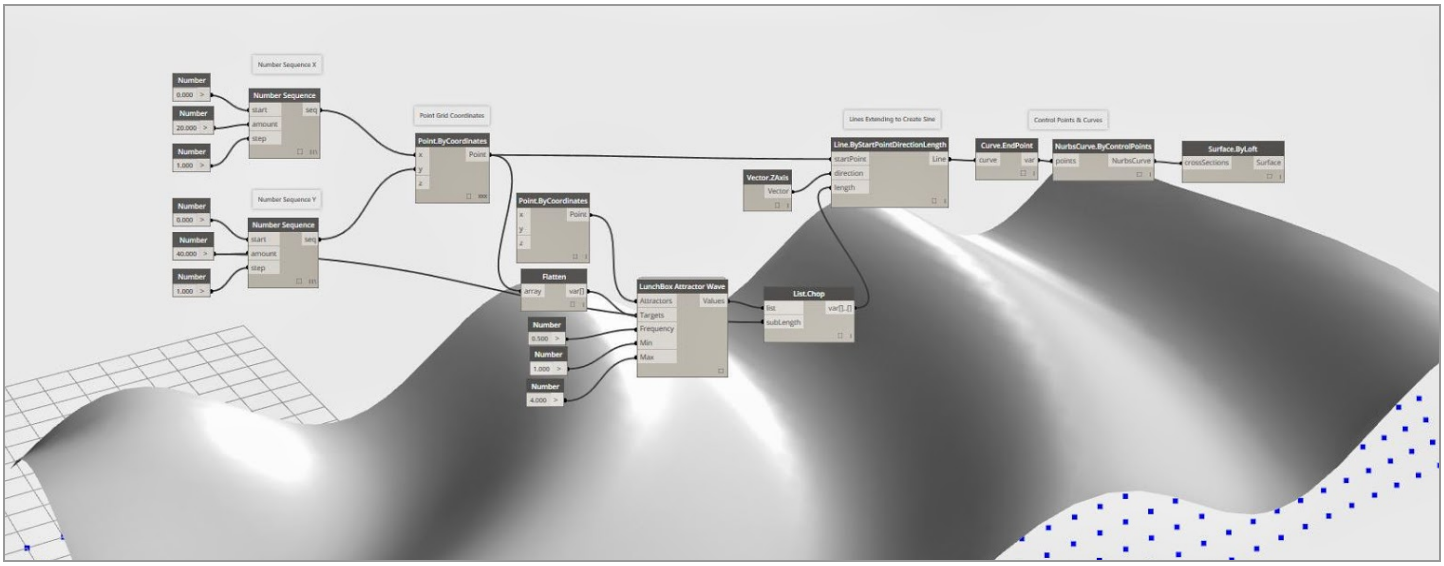
White-Glove Packaging
Creating Custom Dynamo Nodes

Autodesk University, 2014

Nathan Miller, n.miller@case-inc.com



1 - Class Summary

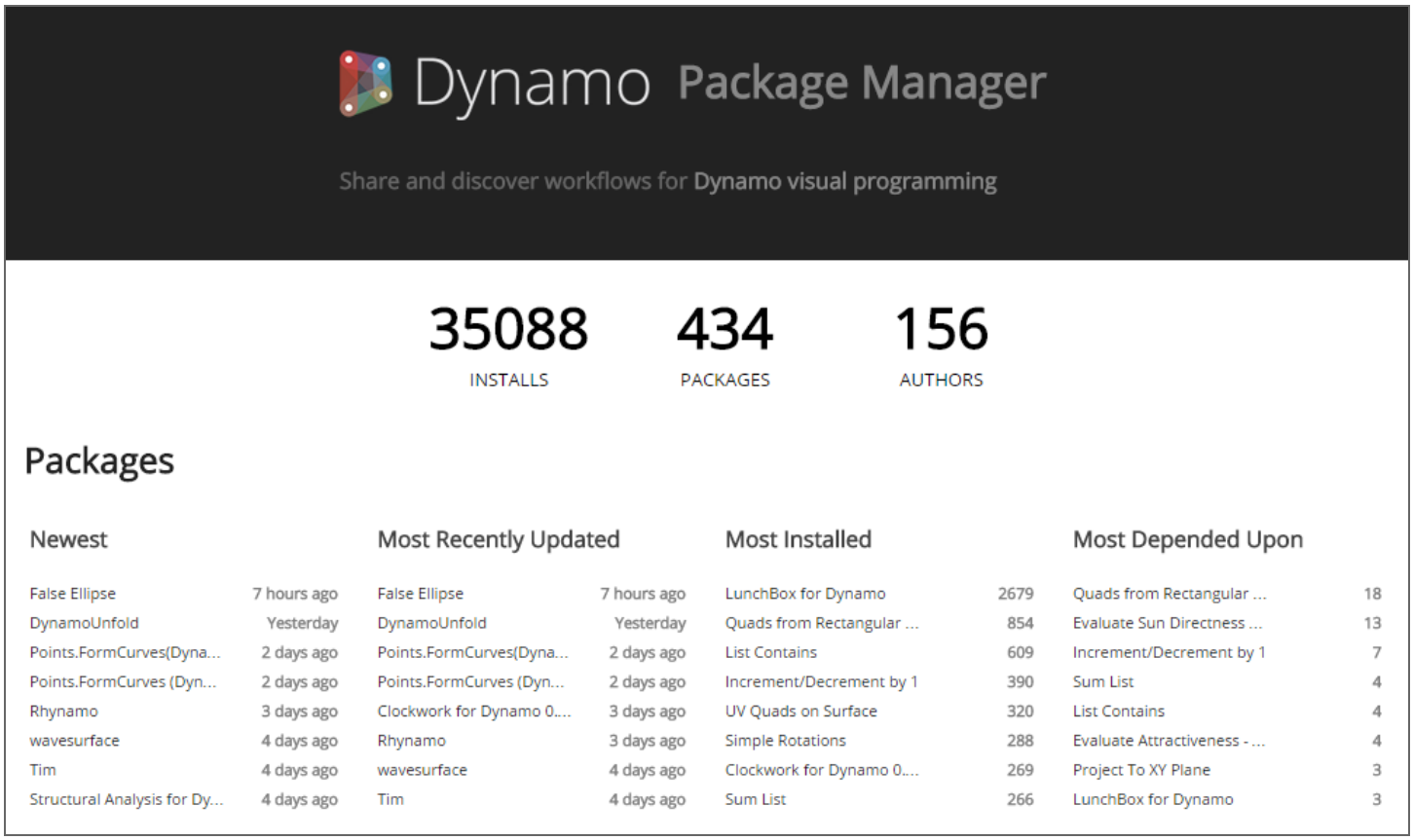
This class will discuss and demonstrate best practices for creating, sharing, and maintaining custom Dynamo extension nodes. The class will exposed three approaches to making Dynamo nodes driven by components, Python scripting and “zero touch” node libraries. We will also explore best practices for curating reusable nodes for sharing through the package manager. Finally, the class will showcase custom Dynamo extension packages, authored by the presenter, and walk through how these packages are used and maintained for the user community.

