JASON HOWDEN: So thanks for coming to my class, Dynamo for Beginners. Hands up who's used Dynamo? Can you hear me down here?

AUDIENCE: We can't hear you.

JASON HOWDEN: That better? Is that better? Cool. So there were a few-- I don't know how that works. They just said on off. I can speak up. So there were a few hands that used Dynamo. So it's good to see that everyone put their hands up, so that's good. So, Dynamo for Beginners, the idea behind this class was to try and break it down to a very simple level and just use out of the box nodes and code that experts put together. I'm not a expert. I'm not like [? Archello ?] or doing really amazing stuff. I'm more of a hacker realist on the shop floor trying to get projects done more efficiently. So, how can we automate with stuff?

And especially with Dynamo we can automate all sorts of stuff. Geometry, you may have been to classes throughout the week where you were creating geometry, creating fantastical stuff, but I find there's a lot of power in just automating the mundane. There's a lot of mundane stuff that we do in our office in New Zealand. Anyone else from New Zealand? It's just me. So hopefully you can understand me. I'll speak a lot of Kiwi today. So a lot of the mundane stuff that I'm talking about, things like renumbering drawings. Maybe some designers come up to you and said the facade just looks boring. Make it more random or something like that, and you come and going what am I going to do with this? Am I going to sit there and swap out curtain panels all day long?

So, trying to explore some of these things and how we can use Dynamo to do this stuff out of the box without having to write Python, or without having to understand C Sharp or any kind of coding stuff. So we'll look at the interface. This is a hands on class, so I'll probably got back there for a bit and we'll poke around the interface. We'll look at this and look at that. I'll cover some of the key places where I find knowledge about Dynamo. There's a lot on Google. Google is my friend, hopefully it's your friend. But surfing around Google, particularly now, there's a lot of hits for Dynamo. I remember when I first started exploring Dynamo it was very flaky and there was like 10 hits on Google. And it was kind of like what do I do with this? Lot of experimentation. Now there are people that have dedicated blogs to this and they spend their weekends blogging about it. So you can learn about it. So that's fantastic.

So bit of an overview with Dynamo. What is this thing called Dynamo that everyone's talking about? It's not him. I had to put-- it's Vegas right? So I had to put a cheesy slide in, that's Vegas. He's magical. I think Dynamo's magical. It's a application for visual programming. So you take nodes and stuff, you connect nodes together, and nodes are like little mini programs and each little node will do a little thing, and you add them all together and they do a big thing. And you connect these, I guess programs, with what we call wires. That's where the string theory comes from. You'll see in some of the examples that you end up with these wires going everywhere and it looks like spaghetti junction, or a bowl of spaghetti.

There's a wide range of applications. I kind of touched on it. From geometry through to renaming your drawing sheets. Think of it as your assistant that doesn't complain about working overtime. It'll just keep doing stuff all day long if you want it to. These are my favorite websites to get you started. DynamoPrimer, really cool. A friend of mine runs DynameNodes.com. Again, he works in the industry, so his blog is very industry focused. He's Aussie so don't kind of hit him up for that. There are some Aussies in the room. I saw you. It's a small world down there. Archi Lab, he actually writes a lot of code and nodes as well. We're going to use some of them today. Big thanks to Conrad for that. DynamoPackages, DynamoBIM, and of course Google.

So the interface, that's what it looks like. It's kind of five features to it. At the top, menus just like Revit and every other application's menus, menus are at the top. Like how the applications we use, the toolbars, and their icons and do things like save and open. The library is where you'll spend all your day in Dynamo. Kind of in the library looking for little nodes and programs that do things. It's searchable. It's just not as good as Google, but the search at the top, you start typing in something and it'll start filtering the list of the library. So if you want to extract an element parameter you just type in things like element parameter and all the nodes will pop up that relate to that. And then you drag them out and you give them a try. A lot of trial and error and you get there.

The workspace, the bit in the middle is where you kind of create your code, your script. Some people call them graphs. It's just a jumble of stuff is what I call it, in that you want to keep that space clean and somewhat organized. We'll talk a bit more about that in a bit. And then coming down the bottom is where it all happens. There's a button here called Run and it'll run in a manual mode, and an automatic mode. When it's in automatic mode, soon as you make a change, poof it's done. And Revit, when you're starting out switch it to manual. You do want to be in manual, because you might do something, and poof, your files gone.

So you want to watch that one. So, what are these nodes that I keep talking about? This is one of them. That's a node. That one does points by coordinates. XYZ. So we just put some inputs into that, and then it'll make a point at that location. So hopefully that's not too tricky. So at the top a name. The body of the node. The inputs and the outputs. And the thing I use a lot. That one. I Like that one because you can see the program's working. And if you're trying to extract information, and you're not sure what's going wrong and you're not getting the results, you can go back there and kind of watch what it's doing in real time and you can be like ah, it's erroring on a panel let's missing.

So you can adjust the nodes and the code to suit by understanding and watching these checkpoints as you go through. The wires. That's the little curvy things that kind of connect the input, or the output to an input, and you just kind of in and out, in and out all the way across. Now this is where you need to start to build your code with a flow. You'll know what you've done, but the next person will have no idea what you did before. So if you don't leave some bread crumbs behind, and one of the most effective ones is just to work from left to right, and you're just flowing the information constantly like that, that helps people to follow what you're doing, and kind of understand where you came from. If you start kind of putting a thing over there because you didn't want a pen, you then start looping back on yourself and when you get a lot of wires crossing over you just can't trace to where you got back to.

So just think of them as a continuous stream. The library. So the library has quite a number of parts to it. This is your code library, think of it like that. These are where all the nodes are. At the top we talked about the search bar, and then it kind of groups into these packages. And some of the packages have logical names, some of the packages have illogical names. One of the packages we're going to use is Bumblebee. Have a guess about bumblebee means? Thought so. It's a tool to interact with Excel. So it's not named like, Interact With Excel, that might be useful, it's Bumblebee. And you'll see lots of stuff like this. Like Mandrill, Shrimp, a whole bunch of really funky names. And they make really good talking points at the after functions when you come to events like this.

Inside the packages they'll be grouped into categories and subcategories. So just like an Revit, you go to railings and you find all the bits about that. So when you look at your package you'll find all the bits about that. So under the Revit package there's some things to do with elements, there's some things to do with parameters, and when you go down inside them there's things like, under elements, there's things like name and then it's like material name and you'll find a whole bunch of packages that relate specifically around those elements and those packages.

