AUTODESK UNIVERSITY

SD468472
Power-Ups and Cheat Codes: Tips and Tricks for the Fusion 360 API

Patrick Rainsberry, Jesse Rosalia, Brian Ekins,
George Roberts, Scott Moyse

About the speaker

Patrick Rainsberry

I am a mechanical engineer. I have an undergrad degree from UC Berkeley and a Masters from UCLA. I also have an MBA from the University of La Verne. I have been working in the CAD industry for over 15 years as well as some time spent as a design engineer. Currently I am a Product Manager for Fusion 360, working on various projects related to desktop and cloud API's and various other projects related to the data management experience in Fusion 360.

About the speaker

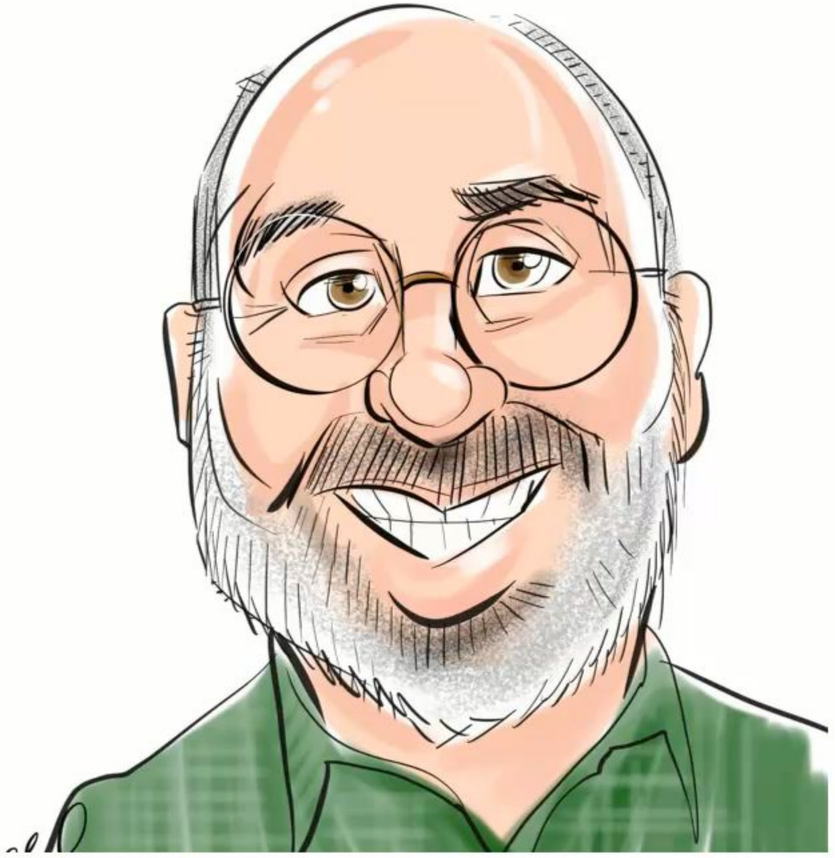
Jesse Rosalia

I am the founder of Bommer, and the developer of the popular Bommer for Fusion 360 add-in. I have nearly 20 years of experience in software development, including 2 years as the CTO of a small robotics startup. I am interested in the intersection of hardware and software, particularly the use of software tools to aid in the hardware development process, and have a passion for building usable tools that delight and improve the lives of users everywhere. Proud alumnus of Georgia Tech, and loving husband and cat co-parent.

About the speaker

George Roberts

Currently a Product Manager responsible for connected manufacturing within Fusion 360 CAM. I was a CAD/CAM applications engineer at Man and Machine with over 5 years of manufacturing industry experience and over 2 years in training CAD/CAM. Throughout the past few years, all of his spare time has been spent using, customising, and testing various different CAD and CAM packages. His firm knowledge of CNC machining and software development makes him the ideal speaker for CAD/CAM packages.



About the speaker

Brian Ekins

Ekins Solutions LLC

- Solid Edge, Inventor, and Fusion 360 API Designer
- Developer Support and Consultant at Autodesk
- Run my own consulting company where I help companies customize Inventor and Fusion 360 to work better for them.