Learning Objectives

- Learn when and how to create custom Dynamo nodes
- Learn workflows for using Python scripting for customizing nodes.
- Learn about advanced node creation with a ‘zero touch’ class library.
- Learn about ‘best’ practices for using the package manager to host your custom nodes.
- Learn about packages currently available for use with the Dynamo extension

Class Itinerary

- Why Custom Nodes?
- Creating Custom Nodes with Components
- Creating Custom Nodes with Python
    - **Case Study:** LunchBox - Python Package
- Creating a Dynamo Node library with .NET
    - **Case Study:** Rhynamo - Node Library
- Curating and Publishing a Package
- Where Can I go to Learn More….?



http://DynamoPackages.com

2 - Why Custom Nodes?

Dynamo is more than an out-of-the-box tool: it is an extensible graphical language that can be customized to suit your needs. Whether you want to capture node sequences that you would like to reuse, or you need to build something completely new, Dynamo offers several different ways to customize your workflow. Dynamo also depends heavily on the user community to develop what are called “packages”. These are collections of custom nodes that users have made and have published to the community to use. Packages might expose brand new functionality or “fill in the gaps” with utilities So while you may not find necessarily making a custom node, you will definitely find yourself using one or more in your workflow… so it’s best to know what they are and how they work!

Custom nodes and packages don’t require you to be coder to make one. A custom node might be a definition you find yourself using time and time again. For the more code-savvy, you might leverage Python to define a new way to process data with Dynamo. For developers, you can create a .NET node library project using Visual Studio or other IDE.

So let’s get started… *your custom workflow awaits!*

3 - Creating Custom Nodes with Components

The easiest way to create a custom node is with Dynamo components. You might find yourself wanting to re-use different node definitions within a project or across multiple projects. A sequence of nodes can be "collapsed" into what is called a custom node which can then be accessed from your node menus like any other node.

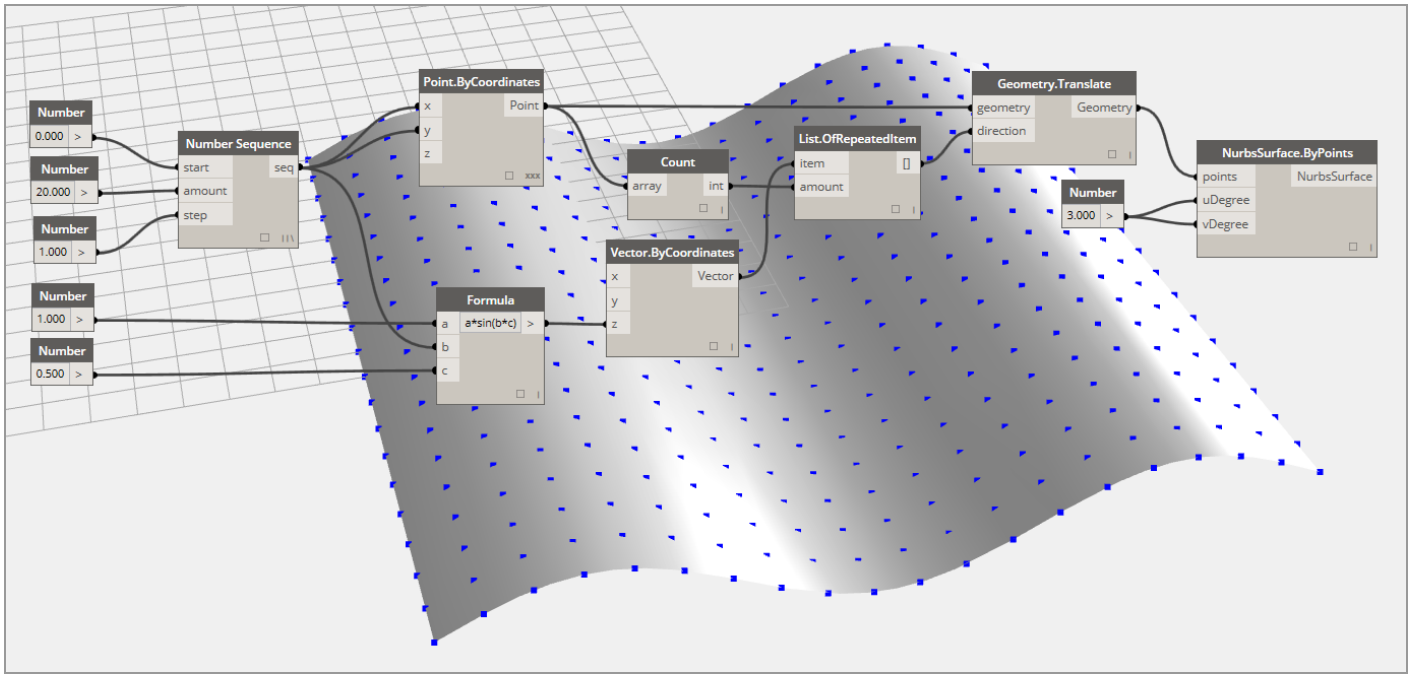
If you are using components to define a custom node, you can do it one of two ways…

- **Start from scratch:** You can always start a new custom node by choosing the "Custom Node" option when Dynamo first starts, or from ***File > New > Custom Node...***
- **Create from an Existing Definition:** If you have an existing definition that you would like to turn into a custom node, you can select your nodes within the workspace, right-click the canvas, and choose ***"New Node from Selection"***

