boolean, the integer, the number all that, you'll notice in the nodes that he's using, actually have that information identified somewhere. So if it comes to, I don't know where do I start? Where do I start typing in to find the right node? The language is built in to what we're providing you with.

So don't be overwhelmed. Don't think, I'm not going to be able to recreate this on my own. This is craziness. One, you don't have to recreate it, because we're going to give it to you. But two, you could actually, again, do a Save As of these nodes. Create yourself, again, that library I mentioned initially, where you can start reusing collections of nodes, just to get yourself in there. And three, I don't have a three. it's [INAUDIBLE] on the one, two.

JOSH MOORE: I got a three.

JUSTIN BENJAMIN: Yes, do it.

JOSH MOORE: So here's three. Also in the handout, what we did was we actually-- so this is kind of funny. So I had an idea one day. I was like, what if I could figure out what nodes I used in all of the

scripts I've written, and how many and get an idea of what I'm using and share that with people?

So I was like, I wonder if I can write a Dynamo script to read all of my Dynamo scripts.

JUSTIN BENJAMIN: Googling Google. That's Googling Google.

JOSH MOORE: Sure enough, you can. You can absolutely parse a Dynamo script with Dynamo. Just basically Dynamo's more or less an XML file. And you can go in there with some little wizardry and read that. It was a little tricky. But for the most part, it worked.

So in the handout what I did was I gave you a chart that basically tells you-- it's two charts. First chart in the middle of the handout basically shows you every node that's used and about 100 or so of the scripts that I've written, just to give you an idea. So if you're wondering, what nodes are used the most often, those types of things. That's a good place to start.

Then at the end of the handout is an appendix. What I did was I ran the same script, but only on the scripts that I'm giving you. And I made an appendix. And I basically said, where do these nodes and all these scripts we're giving you, where do they live?

So now you have a context. You're like, I don't know how this node works. I need to see it in context. First of all, Dynamobim.org is a fantastic place to go to ask questions about that. But if you need a context, and you want to open one of the scripts we provided you, that appendix will tell you where that node lives in the specific script. Now obviously, it doesn't cover every node, because it only shows the nodes that I've used.

JUSTIN BENJAMIN: Super cool.

JOSH MOORE: So here we go.

JUSTIN BENJAMIN: You're welcome.

JOSH MOORE: The finale kind of.

JUSTIN BENJAMIN: Now this is the powerhouse. This is it. This is the beast. This is gloriousness defined, as far as I'm concerned. At Perkins+Will anyway, I'm assuming not just at our office, we have projects

that utilize multiple Revit files. Just one file for everything that wants to happen inside of a building is not possible. So whether or not you're breaking up the interiors, the levels, the shell itself, your point is you have multiple Revit files.

And wouldn't it be great to be able to reference details in other project files? The ability to say, I want to pull the exterior elevation tag inside this interior first floor plan. But man, the exterior to the corner shelf file's over here. And unless I use a dummy tag with no links, or maybe I export-- we've done this in the past. We've exported just the view references of project files to CAD and linked those CAD files inside of our other Revit files. And so that every time you wanted to update those references, you just have to replace that CAD file with the latest push.

That's not glorious. That's not great. That's not super duper. We don't want CAD anymore Come on. This is doing something that really is not possible, as far as I'm concerned. I know there are tools out there that are able to cross-reference Revit files as we're about to. But I find them limited. In some instances, they only can point to a single Revit file. So maybe I have 10 Revit files, and I can only point cross-referencing details to one. So all 10 files can only point to file A. I can't spread this apart.

There is no limitation to this. You could have 10, 20 Revit files. And they're all talking back and forth to one another. That's that super-duper and gloriousness. Come on. So that's what's happening here. Again, going through what's going to happen first. Get into Revit, into Dynamo. Do know this particular routine is utilizing two scripts. You've got to run one, then do some stuff, and then run the second one. And you'll find that what we're doing is really not asking to do too much more than what you're already doing when you're doing some internal cross-referencing.