Scott Moyse

MFG & CAM Technical Specialist

- **UI / UX Designer & Product Owner for the Massif.dev licensing & IP protection Platform**
- **CADPRO Systems – Platinum Reseller in New Zealand**
- **Post Processor Developer**
- **Autodesk Expert Elite**

Email: scott.m@massif.dev

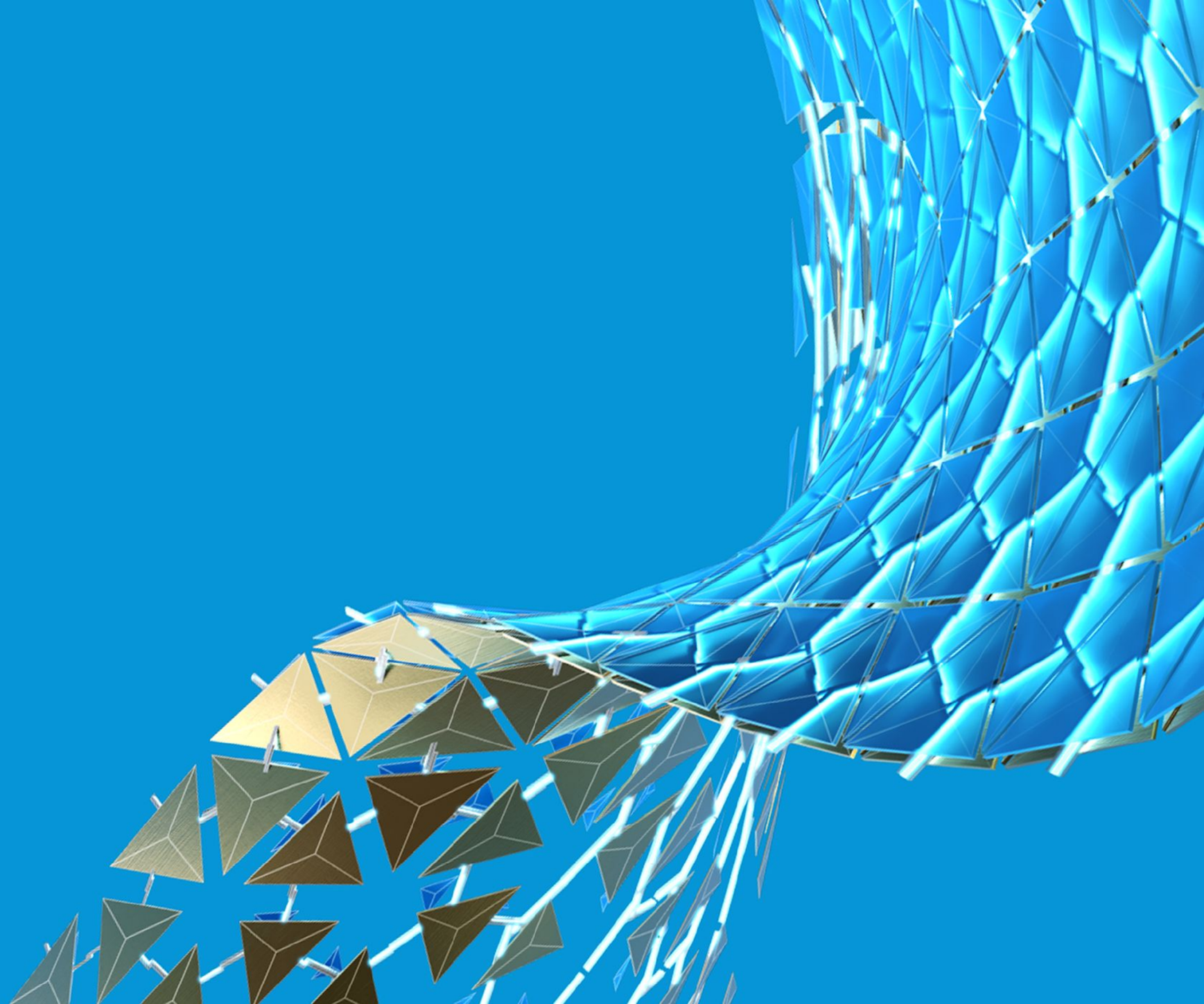
MASSIF

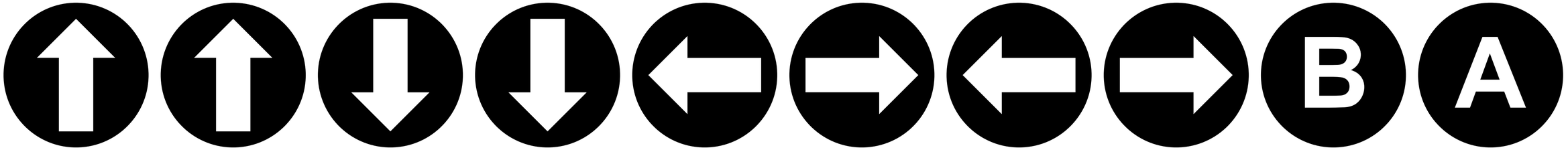
Licensing • eStore • Delivery

Agenda

- Intro (*Patrick*)
- USE THE DOCS!!! (*Patrick*)
- Learning the Help Files (*Brian*)
- Background Threads (*Jesse*)
- Custom Events (*Jesse*)
- Using Palettes (*Patrick*)
- CAM API's (*George*)
- Text Commands (*Brian*)
- Fusion 360 Data ID's (*Patrick*)
- Securing Python and Distribution with MASSIF (*Scott*)
- VS Code Debugging (*Brian*)
- Working with Forge API's (*Patrick*)

Intro





START

Why we do it...

Scott Moyse 2:02 PM

I don't suppose you have a script handy for dropping a joint origin on a group of selected sketch points do you? If not I'm writing a command to do it.

Patrick Rainsberry 2:34 PM

I could... Do you have an API tip for my class? Hahahah How do you want to define orientation? Match global CS directions?

Why we do it...

Scott Moyse 2:02 PM
I don't suppose you have a script handy for dropping a joint origin on a group of selected sketch points do you? If not I'm writing a command to do it.

Patrick Rainsberry 2:34 PM
I could... Do you have an API tip for my class? Hahahah How do you want to define orientation? Match global CS directions?