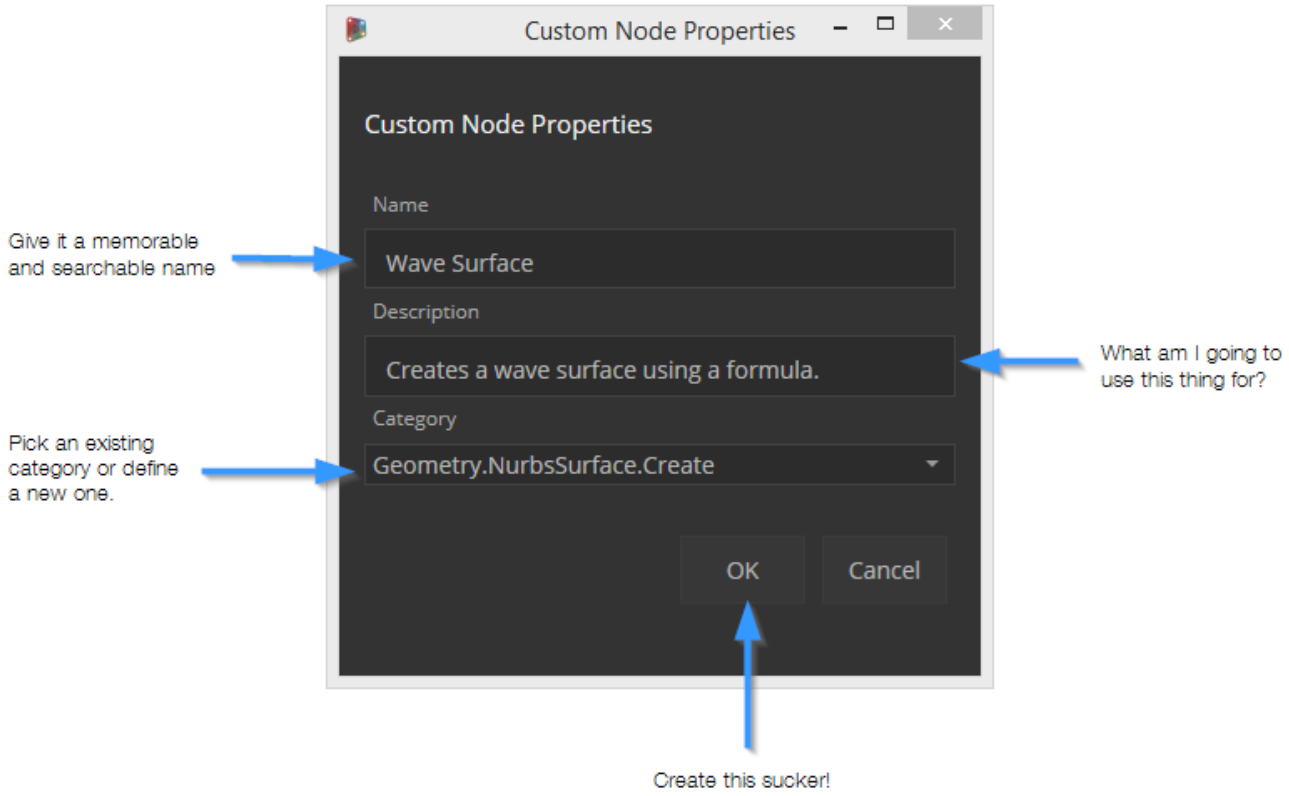
Example: A Custom Wave Surface Node

This example shows a definition that defines a surface using a simple Sine wave formula. All nodes used in this example are "native", meaning they are from the core library that ships with Dynamo. The inputs are a series of numeric values. The output is a NURBS surface. This mathematical shape may prove valuable in a future workflow or design exploration, so it might make sense to turn this into a custom node that we can re-use.

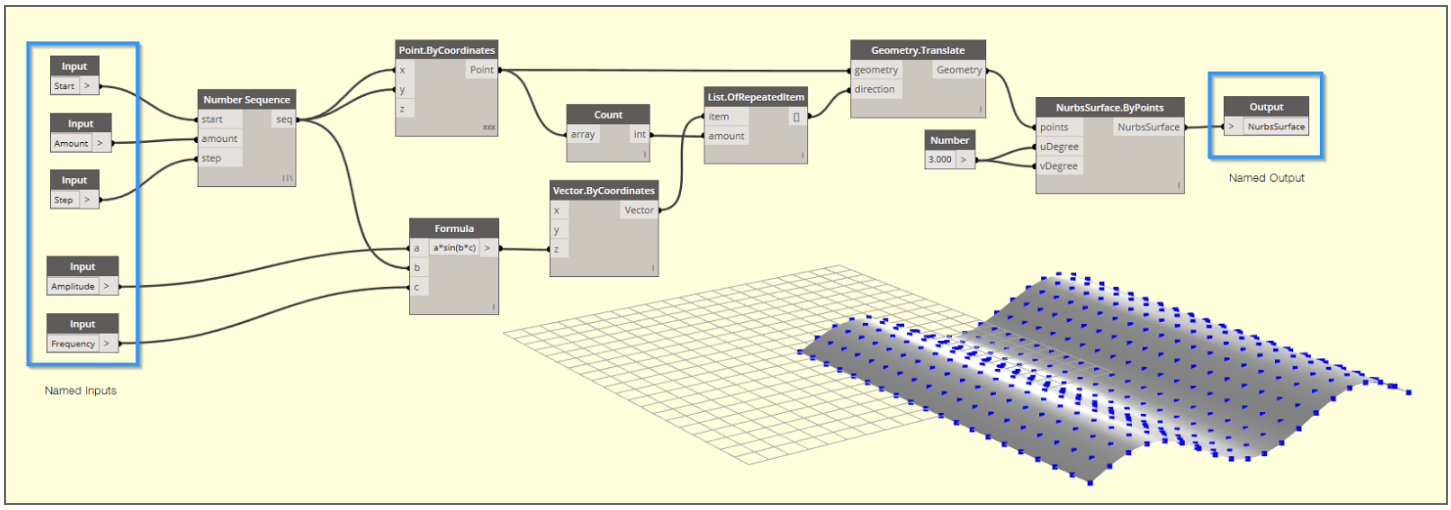
1. Select the definition nodes and choose **"New Node from Selection"**