But here we go. Starting with the top, the Boolean thing, the yes/no, you only need to worry about this particular pop up with a certain particular node when you run this script for the first time. And what's happening is you're pointing, essentially, to an Excel file that's pre-formatted.

Let's go back a second and think back to the first script I showed you. And how when Dynamo pushes that data to Excel, that raw data is a little hard to interpret. It's a little hard to read. it's just these cells or these columns are not auto-adjusting by default. There's no header. Data is not easy to browse.

So the first thing we're doing is we're providing you with an Excel file that looks at the raw data file that's going to come outside of Revit initially. And it's pre-formatted for you. We've already

given it headers. We've given you all sorts of good way to view that data.

So the first thing is saying, hey look. This is the very first time I'm using this particular script. And I want you to copy this formatted Excel file. And jumping right down to this here, all we're saying is OK, look, I"m going to copy this format Excel file. And I'm going to browse, and I'm going to paste this template file at my project location. So project X wants to browse, essentially, to my template location. Get that Excel file. Copy it. And Paste it to my project location. That's all it's doing. So I'm copying and pasting it. And then we're going to give it a name.

So again, where does the template file live? Where do I want it to paste to? And what do I want its the name to be? The center part here is essentially the raw data file. Where am I going to paste the raw data file that this other formatted file is going to look at? Where am I going to paste it? And what name do I want it?

So all you're doing, you've gone ahead. And you're going to have to basically run this script on every single one of your Revit files. And what's happening is it's just pushing all the views into this Excel file. All this stuff, this formatted environment, everything I'm talking about, it's basically creating an Excel file that looks like this.

This is actually the raw data file that we just auto-adjusted all the lengths. So it's pulling data like the sheet number. And in fact, the view inside this file has been placed on a sheet. This is the magic. We're going to talk about that a bit. The view name of all the views in this project file, the tile on sheet-- in fact, there is a tile on sheet-- and then finally the sheet name of placed. So we have the view, the sheet number, and the sheet name, and a little bit more about the view.

So again, if you have 10 Revit files that you want to play this game on, all you got to do is open up each one. Run the script. It creates this raw data Excel file that just has every view. You got your plans, your elevations, your drafting views, you name it. Every view's going to push to this Excel file.

And then what's also doing-- this is the magic part here-- it's creating a unique identifier that basically associates to the view as it relates to the project. And that's essentially the magic. That's what we're going to be cross-referencing. And we're going to talk about that in a bit.

This, by the way, is that formatted environment. We told you we wanted to give you something

a little cleaner, a little easier to read. So you can see here, this format file already has headers, the view GUID, the reference, the view name. So it's prettier. Auto-adjust has already-- it's more attractive. And all it's doing is looking at the raw data file that's coming outside of Revit initially.

And all this pathing is already set up for you. You literally just have to essentially point to the Excel file that's he's formatted for you, and then click Run. Point to the Excel file. Where you want to save it, and Run. And then in one folder, you would create all these Excel file. You have 10 Revit files. You have 10 Excel files in this one folder.

Then what you have to do, now that you've pointed essentially-- now that you've created these unique identifiers, these GUIDs, for each view and every project file, now you just have to go to the projects that you want to cross-reference. So let's just say for a moment-- actually I'm going to jump to this real quick-- this is what's happening.

So I'm going to go into project file. Not going to do that. I'm going to select these dummy family tag references that Josh has created for us. We have a call out. We've got section references. We've got elevations. We've got them all. You're going to select each one. And then you're going to paste the unique identifier of the Revit file that has that view.

So going back, what you do is you'd say, OK, I want to reference or cross-reference a detail in another file. First I have to go in the Excel file that was pushed. That formatted file we talked about, I'm going to go inside of there. That was the output of running the first script. I'm going to find the view that I want to reference. And I'm going to copy the unique identifier. And upon copying it, I'm going to paste it into the view reference of the other project file.