And right at the bottom of this library we can search for external packages. So there's a bunch of really cool experts out there that are writing code for Dynamo to help us all do our jobs better, and they're publishing them up there for free. It's fantastic. Like free stuff that you can just download. You can search for it, takes a little minute to search as lot of stuff going up there, And then you can just get busy and get using it. Really neat. These are all the little building blocks inside. So it's a further look inside the building blocks. So you can see we've got points, we've got geometry curves. All sorts of stuff in there. Keeping it clean.

So I talked about lacing the wires, or having that flow with the wires. Something else that you want to do there is to group the nodes and start aligning them. Group them in kind of discrete little packages, like these four things do something, and group them and put a name on that group saying it does something, or it does exactly this. That will help people understand what you're doing. There's also a right-click function to help you align the nodes. I'm a little bit OCD and I like to line them all up. I guess that's old school drafting for me. These groups, so at the top we can put a, I guess a title. And get creative with these. You'll see some of the nodes and packages you download online will have really clear, detailed descriptions.

People spend a lot of time informing you about what these do. And by doing this, it will make it more effective with your code. We have an open code kind a team inside our company where we try and have a, yes, a Dynamo user group and we get together and we have pizza and beer and we talk about the stuff. But by sharing the code around your office, and you'll find someone will pick up your four things, add another thing to it, and do something you didn't think it would do. All right, going beyond what's in the box. This is extending the Dynamo way beyond. So this is this realm of these Dynamo experts. There's a small group called the Bad Monkeys have you heard of the Bad Monkeys? Yes, so you want to follow those guys. They're doing amazing stuff.

And some of the stuff you'll see me use in this class has come as a byproduct of the stuff that they're doing. And one of them is Bumblebee being able to access Excel, and use Excel as a database of information to create stuff in Rivet. It's really neat. The example we're going to do today is make work sets from a template. I don't know if you guys have trouble getting your stuff to get the work set naming conventions applied on every project but I do, and this is solving that. Because I can have the Excel spreadsheets that are on my network, they click this button, and it's all done for them. Clockwork, so see they have these really funny names. So Clockwork is a collection of stuff. There's an amazing amount of nodes and code inside this package called Clockwork. Make sure you install that one. Don't leave home without it. It's got heaps of stuff on it.

Lunchbox. Who brought there lunchbox today? It's got over 8,000 users apparently. That's what it said last time I talked about Lunchbox. I think it was 3,000, so it's just growing and growing and growing. And the types of things that you can do in these codes or nodes for stripping information, reformatting information. Deeper I guess interrogation of the Revit database. Starting to get into some of the API programming within Revit without having to learn C Sharp. So you can pull out a node that says extract element parameters for, I don't know, curtain walls or something like, that that's the type of stuff that we're doing in Dynamo by leveraging these packages.

Dynaworls, this will help you automate your Navisworks workflow. Really handy if you're doing all those class reports on a weekly basis. You've updated your NWCs, you may automate that stuff, and then you're gonna go in there and run all your class reports. By using Dynaworks you can start to automate that stuff as well. So from Revit you can fire up Navisworks, run your class report, and then when you go into Navisworks it's all done. Mandrill I don't have an image of this one, sorry. Mandrill is a graphical interface for looking at information. Big spreadsheets of data is fun, but when you give that to the boss his eyes roll back in his head and you lose the conversation. Taking Mandrill you can take that information and you can turn it into a pie chart and you give that to the boss and he goes, wow, I forgot all this stuff in this corner of the building. That's a fantastic thing for Dynamo.

It's an early stages of development, so if you do play with this, it is a bit flaky. In fact, a lot of things in Dynamo are a little bit flaky, and I'll talk about them. But we're using Mandrill in the office to do a model audit reports. There's some nodes that we've got to look for the number of warnings, error warnings in model files. You tell the BIM guys to clear out their warnings. Do they ever clear them out? No. So now we're reporting their warnings and saying who's got the most warnings in their file? Oh look, you do. And we're encouraging them to clear them out. And you can use it for all sorts of stuff. And a simple dashboard on an A4 page, or on a screen clip into an email to the project manager for that team, works wonders. The files get clean.

Word from the Wise. One of my examples is now broken. I upgraded to 1.2. All the examples that are up in the data set worked fine in 1.0. The last one doesn't work fine in 1.2, and the fourth one I had to call in a favor from one of my mates at the Bad Monkeys to rewrite a node so it would work in 1.2. So I'll upload some new data sets. So if you download 1.2 and want to play with them, they work. And I'll also update the handout as well where they're broken, at the little computer bug. Dynamo is in a heavy phase of development. It's open sourced. That means anyone and everyone's having a go at creating stuff. Some of those people are, I would say, full time kind of coders. They know what they're doing. Other people are like me and they're just kind of hacking some stuff together, and we didn't really know what we were doing but it worked. So we chucked it out there. And then it breaks, and we're kind of like, well it was free, what did you expect?

So back up your stuff and, yeah, just be careful out there when you're getting into it. Because it can be a bit scary if it all just goes horribly wrong, and then you think it's you. It's not you, it's the software, and it's getting better. All right, so I'm going back here and we're going to do some hands on stuff. And we're going to start really simple. The first thing I feel that we should do is kind of select something. So we're just going to walk through selecting a basic element, it's a curtain wall, and I'm going to look at different ways that we can select the element. And they're going to extract some information from their end. The idea behind this is to show you within half a dozen code, nodes. We've made some code to select some elements, and we've looked at different ways of selecting those elements, and extract information from that.

How you apply that in the real world could be extracting information for your interior's team, on the number of chairs for this type, with this color material, and really extend beyond what Revit can do in a basic schedule. And then you can start to leverage it even further than that. And we'll look at randomizing a facade today as well. So this is a snapshot of what we're going to have look at. It's not as scary as it looks. So this first piece here, this is just the selection nodes. And there's three different ways that we're going to look at selecting a node. The first one is just kind of select, and you go to the model environment, you click the wall that you want. So it's kind of got a manual process to it.

The next one we're just going to grab everything that is a wall in the file. And that'll just go and find all the walls. And then this other one is find all the walls of a particular wall type. So there's lots of different ways that you can select and grab stuff. In the middle, we're going to go grab a curtain panel out of the walls. I've chosen curtain walls because you can do so much with curtain walls. I think they're one of the most useful features of Revit because they can do more than just be panes of glass. And then we're going to add some nodes at the far end to extract information out of those curtain panels. And the example I've got there is just a basic material, and a basic area of those. So we'll have a look at that.