2. Define the Custom Node properties… make sure you name and describe the node properly.

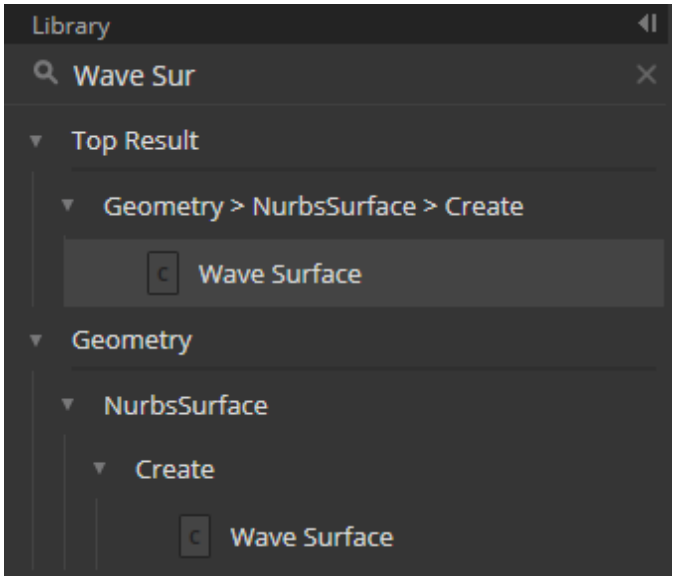


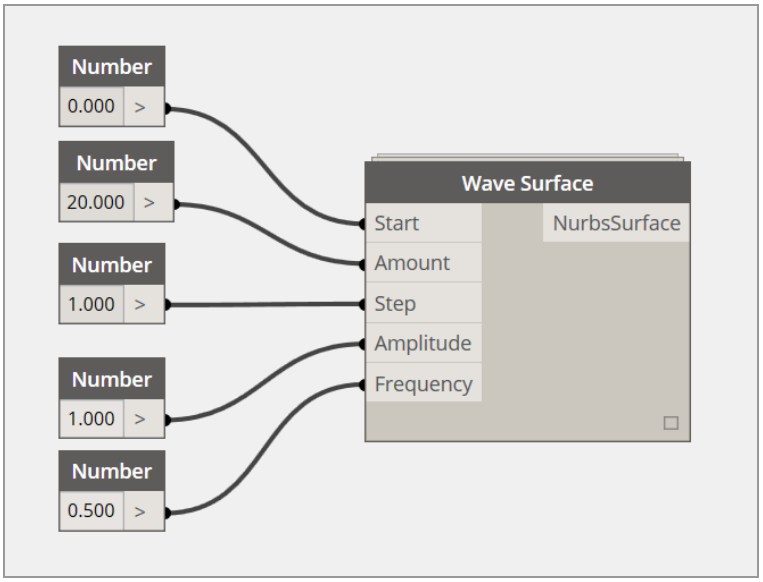
3. Within the Custom Node (note the *yellow-ish* background), you will want to do a little housekeeping to make sure your Inputs and Outputs are named properly.



4. After you create a Custom Node, you will need to save it to your Dynamo definitions folder located at **D:\Users\<USER>\AppData\Roaming\Dynamo\0.7\definitions** Custom nodes have a special extension ***.dyf**