Now before you think this is a lot of stuff to do, before you start saying, I'm already overwhelmed. This is this too much. This is really no different in my mind than clicking on a cross-reference-- an elevation, rather-- elevation tag or pulling out a call out, checking the box that says Reference Other View, and browsing to the view that you want to reference. All we're saying is instead of browsing to the view you what to reference, paste this GUID and to find the GUID you just got to go into the Excel file that's pushed and copy that data.

Once you've done it, once you've pasted this information, then you have to run the second script. Now again Dynamo Player is going to play the game for you. It's going to do this pretty much automatically, as I understand. But the second script says, OK, now all these dummy tags have all these GUID that's been pasted into them. Let's update them. Let's populate

inside that view reference, that view number, and that sheet number.

So what are we saying? The first thing we're saying is browse to the folder that has Excel files. This is awesome. This is so cool. I'm not browsing a single Excel file, clicking Run, then browsing to another Excel file, because I have 10 different files that I pushed data to. No, I'm browsing to a single folder that can have upwards 20 different Excel files. And when I run this, it's going to look at each one of those Excel files and see if there's a GUID that talks one of the dummy tags inside that project file. And it's going to populate for you.

So the first question is where is the folder that you want to play? The second one is name the sheet that contains the-- this is just again, automatic for you. So you're just really pointing to the folder. And then over here, it's like, where do you want the new Excel file? The new Excel file, by the way, the name and the location is just a report as you're going to see. You don't actually, in my mind, necessarily even need to do this. You do. It's good have a report. But technically, so as long as this points to the folder that has all these Excel files, you can click and run.

Now the report, you're going to find, is helpful. But it will work without it. And what's happening is you can see right here. So this is the report. And this is the output in Revit. Let's talk to this real quick. There are three conditions that would happen as a result of running this particular script. The first one is success. Yay. And by the way, all my handouts that have these little call outs, they're all a little bit of Justin in there. I apologize. It's all like, silly goose. Hope all is well.

So I got success-- yay-- in the sense that sure enough, the view reference and the sheet reference came in. Awesome. The next one is no match. what. Do you mean? That means that you went ahead and you pasted a GUID inside of this call out. And either A, you didn't get the entire GUID. Maybe you didn't get the last two characters when you clicked Copy and Paste. So it says, hey look, I got a GUID reference. I want to reference something. But that doesn't exist.

So you could have deleted the view that it was once referencing. Maybe that's the case. I don't know. But the no match means you did paste something in there. It just didn't play well. And then the last one is, look, you gave me a GUID. So of course I don't know where to go. Three different conditions, this is the report. Same conditions happen from there to there. So this is good information to see, especially if-- I'd almost sort this report by no matches and by no fills at all. So I know quite literally, I got some work to do, because I've crossed-referenced in

something that I'm not actually pointing any data to.

In my mind, that is the super duper. That's something, again, that Revit cannot do right now. Autodesk has not given a solution just yet. Not suggesting Autodesk isn't glorious. You guys are super duper, yay. But we're waiting on this, and so a super duper tool. It's free, again, so long as you have Revit, of course. That costs a little bit of money. But yeah, that's it.

JOSH MOORE: He's getting into the weeds.

JUSTIN BENJAMIN: Let's do it. You're guys are gonna be great today, by the way.

JOSH MOORE: So I just want to explain that we're going to run this on a dumbed down file. So I got the question earlier-- where are you sir, right there-- does this work on a big project? So I actually developed this for a big project. It was a million square foot hospital. They needed to be able to reference details across links. And they were going to have hundreds of them, just by the nature of how those kinds of projects are set up. These scripts absolutely work on a big project.

And speaking of the refresh report, when you're referencing hundreds of them and you need to figure out across multiple models where stuff has failed, that's why the refresh reports-- I call it refresh report-- but it's a report that basically tells you what happened. So you can go in and confirm, did all of the references get updated?

So we're going to hop into Dynamo. Now I'm going to give me a forewarning. These scripts are much more involved. So I'm going to, because we're starting to get a little behind. But that's OK. I'm going to give you a lot more overall logic. But I literally have 20 to 30 minute videos on each of these two scripts that go through each node more or less and talk to each of it. So I'm going to give you overall logic for today, so we can move on to the last two sections we have, which is tips and tricks and a little shout out. So here we go. This the one, yes.