Can you still hear me? So, this a little example. These are all in the data set as three curtain walls, and there's about 1,200 curtain panels in here. So Dynamo under 1.2 installs under manage. It's kind of over here at the end now. I don't know why they did that. I guess you can manage Revit with it. Used to live under add-ins, and from time to time it would jump in and out all over the place, but I guess this is where it's living now. I'll talk about Dynamo Player at the end of the session. So I just click on Dynamo, the interface will start up. You get a bit of a dashboard approach to the interface. You can have a new piece of code. You can create a custom node. That's an advanced class. We won't look at that today. And there's some quick links to discussion forums, the web sites, bits and pieces here. And your recent kind of codes and things that you've been working on here.

I'm going to cheat. I'm going to use my examples. Because then I don't have to be searching for things. But to search for things when you've started a piece of code, you just simply click in the search bar, and just start typing. So if you wanted to find the name of something you just start typing name, and it will pop out. Now something that you've got to really watch out for when you're learning Dynamo is you got to read these bits down on the bottom here. See this is wall type Revit. Material Revit. So that's for a material name. You can see there's like, I don't know, 50 names of things. Not always is the name node that you drag out the nerd that you want, unless you read the little bit below. Inside the screen itself, you see I've got it set to manual, that's just a drop down. Got the Run button. And up here on the side we've got a zoom interface. So if you select a node just click on it, it will go blue. This kind of zoom to fit is brilliant because it will zoom you in to where you want to be.

Click away. You can use it like a zoom all feature. You've got a pan feature here. Mouse wheel in and out. And up the top here when we start to make some 3-D geometry we can visualize the 3-D elements that we're working on as well. So when we start to make things, and they have a main three dimensional context to them, we can start to look at that. So as you're building the Dynamo in a geometry sense, you can see a bridge unfolding in Dynamo, then when you hit run it's kind of in Revit as solids as well. So that's really neat. So you can see what you're doing. And you can switch between the graph interface and the 3-D interface there. All right, so, you can run, if you have two screens you can run it on two screens, away you go. And basically if I select on that select element node, I can go and select on the wall. And I've got my element ID.

So I've selected our wall. And I think that's kind of like writing your first piece of code where you go hello world. We've done something. With that element ID I can then pass that across. So that element ID is then kind of the output to the element, and from that element I can pass that to another node and the thing that I can pass it to is to a node that's going to extract curtain panels. So it's saying curtain panels by elements. I've already got my element. Pass it over to there, and if I run after I've connected that, it'll update the next piece of code. So the next piece of code will fire, and I've found 380 panels in that piece of curtain wall. And that might be enough for what you want to do. You might want to just count the number of panels in this wall. That's 380.

It's better than one, two, three, four. Another way of doing it is to use a node that looks for categories. And a node that looks for categories does actually look for all the categories in Rivet. And you can use it for finding stairs, doors, floors, walls, everything. Because I've hit run, it's found all the walls in here, and if I was to take that and pass it through to the element, elements by category it'll then find the elements of that category. So if I connect that one up to that one, it'll find all the walls. And then I can pass that one up to that one, and run again, and it starts to get all the panels of all the walls in the file. So I've got three walls. It's found the three elements, and now I've got 1,200 panels.

Another way of finding this stuff, because we're dealing with walls and things that have types, is we can use say a wall types node. And to find this kind of stuff, you just walls, or wall, and you'll find all the nodes that relate to that. And you can see the icons are starting to change as well. The built in icons start to have pictures these days that relate to the, I guess the application that they're running. The wall nodes look like little walls, and the Clockwork ones, they all have these like little clockwork cogs. So some of the developers are getting quite creative with their branding, and running that right through, it really helps. So I can select all the wall types by choosing the drop-down. And it'll list all the wall types that are in this file.

If I pick my curtain wall type, and pass that through to node, this particular one as a custom node. I think this one's from Clockwork. It's going to find all the elements of family type universal. So if we start to look at things around, say families, and we go by type, see that there's all sorts of different nodes that are very functional, that will work for stair types or ceiling types or roof types. So using some of these third party applications or packages like

Rhythm, Clockwork, Lunchbox, you start to expand the basic functions out of the box, and you can get a node that is universal. So it doesn't matter what you're sending into it, that one node can be used for it. And that really does help.

This particular node has a toggle with it, and the toggle is going to accept a boolean function. So that bowl is for a boolean, and a boolean is on off. True false. And by default when I hover over there it's saying it's true. So it's going to run. So sometimes you may want to sit this note not to run as part of your overall package of code, and sometimes you may want to run it specifically. And then I can pass this one, the elements of this back up to my panels. Click run, and I'll get the same because the three rules are of the same type in this file. So that's kind of cool, but what do we do with it? That's probably the question. One of the things we can do with it is find the material type of each of those panels.

And we can use a node. One of my favorites is another one from one of those packages, is to get the parameter value by name, type or instance. A lot of the other nodes within Dynamo out of the box will either be for type, or for instance, so then you might miss some of the information. So it's really important to look a wee bit deeper than straight at first. And then I can pass the curtain panels up to the element. And then it's looking for a parameter name. Now one of the tricky little things with Dynamo that was a new feature a few versions back, was a code block, and a code block is, I guess, a generic input. It can handle code. So if you know how to write Python and stuff like that, you can copy paste Python directly into this code block and run a program. Create it right there, right then. Or it can be just a simple input, data input type.

And one of the simple data input types that you can do with it. So to get a code block, double click on the blank screen. The code block will pop up by default. And one of the things that you can do is you could put like, 12 in there, and there'll be the number 12. Whereas previously you had to use a number node, and then you put an input into a number node to turn it into. So this code block is basically removing the need to have two or three little nodes to do one function. And to write text to look for something that's like a string, you put the quotes in there, and you might look for something that was say glass. If there was a parameter called glass that would find that. So there's lots of little things you can do with these code blocks.

We got one here, material. If I can just connect that with that parameter name. And then run that, it will start to produce a list of the materials. And it gives all the IDs for those. Then what I got to do to get the information out in plain English, is I want to look for its name. So one of the nodes I can use is element name. And that will return a plain English string. And I can connect the element IDs of the materials, which were the numbers present there that are highlighted green, that's the material ID number, into that element. Click run. And that should give me all their names. Clear glass, glazing. All the way down. Now one of the things you can do with this little fly out, when you hover over the there and you scroll down, you get the pen icon. Pin there and the list will stay up. That's really neat. And a new feature in Dynamo 1.2 is this list levels.