Patrick Rainsberry 2:52 PM
Dude who loves you?

Zip

MoyseJoints.zip
6 kB Zip

Was half paying attention in a conference call and knocked it out

❤️ 1

Why we do it...

Scott Moyse 2:02 PM
I don't suppose you have a script handy for dropping a joint origin on a group of selected sketch points do you? If not I'm writing a command to do it.

Patrick Rainsberry 2:34 PM
I could... Do you have an API tip for my class? Hahahah How do you want to define orientation? Match global CS directions?

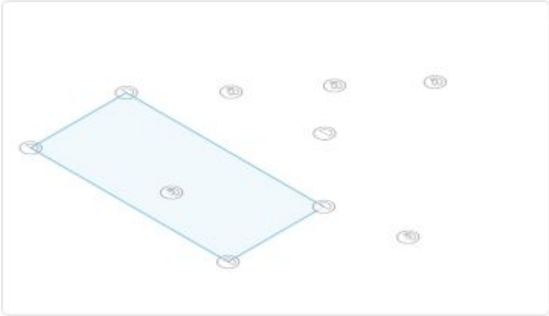
Patrick Rainsberry 2:52 PM
Dude who loves you?

Zip

MoyseJoints.zip
6 kB Zip

Was half paying attention in a conference call and knocked it out

❤️ 1



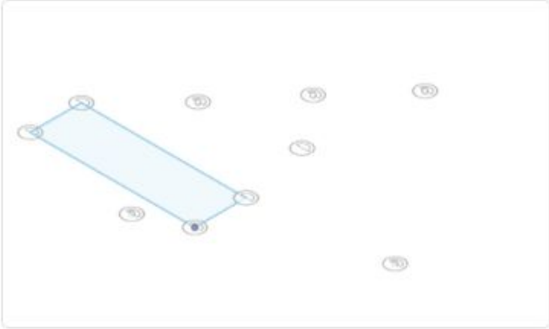
That was easy LOL

Scott Moyse 2:58 PM
Yeah... I figured it would be hence why I was confident I could do it

I was curious if I would have to jump through a load of API bollocks to get them to remain associative to the points moving

Patrick Rainsberry 3:01 PM
Nope they are asccociative.

image.png



Scott Moyse 3:02 PM
epic

Why we do it...

Scott Moyse 2:02 PM
I don't suppose you have a script handy for dropping a joint origin on a group of selected sketch points do you? If not I'm writing a command to do it.

Patrick Rainsberry 2:34 PM
I could... Do you have an API tip for my class? Hahahah How do you want to define orientation? Match global CS directions?

GLORY

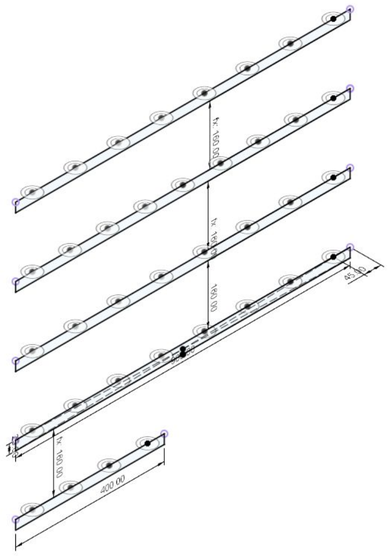
Patrick Rainsberry 2:52 PM
Dude who loves you?