So this is the Revit . I'm going to pull this down here. So I've actually, in my folder structure, I have mocked up-- let's see, where are we at here? I apologize I. I'm just going to go to manually. I thought I made a-- here we go. So in my folder here, in this detail linking folder, I've got several models. And just in case Revit 2017 was giving us fits, I had backup plan.

So ignore the other ones. We're just going to run on the 2017. And I have already pre-done two of these models just for time's sake. So in this folder, I've got the raw Excel data. Let me

delete that one, because that's the one we're about to run. So I've already done B and C. And then inside of here, there is my data template. And here is the data that's going to be-- the actual files that staff is going to read. So these are the templated ones. If I open this up, you can see. There's a templated one. And close out of this.

And then lastly, in this last folder, these are where the refresh reports go. So I'm going to delete that one also. So just to give you a context, I do have several models that have already run the Excel that we're going to be linking to.

So for the push, for the initial push of the data, I'm going to go ahead and hit Run. We've already talked about this. So you can see the result. And then we'll step through it. So we're going to hit Run. And what it's doing is it's reading this Revit file for all of the views. You can see them there. In this case, not very many. But on the files that I've run these on, it's been thousands of views.

And then what will happen is Dynamo-- I wish there was an option. If Autodesk is listening, please give us an option to not open these Excel files, because it's annoying when there's a lot of them. But nonetheless that's what we live with right now. And then this is the refresh report.

Now there's a little magic happening in the Dynamo script that actually generates the link to the raw data automatically. So initially it looks like this. But then when you close and re-open and click on Update, it just pulls in the raw data into the format. So this is what staff would use to search for a view that they need to reference, this formatted file here.

So just to kind of show you that, you can see that's already done. And if I go into here, if I was a staff member, what I would want to do is go open this push data. Click on Enable Content or Update. And there you go. There's that data coming from the raw data file. And then I would find the view that I want, like Justin was talking about. So let's say it's this one here. And I could copy that.

And then I would come into my Revit file. And let's say I've got this section marker here, this dummy tag. What I would do is I would paste whatever reference I need, like Justin was talking about, into that parameter there. Now since you're getting the sample Revit file, good news. You get these families too. So feel free to use them.

JUSTIN BENJAMIN: That's amazing. Thank you Josh. Loving him, loving him.

JOSH MOORE: So I'm not going to go into how these families work. I'm confident most of you will be perfectly able to figure them out. But there's all kinds of good stuff in here that you can-- one of these days, I want to teach a Revit families class, but for another time.

So basically paste that in there. Paste that in there. And then you're ready for the second script. So I'm going to close this first script. By the way, I've already pre-populated this file with several GUIDs. So you'll see this actually update with several instead of just one.

So I'm going to close this Dynamo script . I don't need to save. And then I'm going to open up the second script. Now I'm going to come back to this in a second and step through the scripts, the overall logic.

So once I've got that set up, I would do all my pushes on all my files, which by the way, you need to do before you run this script. So let's say you initially start this up. And then you've added a bunch of views that people need to reference. You would need to rerun these scripts periodically to keep it updated. But that's where Dynamo Player or the Clarity Dynamo stuff could really help you out with this process where maybe you don't have to have to hardly run them manually at all.

So again this script, you would browse to where the folder of Excel files are. And then this is just the sheet name in the Excel file you need to be looking at, and then the names of the refresh report. So I'm going to go ahead and hit Run on this. So zoom in on that, so you can see it run.

It's going to open up again. Please Autodesk, give us a way not to open up all these Excel files. It's going to open up all those Excel files in the background. It's going to read them, whether it's one, two, 10, 100. And you can see, it put the reference in there from the other model. The B stands for file B. So I made that clear this is file A. So that is actually coming from the other model, pretty cool. And then if we move over here, you can see the call out filled out. And then this other section also filled out there.