So the list levels, level one, that's the lowest level. Level two, and depending on how nested your list, is you'll have level three, level four, and you can extract this information at a list level now. Previously you had to kind of get element at level, and you had to write several nodes to get to that level, and then several more nodes actually extract that information. So the guys are trying to make Dynamo easier and easier and easier to understand at a level that we function at. I can take the same node, connect it over to an element here. Connect the area to here, and start to extract the area values now. So I can see all the areas. And some of the other things that we can do, is we can start round the areas. If we needed to around the areas we can put the number in here. We can set a precision, and we could do some rounding on those. And it might complain.

So Dynamo I will give you a complaint message like this. The node will go yellow, and then the little warning box, it'll say hey, I haven't been able to do something. So that's what that means when you have something go wrong. It'll flag up yellow, and the little flag will come up and say hey, you asked me to convert something, but I can't convert that. There's something in that list that it couldn't convert. And it's probably that one of the areas is zeroed out, and it can't round the zero. Because if I look at the list, you'll see that it's rounded some of the items. Now if I scroll down far enough, I bet there's one in there that's zero. It doesn't necessarily mean that the code is broken. It's just giving you a warning. That's kind of our first step.

So, working with lists. One of the things with Dynamo that I think is often, misunderstood is probably how I'd put it, is people think it's really complicated. I don't think it's that complicated when you understand that Dynamo is just managing a list of information. You saw each of those nodes was returning a big long list of stuff. Essentially that's all we're doing with Dynamo is we're managing a list of information. And that list of information could be points on geometry, it could be parameters about geometry, it could be information that is contained in parameters. And if you want to sort that, all you we need to do is sort the list. If you want to pull stuff out and move it around, you just want to pull it out and move it around in a list context. So understanding how lists work is, I think, the key to getting the most out of Dynamo.

And I've got a good little example here that we'll look at, where we build a simple list to make a circle, and extrude that up to make a cylinder, and all those points are contained in a list. And then what we'll do is we'll look at what happens if we reverse the list, and if we stagger the list, and how that interacts with the geometry. Now I've played this little example in another location in the world, and they thought this example was probably the best way of describing what's going on. Because a lot of people were looking for the twist in my geometry function, but you're really wanting to manage the list of information when you're dealing with Dynamo. From where I sit and understanding it all. So we're going to start with making an origin point, then we're going to put a circle on there. Then we're going to copy the circle m and we're going to create a cylinder.

So something's different this time. I've gone into the family editor environment this time. So I'm going to make some geometry. And the easiest why I'm going to make it this way is just in the conceptual mass family editor environment. I'm going to make some dots and the dots are going to appear as a little cylinder. If I zoom in far enough we'll see it as a cylinder. It's going to be really tiny. Put that there. All right, so first things first, we're going to start with a node that's going to place our circle somewhere. And we're just going to place it at 0,0 right in the middle of the universe. And to do that, I'm just going to create code blocks. So double click, create a code block. I'm going to put zero in there. I'm going to take the zero and put it in the X, the Y, the Z. So I'll put it at the center of the universe.

And if I run that, there's a little dot. See the little dot? That's all it's done. Now this point I'm going to then use as the origin for a plane. So I'm going to take that point and I'm going to parse that into the origin of the plane. Because I want this to be the center of the plane that I'm going to draw a circle on. And we won't to see a plane, so there's nothing to really run at this point. And then I'm going to draw a circle on that plane. So I've got a node, circle by plane, and a radius. So I need to input their information. So I've got my plane. I clicked the plane to the plane. And I want a radius. And I'm going to place a radius of 50 units. So it's going to be tiny. And then if I click run, it's made a circle. Now there's some other nifty stuff that we can do with their geometry. One of the things that we can do with their geometry is translate, or move my geometry.

So I'm going to take the geometry that I've just created, and kind of copy that into a different spatial context with this node. So I'm going to write my circle, because that's the geometry. I'm going to put it into the geometry here. And if I want to make a cylinder, I want to move it up in the Z direction. So another simple node block, double click. I'm going to move it up 100 units in the Z direction. And if I click run again, I should get another circle 100 units up above it. Starting to make some sense? Right, so the next thing, to turn this into a cylinder and start to do things with it, I want to add some points to the cylinder. And what I can do is I can use a node that will create a point on a curve at a parameter. So I'm going to take my curve, my circle, yep. So this one, that's my circle. I'm going to curve it into three. So that's my curve.

And I'm going to place my points using a piece of design script. So I'm going to place a point between one and zero so it's at one point, and I'm going to place that at intervals of 30, and at a total number of 30 around my circle. And there's different ways that you can format the design script. If you use the little tilde, the little squiggle, that is like approximately at this. So you can use that as a spacing. I want to place my chairs approximately a meter apart, and then depending on whether you select a line, or a floor, or a room boundary or something, it'll go and start placing all the chairs roughly a meter apart in that space or zone. So that's how you can use the design script to start to place things. So if I connect into my parameter as the rules to place a point, one point, and 30 of them around that circle and hit run, I should start to see some points. They'll be small. It'll get better once I've connected them to the lines, we actually see some lines. If I then connect and do the same thing to the top circle, I've then uncreated points around the top circle and the bottom circle.

And I can then start to lace those into some lines to connect the dots. So I'm basically going to draw some lines from the bottom circle to the top circle all the way around. I need this one. It's all right. That one goes into there. Bottom circle, top circle, there we go. So I've got a basic cylinder. And you can see in this Dynamo area now that I'm working with geometry, about the geometry behind, and also in Revit. So one of the things that I can do, and talking about that list management and why we're looking at this one, is what would happen if I just adjust that list of points? So I've got my list, if I look at my list, there's my list of points, for all of those points on that circle. And if I was to do a simple list reverse on there, so I take those points as a list and put it into this node called list reverse, take them out, put that back into there. And then run that. I reverse the lines and I'm starting to make the geometry do different things just by managing that list.

One of the other things that I can do, is there's another node called shift indices. And that basically is taking the point and it's shifting the list by a factor of something. So factor of 10s. So instead of going to the first one, go to the 10th one, or go to the ninth one in my list. I'm going to move my list by a factor of 10. So I'm going to take my original list. And with a code block, double click code block, into there, that's the amount. And then take that list, and put it into the my start and hit run. I've now twisted my object. So just by managing the list, I'm creating different forms. And I think this is the easiest way to describe how the lists actually run in Dynamo. And how you can use them to control things. So far so good? Too fast? No?