5. Within a home workspace, you can now search for your Dynamo node and reuse it throughout your Definitions.

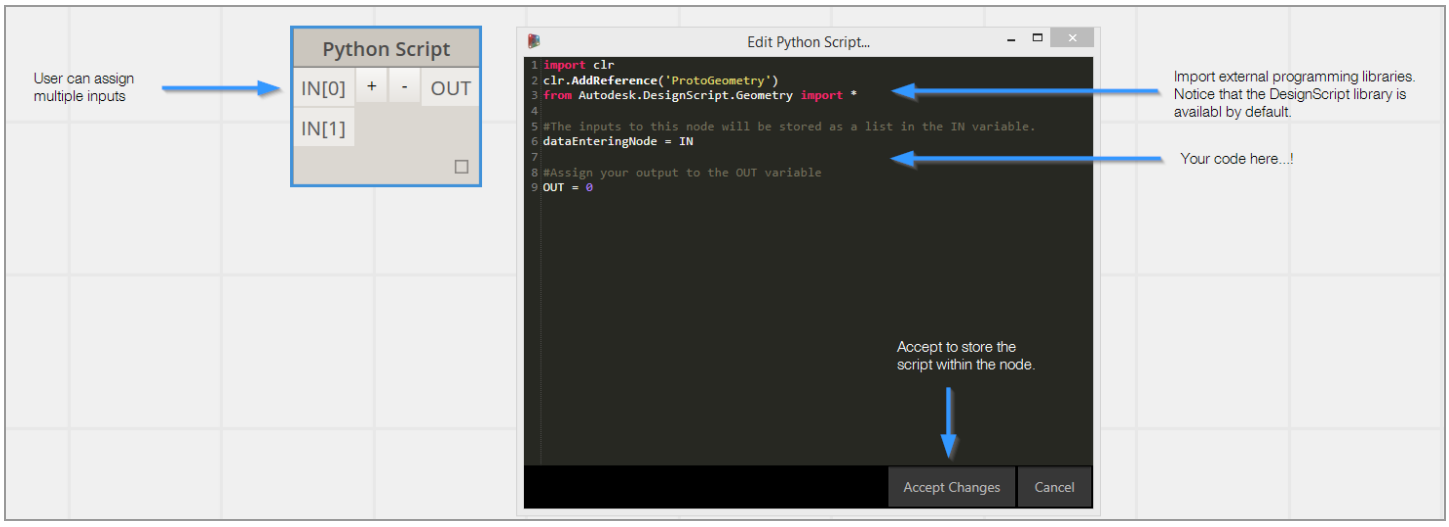




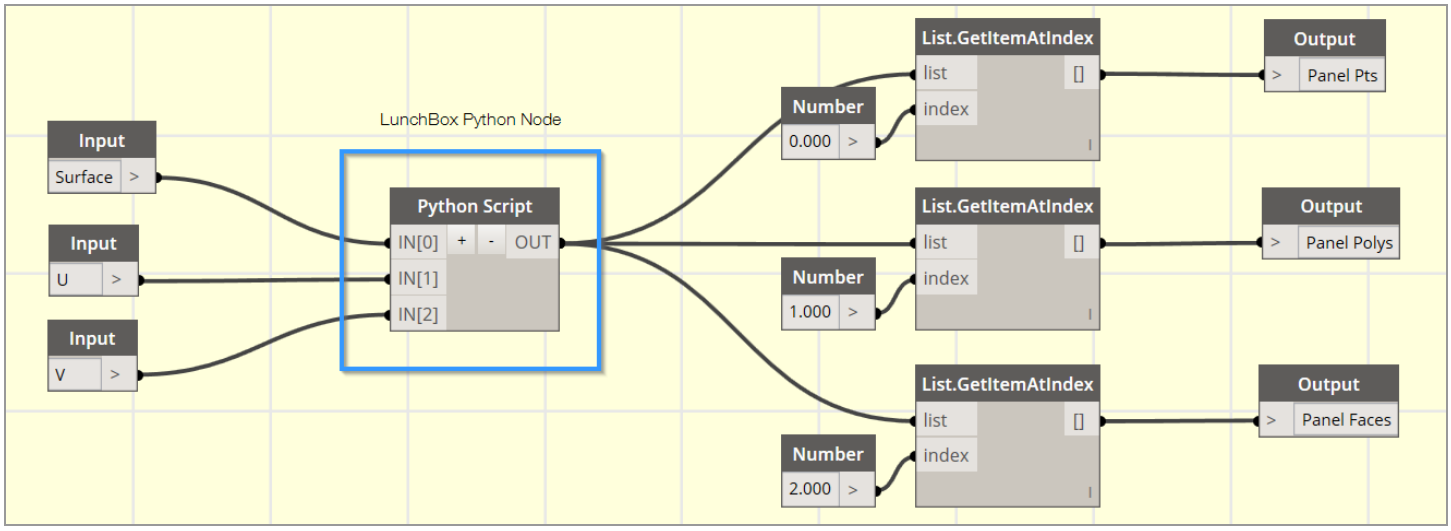
4 - Creating Custom Nodes with Python

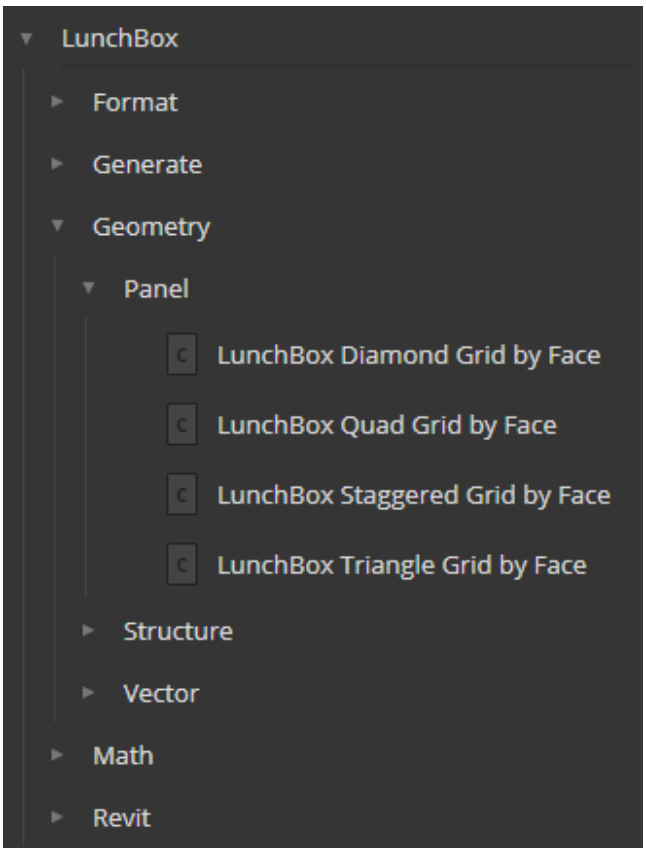
As you start to build definitions with Dynamo, you will find times where you will need to make new nodes that need a little scripting elbow-grease. Luckily, Dynamo exposes Python nodes that let you extend Dynamo with a popular programming language. While I won't be able to teach you Python in this course, I can give you an overview of how Python can be used to create new nodes.

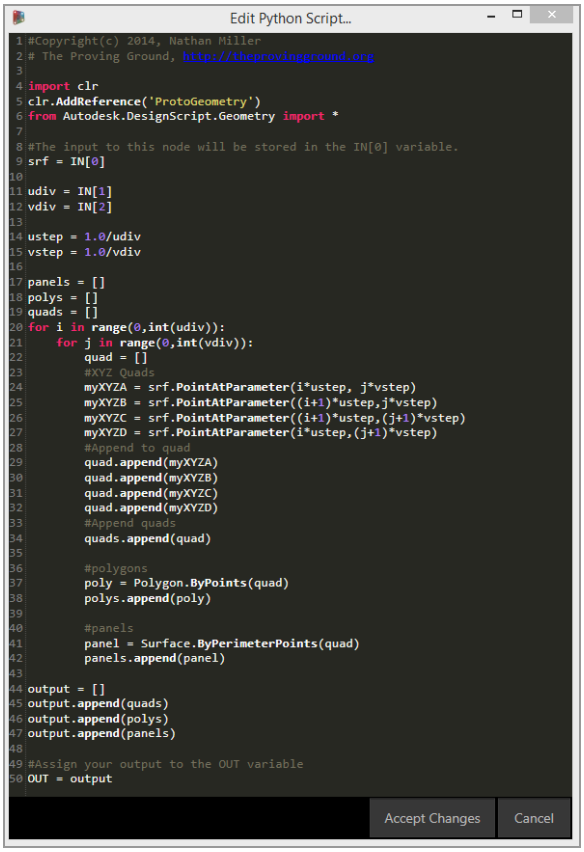
- Python nodes allow you to develop new code within Dynamo and have it execute like any other Dynamo node.



- Once you have your Python code developed, you can encapsulate the Python node within a Custom Node just like with other Dynamo components. Many Custom Nodes available in the Package Manager leverage Python to introduce custom functionality.
- The LunchBox tools for Dynamo includes over 30 nodes that use Python within the Custom Node. The Python code is exposed within this Package and show a variety of different applications.