So who thinks they could actually use that on a project? Yes, some of you. Some of you work on big projects? Yes, it's great. So a little bit involved, but for the most part, it works pretty good. And when you don't have any other solution that's a good solution, the only other manual way to do it is to manually check them all, all the time. And this is a great way to go in between there.

So we're going to open up this first script. And I'm going to talk to major logic here. I'm going to maximize this. So just to zoom out and give you a little bit more context. So this one isn't quite as involved as the second one. But you can see, it's much more involved than the other two we showed. So we're stepping up our game here a little bit.

So what we're really doing-- and I'm going to talk more to bigger chunks for the class-- but again, I go through all of these individually for the most part and what they do. What we're doing here is we're basically getting the central file name. So that way, there's no manual pathing to the stuff. We're getting the central file name and the central file path of the file that you're in. And we're going to use that downstream for various couple of things.

And then down here at the bottom, what we're doing is we're getting all of the views in the project. And then all we're doing is some filtering. So we don't want view templates, which unfortunately this will get you. We don't want sheets. We don't want several different things that are happening. So we're basically doing some filtering out to get down to just the views that we want, so no sheets, no legends, no schedules, and no view templates. Just a little bit of manipulation on the data.

Now we start getting a little bit more into the weeds here. So we've got a couple of things happening here. First of all for those views, we need to pull out a lot of data. So remember all the columns in the Excel file? We had a column for sheet number. We had a column for view name. We had a column where we made that unique identifier. All this is doing is doing all of that.

So up here, we're basically creating a list of the file name, the same length as all the views that we're getting. Again data structure matching, it's really important, sometimes, to do that especially depending on the nodes that you're using. They may expect lists that are all the same length, really important. And you have to play around with it. Sometimes you don't always know. So you're like, well, do I need to do that? Do I not need to do that? Well, just try it. Don't be scared. Just try it and see. If it doesn't work, then try something different.

So this next one here, this next section here, what it's doing is it's adding the separator between the file name and the unique ID. And then down here-- so I'm getting the GUID of each view-- and then down here, I'm combining that. So you can see-- so sorry, I should have left that. But if I had rerun this, you would see in that list there that you would see all of the same things we got in the Excel file.

Now down here, we're getting-- so all of that was just to generate that magical column that Justin was talking about. And down here, all we're doing is we're getting all of those things from the view, view name, title on sheet, detail number, sheet number, and sheet name. That's all we're doing here. And if we keep going over, again, I'm going to have to move a little faster. We're starting run out of time. What we're doing is we're combining the sheet number and the view number to a single value. So we don't have two separate columns in the schedule.

Keep moving over, this is where we're building the formula data for the template. So basically I'm taking the path that I parsed earlier, the central file name, making a list of columns and rows basically, and generating the actual Excel link that is linking from the formatted file to the raw data file. So that way, you don't have to manually create the link later. It's all just stuff that Dynamo can do for you.

And then finally, we have three outputs. So the first thing that's happening is it's getting a copy of that formatted file if you have that toggled to true. The second thing that's happening is it's writing to the copy template Excel file all the formula data. And then the third one is writing the raw data that this then linked back into that.

So that is the first script. Like I said, I apologize. We may have a little bit more time to go a little bit more in the weeds with that. But the video that I have does provide all that. So let's go back into the second script and look at this mamma-jamma.

JUSTIN BENJAMIN: Mamma-jamma.

JOSH MOORE: Is than an old school word?

JUSTIN BENJAMIN: I don't know.

JOSH MOORE: I think so.

JUSTIN BENJAMIN: It's Good

JOSH MOORE: So this one is even more involved. But it's broken out into bigger sections. So this is what I was

talking about. You can step yourself up. So wherever you feel like you are on your Dynamo journey, maybe you start with that first script and learn there. When you're a little bit more advanced, maybe you try the second script, or you want to work with geometry. You really think you're getting the hang of it? Maybe take a look at this one and see how some things are working.