So onto the curtain wall randomiser. So this comes from a real world scenario. I came back into the office on a Friday night, we have drinks on a Friday night, do you guys have drinks on Friday nights?

AUDIENCE: No

PRESENTER: No? Come to New Zealand. We're looking for Revit people. We got drinks on Friday night. The catch, there is a catch, we have earthquakes a lot. They're big, and they kill people. But we have drinks on Friday night. And we need to rebuild everything that has been destroyed, and we're rebuilding it all in Revit first. And then for real life I need to have it fall down again.

AUDIENCE: Job security right there. [INAUDIBLE]

PRESENTER: Oh, just the lines. Yes, so it was just lines. You could then go on and connect it to solid geometry and keep going with that example. So think of analytical lines for structural analysis, I could then go and connect those points to structural columns or beams, and it would do beams that were twisting around just by picking out the start and end point of those nodes. And all I'd need to do is just grab the family and say start point, endpoint on the same thing and it would wrap them around. So on a Friday night came into the office, there was an architect sitting there, he was having a problem with his facade. We do a lot of curtain wall facades in New Zealand, and we like to make little curtain wall panels that look like ceramic tiles, and they have different shapes and stuff and then mix them up and jumble them around with clear glass and bits and pieces. And the issue he was having is he only wanted to have 20% of this type randomly displaced, and 30% of that type, because he liked that one, other places, and then he wanted the rest to be all clear glass. And he didn't really care for what pattern it made, he just wanted to see options. And I was like, oh, that'd be easy. And after a few more beers it was kind of easy. So this is the basic premise of this one. So we're going to select a wall, we're going to interrogate the panels, and then we're going to look at the panel types, and then we're going to change them, so we can explore how we can change all the panel types across the wall. And then we're going to take this piece of code, and we're going to extend it a wee bit further. So this is how far he got, and then I came along to solve his randomization problem. And we added a number slider. The number slider is useful for when you're doing randomized code, because you need a seed number to randomize stuff. Nothing in computer speak is really random.

You just change the number it starts at, and it shifts its kind of logic a step. So that's a seed number. And then we kind of pass that off to jumble the facades, and we've built a bunch of code or nodes to select the list after it was randomized. So to get the first 20% to the list, the second 30% and so on and so, off and then apply the panel type he wanted to that list. And then if you repeat that, you can have several different panel types, and you can adjust the percentage of panels that you want to select and make that type, and then by adjusting the slider bar you can just randomize that list so it actually randomizes the whole facade.

So we'll have a look at this one. I think it's quite fun. It's the same one. Yeah, that's talking about how to build the next piece of that one. And I'll talk about that down here when I actually do it. Again, all these examples are in the data sets that you can download. And there's like a complete one, and a start one. So you can kind of just use the end product if you want to randomize your own facade without needing to build it. Or if you want to build it you can build it. So I'm just going to start with these three pieces of curtain wall like we had before when we interrogated them, and we're going to use a lot of the same functions as that very first kind of select elements step. So into Dynamo. Wrong one.

All right, so the first kind of step that I just want to show you is how to select the element and transform all of its panel type to a different type. So take the basic select element thing that we talked about in the first instance, and we're going to pass that over to the curtain wall panels. And we're going to take the curtain wall panels and then pass them over to the elements. Right, let me just find where I put them. Here they go. No, not that one. Further over, sorry. This one here. So you can go and copy paste in Dynamo. Just control C, Control V. I'll put it up here for now. So I'm going to take my elements, and I'm going to, with this elements, I'm going to set a parameter. And the parameter I'm going to set is the type. And then I'm going to use the family types to find all the family types of my curtain wall more panels, and parse that into the value.

So double click for a code block and type. So I can connect that into my parameter name. Got my curtain wall panels going into the element. And now I'm just going to steal this one. Control C, which is by family types. Put that here. And with the family types node it will find all the family types that are in your project file. So if you're dealing with curtain wall panels, make sure that you're selecting the curtain wall panel type that you want, not a railing type. Because it will complain. And I can select say red. And I can push that into the value. And if I select that element, hit run. It's now red. Cool. First step. So I'm just going to unplug those ones for a wee bit. I'll move them out of the way.

Now with that list of information, one of the things that we can do with that list of information is use a node to randomize the order of that list. You will find when you Google and search on the internet there, you can use a whole bunch of complex nodes to create a randomizer within Dynamo and try and randomize all the elements at an elemental level. I kind of figured it was a bit simpler than that. If I just had a list of information, what if I just jumbled the list up? Isn't that doing the same thing? And pretty much that's all it needs to do. Now to jumble a list of information up, we kinda use a slider bar here to see that. And that slider bar could be from one to seven, or it could be one to 900 and whatever. That's 488 for those who are into motor sport, they'll know exactly what that number means. Google it and you'll find out why.

And the ones who know me best in the room know instantly. So I'm going to take my curtain panel. I'm going to poke that data into this randomizer, and if I run this the results that come out the back will be different as I change that slider bar. So I change it over here, I'll get a different order of numbers or elements. See the number at the top there is changing? It's changing the order of that list. And that's all it's doing. And that's creating our randomizer node. Now the tricky part of all of this, came about by needing to create the series of nodes that took 20% of that list. I want 20% of that panel type, spread randomly throughout my wall. So there's a nifty little node called list slice. And all it does is slices a list. And it starts at the start, and stops at the end, and it has a [INAUDIBLE] or a count. So every individual item, or every 10th item I'm going to slice out of this list.

So it's really easy. So I'm going to take my randomized list. And I'm going to poke that into my list slice. And I want my list to start at zero. And now this is where the tricky bit was, the N needed to be 20% of the original list. And I wanted the first kind of 20% of that. So I used a little code block to take two numbers, so take a percentage of our things times 0.2 of the whole list, and I'll get the 20% number I need. So to get the number or the count of all my elements in my list, I can use a node, simple count node. Just count the list of objects. And it will give me a number. So I'm going to grab my curtain panel. And I'm going to put that into that node, And that'll count that list, the length of it, all the objects in there.