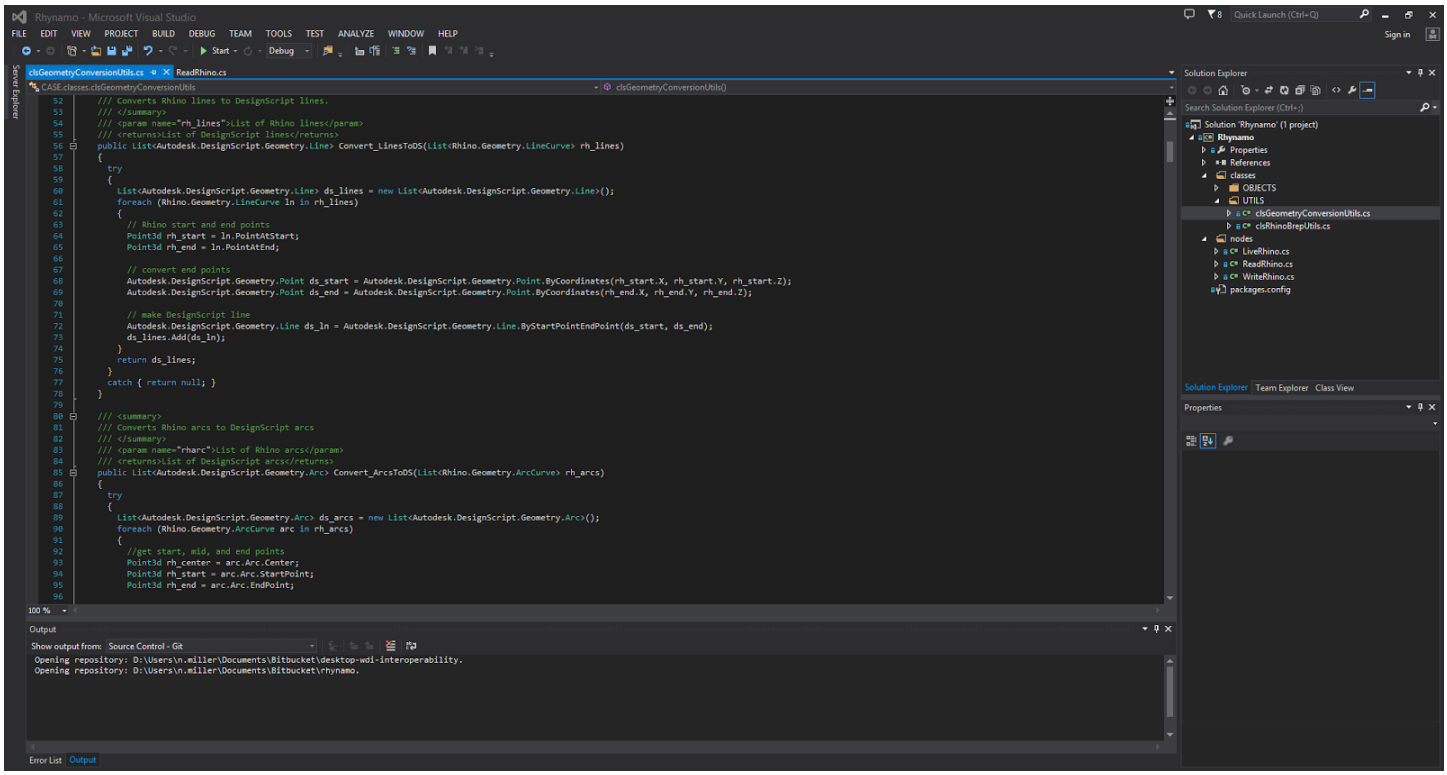




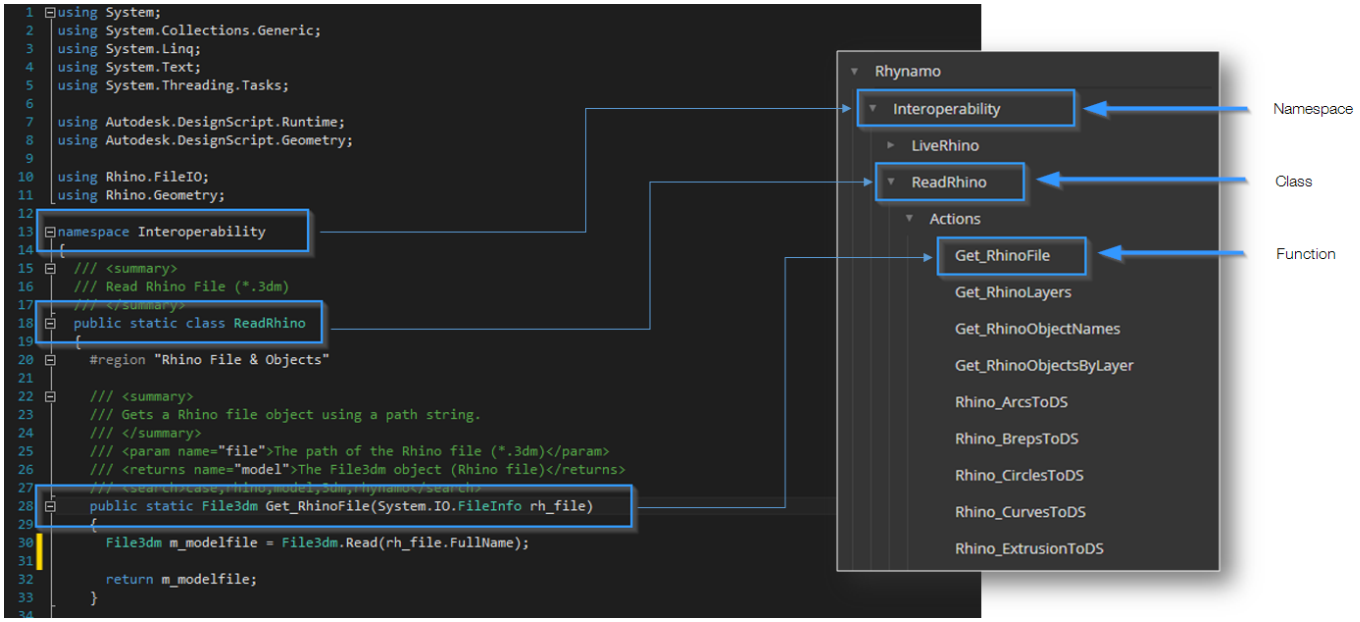
5 - Creating a Dynamo Node Library with .NET

Beyond creating custom nodes within the Dynamo application, it is also possible to develop a node library as a .NET DLL. Developing a node library in this manner requires deeper object oriented programming chops and a familiarity with integrated development environments (IDEs) such as Visual Studio. There is no hard or fast rule as to when to develop a .NET node library over, say, using a Python script. Where I draw the line is in the scale of development effort. I generally only will use Dynamo's Python node for smaller, contained scripts. A .NET node library lets me manage greater program complexity, track bugs, and leverage all the comforts an IDE affords. I have also found node libraries to be much faster within Dynamo.