Now I will also say that some of these scripts, I tried to optimize. Because when you're learning Dynamo, you will start out doing all kinds of wacky things just to make stuff work that aren't very efficient. That was totally me. And I've got people telling me that are much smarter than me, hey, you really don't need all those nodes. You just do this. And you got one node that replaces 10.

So it's a very common thing to happen. So I've tried to optimize most of these scripts. But certainly there may be even some of you in this room that are like, oh, this is easy, no problem. Maybe give us some tips on how we can optimize this. We'd love to hear back from you.

So overall we have two things that are happening here. We've got this data stream and this data stream. What we're doing is we're doing two things. The first thing we need to do at the top is all we're doing is we're getting all of those dummy tags that your staff have placed into the model. We got to find those things, because we want to set the value for those things eventually.

Then down here, what we're doing is we're basically just doing some things that's going to eventually read the Excel file. So if I scroll in here, this is where I'm getting that whole directory of Excel files. And there's that node Excel.readfromfile. And all I'm doing is doing some list manipulation to get the data that I want pretty much all the way down to the end. So all the way down, when I say end, all the way down to the end of this section.

And at the end of it, all I'm doing is separating that list into the important things that I need to read. So we had six columns in the Excel file. Three or four of those are only for your staff. Dynamo doesn't need those. Hypothetically, we could have one Excel file with just the GUIDs. Dynamo would be perfectly happy with that.

The problem is, you need your staff to be able to know what that GUID represents. So that's why we have those other columns. That's really for your staff. Dynamo doesn't need all of that. We just need a couple of those. So that's what I'm doing, really, in here is I'm just getting the

columns Dynamo cares about.

Now up here again, you can see I'm getting all the detail items in the project, filtering that down to just the ones I care about that contain this word. So if you go in, and you take my family and you rename it for your office, you might need to change that value right there. If you don't, this won't work. So just keep that in mind.

If we keep going over, you can see that basically, we're getting parameter values from that guy. So that way we can reuse it down the line. Now I'm going to keep moving. Right here there's a little bit of magic happening that's basically saying, does the value in the family type equal the value in Excel file anywhere? And if it does, I'm going to take that data and do something with it right. And if it doesn't, I'm just going to ignore what's missed. And that's really what's happening here in a nutshell.

And then we keep moving over. We're almost there, guys. This is really what's happening in Revit. So I'm basically taking the matches. And I'm saying, OK, push the view number into that dummy tag. Push the sheet number into the dummy tag. And that's what's happening there in orange.

And then lastly, this big chunk of stuff at the end is really just for making that refresh report. And all it's doing, again, is just getting the data that I need for the refresh report and writing it out to Excel. So without stepping through that again, because I want to make sure we get to some of this other stuff-- we're almost out of time-- that's really what's happening there.

JUSTIN BENJAMIN: I forgot. Is there a handout with all of these nodes blown up and elaborated upon?

JOSH MOORE: There is, yes. There is.

JUSTIN BENJAMIN: Is there also a video that walks them through it too?

JOSH MOORE: Yes. That's in the data set.

JUSTIN BENJAMIN: I say Chris, you say must. Chris-- or holiday. I'm sorry, holiday, holiday, that's what I'm going to say. I say holi, you say day. Holi--

[INTERPOSING VOICES]

JOSH MOORE: So we've got just a few more minutes guys, about 10 minutes. We're going to rip through these tips and tricks. There are more inside of the handout. Imagine that. so we've got--

JUSTIN BENJAMIN: This gift keeps on giving.

JOSH MOORE: We cut some out, because we had to. But there are more inside the handout. So please look at the handout. So for those of you that are a little newer to Dynamo, this is super important. When you're first getting into Dynamo, and you're trying to figure out what the heck is going on. Why isn't it doing what I'm doing?

Basic Dynamo 101, watch out for red nodes. If you're doing a good job commenting your scripts, you should be able to figure out that you're missing a package. What the red node means is you are missing a package on the install that you need to run this script.