I'm going to take that number, and I'm going to put it up here into my little percentage calculator. And I'm going to take my percentage number, 20%. Put that in there. that's going to return a number. So if I run that, that will get something. There, 76. So I want between zero and 76. And if I take that and put that in the end, I've created a percentage selection tool. Just by managing that list. And then I can go and grab my elements over here, and I can take this list of elements, because this list now is a list of all the elements from the original thing so Dynamo understand what you're doing, and I can feed that into this element. Set parameter node. I'm going to set the type. And I can say, I want that to be-- let's make it a color we'll see --yellow. Put the yellow into there. Hit run. And I got 20% in a random order. Easy? Happy? I see smiley faces.

AUDIENCE: [INAUDIBLE]

PRESENTER: I will get the same 20% because I haven't changed the slider. And if I go and change the slider-- because this is a really good question --I'll get another 20% in a different order. So I'll get another 20%, and some of those 20% maybe the same as the first 20%, so I've got now 40% or somewhere between 20% and 40% yellow. So I haven't got a true 20% anymore.

AUDIENCE: [INAUDIBLE]

PRESENTER: Yes this is the beauty. I'll show that in the next step because it more fun when we have completed the next piece. So to solve the next piece of the puzzle, is we need to kind of reset our curtain panel randomizer back to clear. So we want the last remaining 80% to be clear. And then when we adjust it, it's reorganizing the first 20 every time. So to do that, we basically copy and paste all of this stuff again. Which we did to get our first 20%, but instead of ending at a percentage, we end at the total list length. So get the remainder of the list. So it's really easy to do this one. We take our randomized list, and we put it into the slice. Now the slice doesn't start at zero this time, it starts where the last one left off. So we want it to start at the this one, at the 76, and we want that to go down into the start.

And the end is the complete list. Which is this one, the count, that's the end. And we put that into the end. And then we take this list of items, put it into the element, that one into there, that one into there, and I can sit this one to clear, and then click run. Now I got my 20% yellow. And if I go to automatic, and I go find my slider bar, and this is what made this designer like whoa, and then he just sat there all night playing with the slider. You can see as you move it, it just updates. And he just sat there just like whoa. And the beauty about this, is you can go forward and back to previous designs now. Because I've got my list and I'm just jumbling it.

So if he liked number 122, he could write that down and then as he was scrolling along, he got oh, I like 276, and then if you go back to 122, you end up with the same pattern that you had before. And that's simple randomizer. Cool? Lost it. Right, so, one of the other issues that we had, so once we did it with one wall, he then got really excited, he wanted it across all of his walls on all of his building. So to go from the model element here, we went to the wall type. So we wanted to find all of that particular wall type and farm it though. So we use our wall types selection again. So we passed that into our element of types. Hang on. Let me check my notes. Yep, into there. And then I get my elements into my curtain panels. And this is where I've got to do a little bit of a dogleg. I can't just take this list and push it into the end, because this list has now got three elements in it, and I need to get a particular level of those elements.

So I need to flatten my list back down. So I'm going to flatten the list down to a single list of the panels for all three walls. Because if I run it now, I'll get a list, and there'll be panels in it, and then another list and panels in it. So I'm flattening it down. So I can use the list flatten function, and I can take my list, I can put it in there, and then it wants an amount to flatten it by. And I'm just, simple code block, double click into there, so I'm going to flatten it by one element.

AUDIENCE: [INAUDIBLE]

PRESENTER: I could. But I kept it like this so I didn't have to rewrite all the handout. Sorry. But yes, you can use that one. So instead of doing this, you can click on here and use levels. And then from here, I need to grab my flattened list, put it into my count. Because I'm counting a different list, and then take this one and I'm going to feed that over into my randomizer. And then when I run it, it should run across. I need to select the right wall type. This one. Run. And now it's across all the walls. So it's 20% across all the walls. So not necessarily 20% in each wall, it's 20% across all the walls in this one. And you could rehash this script to do individual walls as well if you wanted to.

And to extend that beyond, and to add additional layers, essentially all you have to do is highlight this bunch of nodes. Copy. Paste. Most of it's already all connected up for you. And then change the way the start, the end finishes. So this one here, our original end point is going to feed the start of the second one. And then this end point is going to feed our overall start. And then I can set this to be 1%, and I could make 1% say red. And then if I run again, I should give it a red one. No, I didn't. Red, red.

Yes, there's my mistake. I want the-- I think I'm pretty sure I need to put that one in there for my percentage, to get the right amount of my percent. And then have I got it? No. Live demos. Go back to my notes. Actually, one better step. If I go to the complete one. So this is the one that's in the data set, and I've used little notes and stuff. And I've got little notes that describe the 20%, the 60%, and so on and so forth, then connect it all up for you. So essentially I can go in here and I can say, I want 30% to be this particular panel type. Say red. And this one could be blue. The rest I'm going to make clear. I got up there. I've already got blue, let's say green.

And I can put that one into there. Grab my thing, run. And you start to create some really nifty patterns. And again, depending on the speed of your computer and stuff, if you set it to automatic you can start to just see the patterns augment in real time. Cool? All right. So the next kind of challenge in the office was to get the work sets sorted out. So work sets are a real pain from a number of levels. We do a lot of collaborative working in New Zealand with partners in Australia and all over the world. With those earthquakes, there's an entire city called Christchurch that got demolished. Like literally demolished. 1,000 buildings came down, and we're rebuilding 1,000 buildings. So there's not enough people in New Zealand to actually complete the work, and we're having to work with people all over the place and dealing with that.

And that poses significant challenges with Revit. As you all know, sharing files at great distance is a problem. And contrary to popular belief, Australia is not next door to New Zealand. It is a plane ride, and a long one. So work sets. We have a strategy of trying to get all link files on work sets. All our work sets broken up by discipline, and bits and pieces like that. And our biggest challenge was getting our designers to work enough work flow around that. So we tried encouraging them to look at our startup screen in Revit. And on the startup screen in Revit we listed the work sets so that they were just there and you could find them. That didn't work. People were like, I don't know what to call my work sets. I'm like, it's on the startup screen. Oh, I didn't see it. OK.

So we figured Dynamo may help us with this one. And we had a bit of a think, and we found Bumblebee. Bumblebee came out, and I figured if I put all my work sets in an Excel spreadsheet, put the Excel spreadsheet on the network, and I connect Dynamo to the spreadsheet, I can then hit run, and it makes the work sets and no one else has to think about it. So that's what this one does. So the first piece is basically read the information from Excel, and I use Excel read file basically. And the three parts of it, what is my file? And I use a node to go find the file. And then there's a node, a code block to go and put that into the sheet name, and then there is a node to go and say I want to read their information.