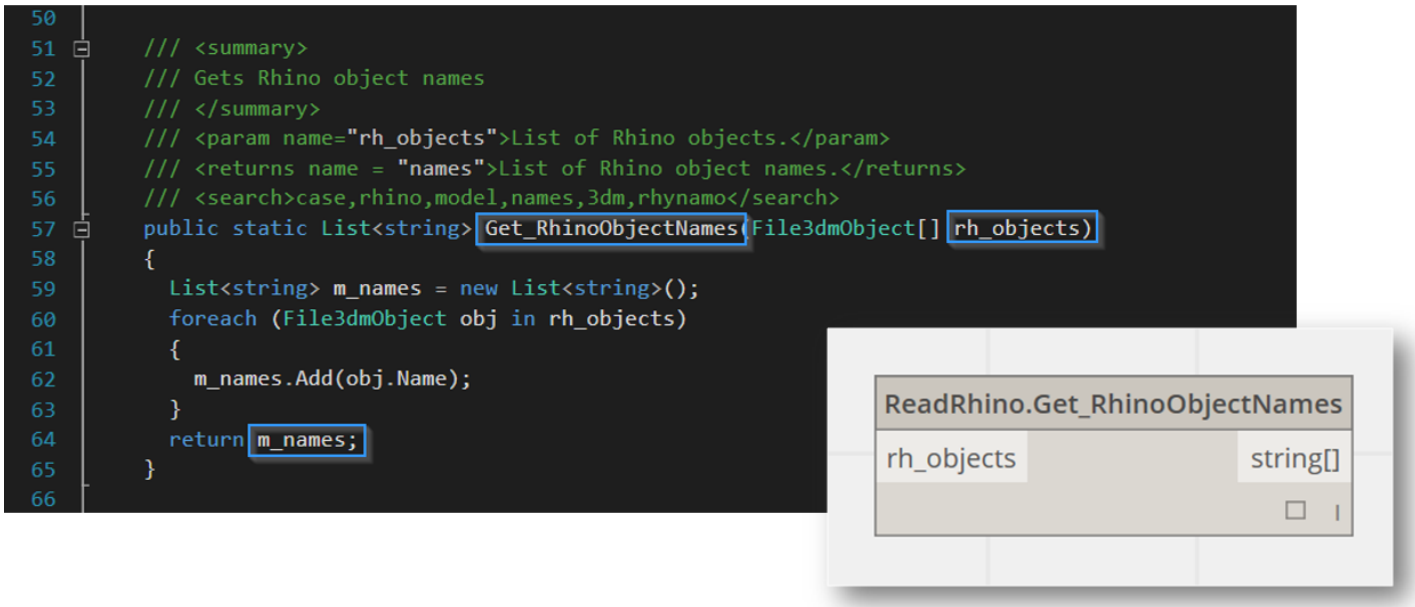
- So what are the rules of the road? Well first you will need an integrated development environment (IDE) such as Visual Studio (I use 2013… in 'Dark' mode… because it's cool)
- A developer will build their node library within the IDE using a .NET programming language. Dynamo is written in C#. There is also documentation for developing nodes in C#. However, if VB.NET is your preferred language, there is nothing stopping you from using that language.
- There is very minimal setup needed to ensure your project is "Dynamo" ready:
    - Make sure your project debug program is point to **DynamoSandbox.exe** in your Program Files.
    - Reference any libraries you anticipated needing. The most common ones I use are...
        - **ProtoGeometry.dll** (for Dynamo geometry)
        - **RevitAddinUtility.dll** (for Revit API access)



- Next we need to build a class that Dynamo will understand. The Dynamo interface and nodes are a very close graphical representation to an actual program. For example, you can see that the node library menu reflects how the a class is set up wherein you have a “Namespace”, a “Class”, and “Function”.



- A Dynamo node is actually a function represented graphically with input variables and a return variable as the output. The following is an example of a node contained within the Rhynamo node library. Rhynamo is currently available on the Package Manager. The source code is hosted on BitBucket.org.



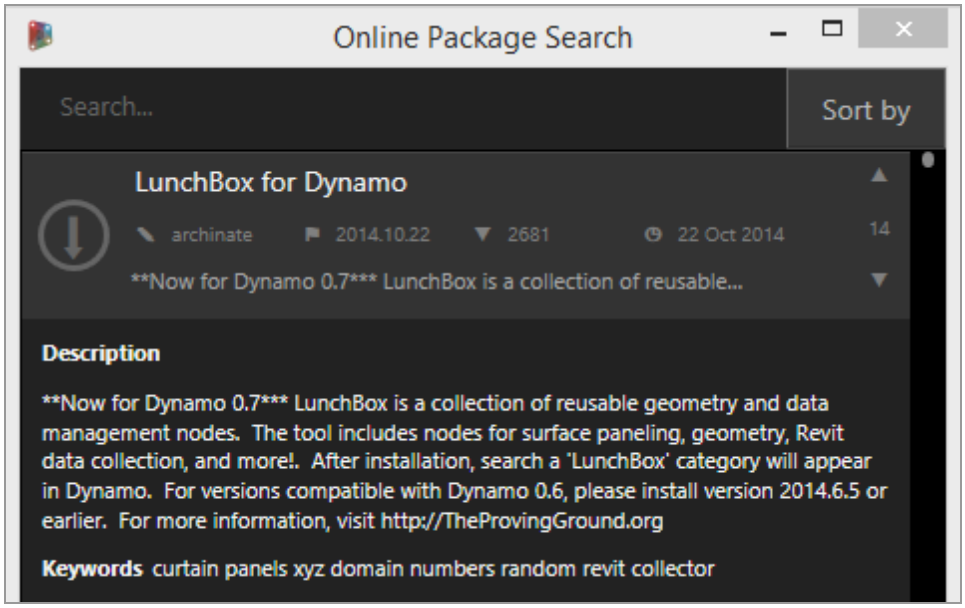
6 - Curating and Publishing a Package

So now that you've spent a good amount of time developing your very own collection of Dynamo nodes, you think you'd like to share it with the greater community? Great! Dynamo uses what is called a "Package" that allows user to curate and share their nodes with the rest of the community. Users are free to include just about anything they want in their package including custom nodes, example definitions, node libraries, and reference files. Users are able to publish new versions of their packages as they evolve and grow.

There is also a "dark side" to the package manager: since anyone can publish to it, there is going to be a lot of junk floating around in there. So how can you ensure the work you publish ends up at the top of everyone's list?