The yellow nodes represent things that are giving you warnings or errors. So if you run a script, that doesn't always mean that it's not going to work. Actually Dynamo, in version 1.2 and the more recent versions is doing better about just skipping over certain things that are null values, and just being like, that's OK. I'll just run the rest of it. Sometimes that doesn't work though. And sometimes you've got to deal with those null values. And we didn't really go into how to deal with those. But that's in the handout too. So be on the lookout for that stuff there.

JUSTIN BENJAMIN: This is seemingly an obvious one, but definitely something you have to consider. The defaults in Dynamo is to be automatically run as you're creating the script. So you don't really want that to be the case if, in fact, you're flexing and learning, especially in Dynamo. So it would behoove you to go in and move it to manual.

And not just without reason, as you're learning Dynamo, you don't want automatically be acting on your model. But also we've done trainings in the past where we'll try to have everybody working and playing along. And they'll forget to move this to manual. And then the processing power of some of these scripts, as you can imagine, with that topo mapping, things like that, sometimes takes a little while. So now they're holding, waiting, not necessarily on a flawed script, but because they didn't say manual and say run later. So that would definitely help you out.

JOSH MOORE: So this is one big thing that I-- it was such a challenge at first. So if Autodesk is listening, please fix this. At least in my opinion, this has been a problem. So when you're testing out your

scripts, and a natural workflow is to write part of the script, hit Run, and see if you get in the data that you want, and then keep going.

Well if you're interacting with Revit, and you do some of that work that's actually doing something in Revit, a lot of times, at least what I do is I go back into Revit, and I undo that Revit transaction. I go back into Dynamo. And I keep going for an hour. I hit Run. And I'm like, nothing's happening. It's not working. And I can't tell you, when I first started, how many hours I wasted trying to figure out what was wrong with my script, when all I needed to do is close the script and reopen it.

Super simple, basic thing that you think in your head, wow that's so easy. But it's always not always obvious. So this is my suggested work flow. You run the script. You undo the Dynamo transaction in Revit. Close the script and reopen. That's like 10 to 15 seconds of your life. You're not even closing Dynamo. And I promise you, it will save you time. do that. It will help you a lot.

JUSTIN BENJAMIN: So no commands active in Revit, another seemingly simple one, but definitely be aware of this. You cannot have the door command activated. You can't have the wall command activated. You can't even have things really selected when you're trying to use Dynamo. It would be best if everything's not selected. No commands are active. Couple hits of the Escape button. Make sure that whether you're in that environment. Otherwise the script will hold, could hang up. You could get really frustrated.

JOSH MOORE: Yeah, so this is another one of those things, where literally I've spent hours trying to figure out why my script's not running, just to realize that I had the door command active in Revit the whole time. And I'm not kidding you, it just sits there. It acts like it's running. It will say run started. But nothing's happening. So make sure you just, before you run, if you're second guessing, just Escape, Escape in Revit. Hop back over. Hit Run.

AUDIENCE: [INAUDIBLE]?

JOSH MOORE: What's that?

AUDIENCE: You can't use escape in [INAUDIBLE]. You have to Control-Z, Control-Z now. Cancel everything all together. [INAUDIBLE].

JOSH MOORE: So pay attention to Active View in Revit. So this is another one, where it's really important that you, especially, are aware of when you're running scripts or giving scripts to people to reuse

them. That's really what we're after here. A lot of you guys are probably the BIM lead in your office or want to get into this and start writing scripts and share them with other people. And that's really what we're talking about when we're talking about everyday Dynamo, being able to get in and just reuse simple routines.

It's important to know that if there's a script that you write that's reliant on the active view that's open in Revit, that you make that known somehow. And also that if you're running a script, that you're looking for that and making sure that the view is active.

Another important thing that I would mention is that I've had some weird issues when I try to have multiple Revit files open in the same session of Revit with Dynamo open. Sometimes even though it is supposed to read the file that's active, it gets confused, it seems like to me. So I would just encourage you. You may be able to get that to work. But at least in my experience, I had more success if I need multiple Revit files, just open multiple Revit sessions.