And then I extract some information out of that. So the first piece is to get rid of the header out of the Excel file. At the top it says work set name, and bits and pieces, the description and stuff like that. So I didn't want that as work sets, so I removed that. And then the thing when you're dealing with Excel, it's in rows and columns. That's in kind of like the reverse format that Dynamo needs. So there's a function called list transform, and effectively takes columns and rows and turns them 90 degrees in a simple sense. So we transform the list. So the list we get in Dynamo is effectively taking the rows, and instead of being single sub lists within a list, turns it into one continuous list of information. And then we just plug them into another node to create work sets based on the names that are in there. So I'll show you that, and we'll crack into the next one. So, blank Revit file. Nothing fancy about this file. First things first, gonna make it a work set of files. Going to collaborate on this one. And I've got my two default work sets in here. Now as part of the data set you get, you get the work set template as well.

And they Excel template, there's nothing fancy about it, but I want to explain what's in it. So this Excel template has some category names. And it has the, I guess the Revit category name. So in behind the scenes, Revit doesn't call the furniture category furniture, it calls it OST_furniture. So if you try and create something in Dynamo and you're looking for the furniture category, won't find anything. It needs to be OST_furniture. And then it'll find the furniture category. And then out the other side I've got in this Excel file a list of work set options. Now I'll explain what's going on with this just for information. But on sheet two, I've got our list of possible work set combinations. And in New Zealand we have a thing called MasterSpec, you have a MasterSpec over here, they're completely different.

Our numbering system is not anywhere near as complex as yours. We just have four numbers, yours is like eight numbers long. The So I've taken that specification list and turned them into possible categories for work sets. And then just used some Excel trickery, combined them to get a single number. [INAUDIBLE] I like that one. And then created a little drop down feature so people can kind of drop down and select the little work sets that they want. And, we're getting low on time, but this is the essence behind example number five. So if you know what your work sets need to be, and if you know what all of the subcategories should be on what work set, you can then use Dynamo to grab all the elements on that subcategory and put them on the right work set.

And then all someone has to do at the end of the day is just click run, and like magic it all reformats itself. Caveat, it broke in 1.2. So the out of the box node that used to accept an element by parameter and would take a list of information into it, doesn't want to do that anymore. It only takes a single work set into it. So I'm asking some friends for some help on that one. Why did they break it was the first question. Because it was really useful and now it's not. So that's the Excel spreadsheet. And what we're going to grab as part of this Dynamo example is the subcategory names in example five. But in example four, we're going to grab the work set names.

And you can see there's quite a big list of work set names in this spreadsheet. If you were just doing a simple work set template, you might just have the five or the six work sets that you actually need. But this spreadsheet goes a wee bit further for example five as well. So I'll just close that. Close that, and into Dynamo. So I've cheated again. I've left all the node blocks in there just to try and minimize the amount of time I'd spend searching. But essentially the file path one, if you're wanting to connect to a file, you want to connect to the file path so Dynamo knows what file you're going to grab, and that will allow you to browse to something. So I'm going to browse to that spreadsheet.

This one here. And then once I've got that, I can then connect that to a file from path. So I'm getting the file this time. I've got the path in the first one. If you try and connect this to something, you haven't got the file. You've only got the file path. And then you've got to parse that to a node that's going to say, hey grab that file. I want that. I'm then going to put it into the Excel, or read from Excel node. There's several nodes in this area. So if you just do Excel, you can read write with Excel, you can do all sorts of stuff in Excel now with Dynamo. So if you want to push the random facade panels out to an Excel spreadsheet to give to a contractor to say, go build me these panels. And by the way, these panels are located at row column across my facade, you could do that. And you could write that out to an Excel file.

And you could give them all an ID number as well. So number 51 goes there kind of thing. So I'm gonna parse the file into the read from file. Simple code block there, double click, and I want sheet one. So my sheet name is sheet one, and I'm going to read as string. So I'm going to read it as just plain text. So I'm going to have a boolean function, and I'm going to said that to true because I want to read that as plain text, not numbers. Because Excel can have numbers and text in it. Now I talked about that we want to remove that header row. So at this point we've got a list of information that's starting to build up behind the scenes. I want to remove the first kind of row of information.

So I'm going to take the [? dotted ?] the list, and I'm going to put it into my list remove item at. So it's going to remove an item at a location. And the first item is always zero. So code block for zero. So I'm going to put it there. And then as we're dealing with Excel information, I need to transpose it. So I can go from the-- if I just hit run. So you have got lists and sub-lists of information. So the first list has got a first row, and first column, second column, third column. And I want to organize this so the columns are individual lists. Not each row is a new list. So that's what the list transpose does. And if I connect that, and I just run that one. You'll see that the list is now different. And I've got the first column of information, which was my subcategory names. And then I've got the second list. This one. That's all the Revit subcategories. And then further down, I have list two, which is the third column. And it's my work set names.

Sp I want to grab that list, and again, I could use that, and kinda get the thing at level two, but I'm just going to use the one that I had in the handout. I'm going to get the information from item number two, which was the third column of information. And because my list of work sets keeps repeating all the time, and I only want to get the unique items, I'm going to use the node to get unique items from my list. So I just end up with the work set names. So item, into there. If I run that one. I've got my work set names from my big long list. And if I want to create work sets, there's a node that says create work sets, and I take that list, put it in there. And if I just make that up a smidgen. So I've got those work sets, click run. Now I got those work sets.

Cool? Easy? Cool. We've got five minutes, so I'm going to-- the next example is quite long. The work set manager, I'll talk through it, but I probably will build it from scratch like these other ones. So we can stay on time. So the work set manager would basically took the same principle that we had before, with making the work sets, and took it to another level. I figured that if I got all my work set names so I know where all my subcategories are, and I want this list of information that's in my Excel spreadsheet, I want to automate that process of getting all the model elements on to the right work set. So it kind of looks like this at a conceptual level. So this first place is the same as what we've just done.

And then we add a piece up the top there to get all the work sets by the subcategories. And then down here we do some error checking and cleaning up the model elements per subcategory. And then right out the end we go and say, with all these little model elements, if you're on a particular subcategory then you should be on a work set that relates to that subcategory. So that kind of, we've already seen that. That's what we've just done. So we can cut and paste that from example full. Then we do a transpose. And now we're getting the work set names, ad we're matching that with the subcategories. So back in the Excel spreadsheet I had that list of subcategories, that's why I had that list there. Because we're using the-- watch that one I believe. What is it? Match key values. So we take the Excel spreadsheet, and we're saying for each subcategory, go over and look to see what work set it should be on from that spreadsheet.