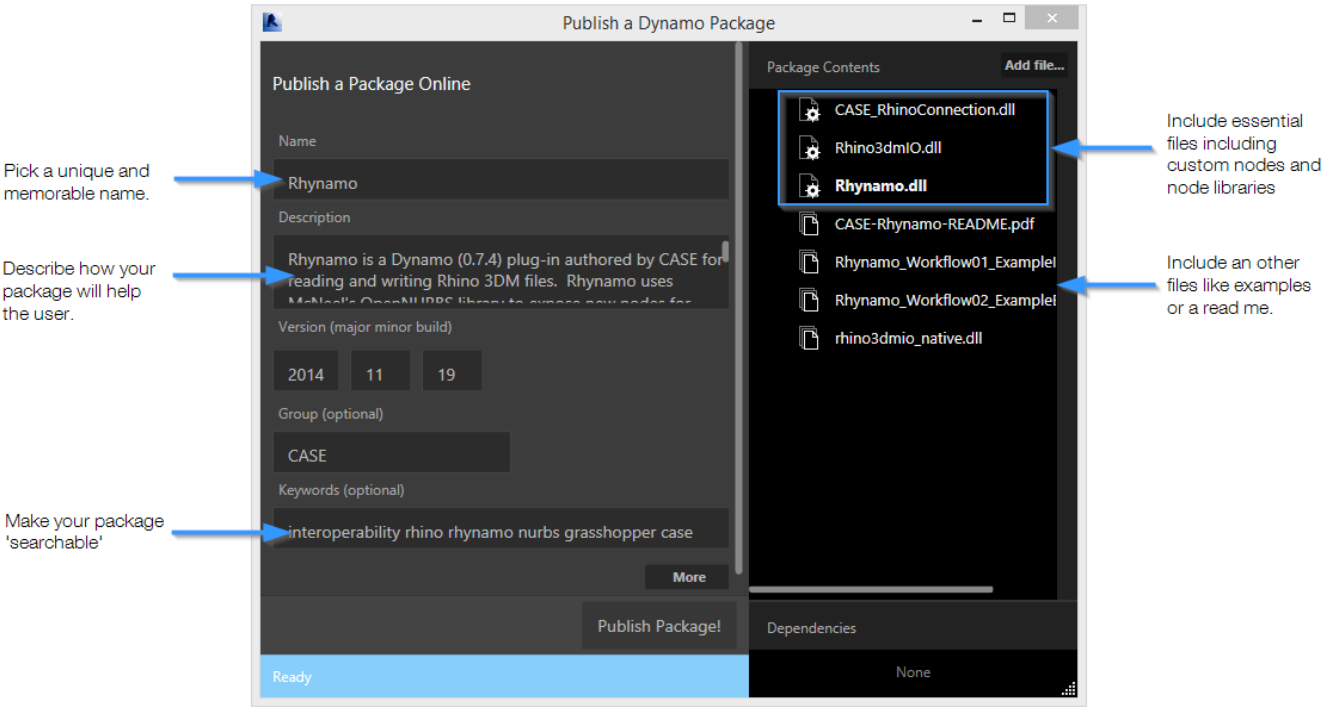
Nate's Tips for a Successful Package

- **Brand your package.** It may sound trivial, but the most used packages have a clear identity to them. Try to capture what your package is 'about'. Is it a growing collection of helpful utilities? Or are your nodes designed for a specific purpose? Find a way to tell the story of what your new tools are all about.
- **Be aware of what's out there.** What need is your package attempting to satisfy? Are you offering something new to the community? Are you creating a useful alternative to some existing packages? It's always good to know what is out there.
- **Maintain your package.** Dynamo is always changing and growing. Creating a package takes commitment. People will use them and come to depend on them. Be committed to maintaining your package and publishing new version to keep up with the latest in Dynamo.
- **Add value, not noise.** Be critical about what you are about to release to the world. There is a lot of 'noise' in the package manager these days… don't just publish for the sake of publishing.

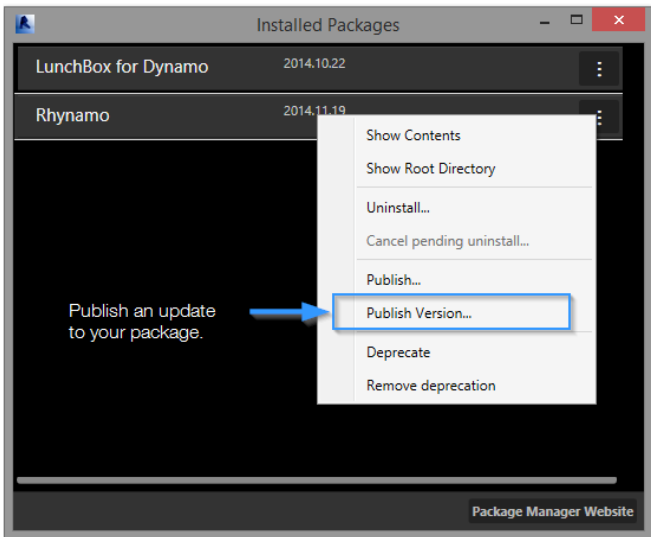


Example: Creating a Package

- Creating and publishing a package is easy. When you have a custom node or a node library ready, go to **Packages > Publish New Package**. Be sure to include essential files. Read mes and example files are also useful to demonstrate the new functionality your are offering.



- After you have published a package for the first time, it will become searchable and listed within the package manager. You will probably find the need to publish new versions of your package as you update and expand your tools. Dynamo makes it easy to publish versions. Simply go to Packages > Manage Packages. Click on the menu next to the package and select “Update Package”



7 - Where Can I Go to Learn More?

By now you have realized that there is a lot to the subject of developing custom nodes and curating a Dynamo package. Dynamo makes it easy to immediately begin customizing nodes while also offering more advanced means of developing entire libraries. Creating and publishing packages is fairly easy, but there are many important considerations if you want to make a meaningful contribution to the community. There are also many learning resources out there…

- http://DynamoBIM.org has a great user community and introductory videos. It is a portal to the rest of the Dynamo world and will often be your first stop.
- https://github.com/DynamoDS/Dynamo/wiki is the source code wiki that contains some rather important articles. For example, a more detailed overview of writing a "Zero Touch" library is described here: https://github.com/DynamoDS/Dynamo/wiki/Zero-Touch-Plugin-Development
- http://www.dynamopackages.com has an updated list of the newest and most popular packages in the community. I am an advocate of learning by doing…download packages that are interesting to you and see how the authors made their custom nodes. Analyze their Python scripts and see how they have organized their library. You can learn a lot from examples. And you might even want to ask the author to show you a thing or two.