So this is a big one. And we're going to shout out in just a minute. But I have to say, we love what all the third party developers are doing. If you notice, a lot of the final transactions in the scripts I've written are absolutely relying on what a lot of people are doing in the Dynamo community.

But there can be issues, especially with re-usability. So we're going to talk about shared libraries in a minute, which would really help. But issues can occur when our friends update their packages. Just a basic example, when the Clockwork package got updated with the Run Me input, all my scripts wouldn't work until I went and fixed them. And these are all scripts that my staff are using. So just keep that in mind.

Also some nodes are just shortcuts. So if it's just a shortcut of stuff you can do out of the box, for re-usability purposes, I'd highly encourage you just use the out-of-the-box stuff, even if it takes you a couple more nodes. This is a visual scripting for architects. So the less corners we can cut, the more you're going to encourage your staff to understand what you've written.

I'm not saying there's not a place for Python and design script and code blocks and all that stuff. It's all good stuff. But the more that you can actually show them visually with the nodes what's happening, the more they're going to understand this. And the more adoption we're going to get. And the less push back you're going to get from people in your office.

And then I said the Python thing. But one thing, when you're testing somebody else's node

that they've written, one thing you can do is take their Python script, especially if it's a simple thing where they've just written the Python. And instead of use their node, you can embed that into your script. One benefit to doing that is a lot of times, you'll get better error messages if something is not working. And that can really help you figure out what the heck's going on.

JUSTIN BENJAMIN: Josh, we have three minutes.

JOSH MOORE: We're almost done, I think. So this one is version control. Biggest thing with version control is just making sure that you're keeping the Dynamo version as a part of this. And then I'm going to kind of fly through this.

Biggest thing with this is shared location. So making sure that you are using shared locations, this is covered in the handout. This is super important. I would highly suggest you read that section in the handout. We're going to try to fly through this. And then making sure that you're commenting your scripts, color coding things like that.

So a shout out real fast.

JUSTIN BENJAMIN: Do we want to do it?

JOSH MOORE: Yeah, let's do it real fast.

JUSTIN BENJAMIN: So just real quick, these are the top five, essentially, people that have created content inside of Dynamo that we've been able to utilize a lot, and not only us, but rather, of course, the world. So I always like to say, there's a couple of people that were registered in this class that, in fact, helped create some of these really super popular ones. So awkwardly take a look at your neighbor left and right. And if you see them, go and give a little bit of a pat on the back, little shake of the hand.

JOSH MOORE: We're going to fly through these.

JUSTIN BENJAMIN: Dimitar Venkov, super duper, 8,000 [INAUDIBLE], spring nodes. What other spring [INAUDIBLE] spring. That's all I have.

JOSH MOORE: So the most used node there is basically what we use the most. So in this part of what we use everyday Dynamo for.

JUSTIN BENJAMIN: Julian Beno-- I can't do it. Julian, again my French accent comes out. I do not know. Tool.eraser, steam nodes, rather, super duper. Give him a pat on the back back, high five. Nathan Miller, thank you for your name, super simple to pronounce. So lunchbox, I think a lot of us have heard about it, have used it. Hopefully, if you know Dynamo, you know Lunchbox.

Archi-lab.net, Konrad Sobon, all I want to say is dude, little bit more of a smile. Otherwise than that, thank you super duper much, Get All Views. I'm loving him, though. I'm loving him. I don't want to do this name at all. Andreas super duper, love that last name.

JOSH MOORE: So y'all give a round of applause for these guys. If you're in here, thank you so much. So last thing, we're launching a web site, Dynamojourney.com. Also you can email us at dynamojourney@gmail.com. Go Check this out. There's a behind the scenes video of us rehearsing this. And it's, needless to say, kind of comical. So check it out. It's on Dynamojourney.com

Please fill out the feedback. If you want us to come back next year, we'd love to come back. Talk more of Dynamo, maybe some Revit or whatever. We'd love to hear your feedback.

JUSTIN BENJAMIN: Do you guys think I could talk faster?

JOSH MOORE: So thank you very much everybody. Thank you.

[APPLAUSE]