And we're matching those up. And that builds me a list of all the work set IDs. And that's what this piece of code or nodes does up here. So that was the one that I got Conrad from Grimshaw in New York to give me a hand with. That broken in 1.2. But he did a favor for me and he fixed it for this class. So, yay. But I couldn't get the last one fixed unfortunately for this class. So that basically goes in and creates a list of the work sets, and their true ID. So again, like the subcategories that have the OST_ on them, work sets, when you want to deal with them in Revit, they have a number. And Revit assigns that number based on the time it was created, times by pi and the square root of something on the other side of the planet. It goes, that's the number for your work set. They are definitely not one, two, three, four. They're like three, nine, two, and 56, and they're all over the place.

So you need this piece of code up here to generate or to fetch the work set IDs, after they've been created. And they'll be different from project to project depending on when they were created. And then we match those IDs using that map the IDs to the names. Now the tricky part in here was getting all the subcategories. We're getting all the elements of the subcategories, and not all subcategories have model elements. So you want to filter your list to be sensible. You don't want to be grabbing all the, say the grid lines or something like that, or annotation labels or things like that. Some of those things have predefined work sets, so you want to filter your list of subcategories to get the model elements. Because that's what we're trying to control.

And we take the index of the category name. So we pass that through to the category name, and we grab the elements of that category name. And then we want to grab the length of the list for the elements by category. So what you'll get in that list is a big long list of numbers. The list number two might be 27. List number 56 might be 3,000 and something. And what that is saying is there are 3,000 elements on that subcategory. So we need to use the list, sub-list lengths to do a little bit of list management. So our work set IDs are just the work sets for the subcategories. So they'll have a lineal number one after the other, but our model elements will vary from subcategory to subcategory, and we need to be able to manage the two lists. So when you feed the parameter name into the element node right at the end, it's looking at the same list of information.

And what I mean by that is the model elements, if there's five model elements, it will be expecting to see the work set name five times at that point in the list. If it doesn't, it will break and it will start putting some of the model elements on other work sets, and then you're back to the way the guys just did it in the team anyway. So to do that, we have to create a list repeating node, and this is what this is doing. So the basic error checking here is to find any subcategories that have no model elements on it. It's highly likely that you'll choose a model element subcategory, like railings, and you'll have no railings in your project. So it'll return zero. No model elements.

And if we want to kind of manage the work sets for that thing, in the way the list is trying to repeat that is going to break at zero. So we need to kind of find all the zeros and add one to it. So this is what this error checking is doing. It's basically doing a little simple test, and it's just saying, hey, if it's greater than zero, don't worry about it. If it's less than zero, add one to it. So I got a list of ones and 10s and 20s. And then I use the list repeat item by length node and that one connects back up into some lists lengths because that's telling us how many times I need to repeat the same names by and it reconnects back up into that one which is the list met the IDs. So I've got my works days to my subcategories of got the number of elements in my subcategories with that list lengths, and I combine it into that one, and it will produce a list that creates a series of work sets based on the number of objects.

And then I feed that into one that we've already used before, which is an element set parameter. And I'm going to set the parameter of a work set, and it works. Or it did in 1.0. But it doesn't work anymore, because the bug came along and killed it . So I'll show you it's sort of working, but not working. And I'll fly over it just by cheating, by going to the end. So you can see. So everyone's seen this house. This house that's got the work sets in it already. And if I go and turn on the work set visibility, everything's all on one work set. And if I fire up that code, and the way that code should work.

It's going to browse for my spreadsheet again. It's all connected up. And we'll come back and look at some of these lists things and lengths and stuff. But down here is where it's supposed to work. This one here feeds a list of information into this value, and if I click run I get a big kind of error message down here, and you can see all the values. They're null. It's not accepting the list of information. And previously it did accept that list of information. This list is in the right format based on the elements it's expecting, but it's not accepting a list anymore. And I've done some testing with it, they've changed the way this element, set parameter thing works, and it's not accepting a list for a value. It will accept a single number. So if I take the work set number for this one here, say 361, and I run that into there it will work, and it will move all the model elements over onto that work set.

I'm not sure which one 361 is. It was probably the previous one. Let's change the color. Say, 357. And hopefully it'll change color in the background to a different one. There you go. So you can see in the background it's changed all the work sets, but all the ones are on the same work set. So I hope to get a resolution for that, and I'll update and I'll email you all the list and saying it's fixed. Apart from that, there was a couple of slides on what's new. What's new, and what I was going to talk about was Dynamo Player. Dynamo Player is a nifty little app for all of those people who are wanting a simple way of just giving people something to just go run. So the concept of this is you build your nodes, you file those in a folder on the server, and then you just tell people to go run that one. And all they have to do is run that in Dynamo Player, and as long as you don't have any kind of tricky kind of selection stuff going on, it should run without error.

I forget which person it was. It might be John. One of the guys out there on the Dynamo community has made a special node so when it works in Dynamo Player, it will pause and allow you to select an object . and then run again. And I believe Autodesk is trying to work on cleaning up the stuff as well. So when you hit run and you have to select an element, it'll prompt you and you go and select it, and then it will continue. So that's a nifty little tool. And I think that's kind of a game changer for Dynamo.

AUDIENCE: There is no data shape package. [INAUDIBLE]

PRESENTER: Yes. Yep. And there's another node out there that's also providing a multi input as well. So if you need to do a lot of multi inputs, it'll pop up a thing for multi inputs as well. So there's a lot of cool stuff there.

AUDIENCE: [INAUDIBLE]

PRESENTER: So Dynamo Player comes in 2017.1, download that for Revit. And Dynamo Player appears. That's where that one comes from.

AUDIENCE: The pop-up?

PRESENTER: The pop up, that's another package. Yeah. It's a package. Yep. Reference. List go up. So there's a list of references. Conrad, Marcello, Adam Sheather, he built DynaWorks. I'll close that so you can see it. Again, this is all in the hand and stuff as well. So they're all there. They're my favorite kind of links, and these guys, they're writing code as well. So this stuff's pretty stable. And they're giving, I guess the Dynamo developers a hard time about improving stability. Stability is kind of a big focus with Dynamo at this point in time. So expect future releases to slow down. Instability to get much improved in that space. That's them. They're good guys. Look out for them. And I'm glad Marcello left, because he's way above this class. And that's me. That's the class. Hopefully that was useful.