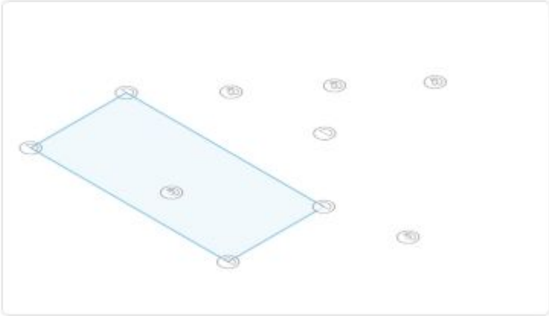
Zip

MoyseJoints.zip
6 kB Zip

Was half paying attention in a conference call and knocked it out

❤️ 1





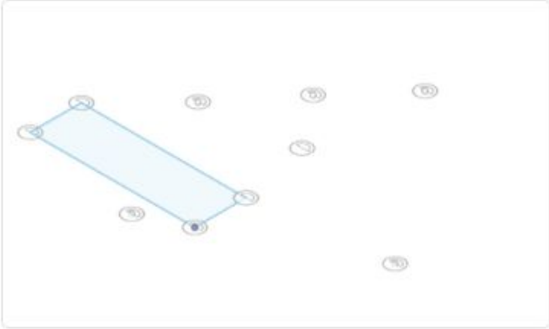
That was easy LOL

Scott Moyse 2:58 PM
Yeah... I figured it would be hence why I was confident I could do it

I was curious if I would have to jump through a load of API bollocks to get them to remain associative to the points moving

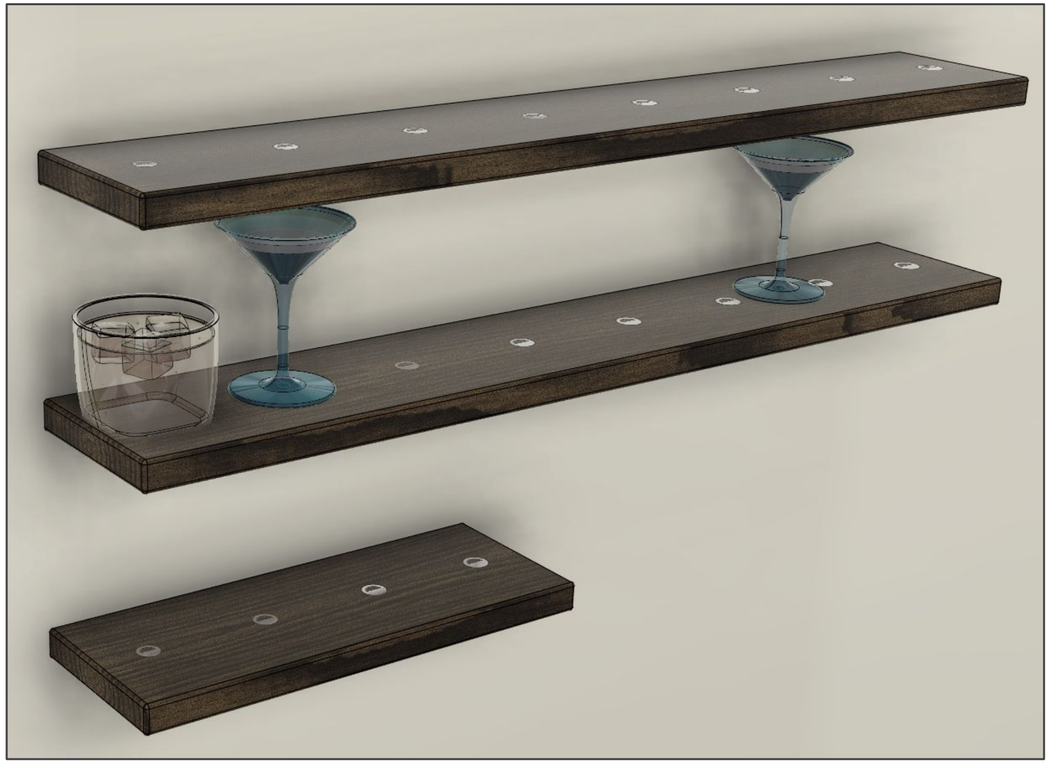
Patrick Rainsberry 3:01 PM
Nope they are asccociative.

image.png

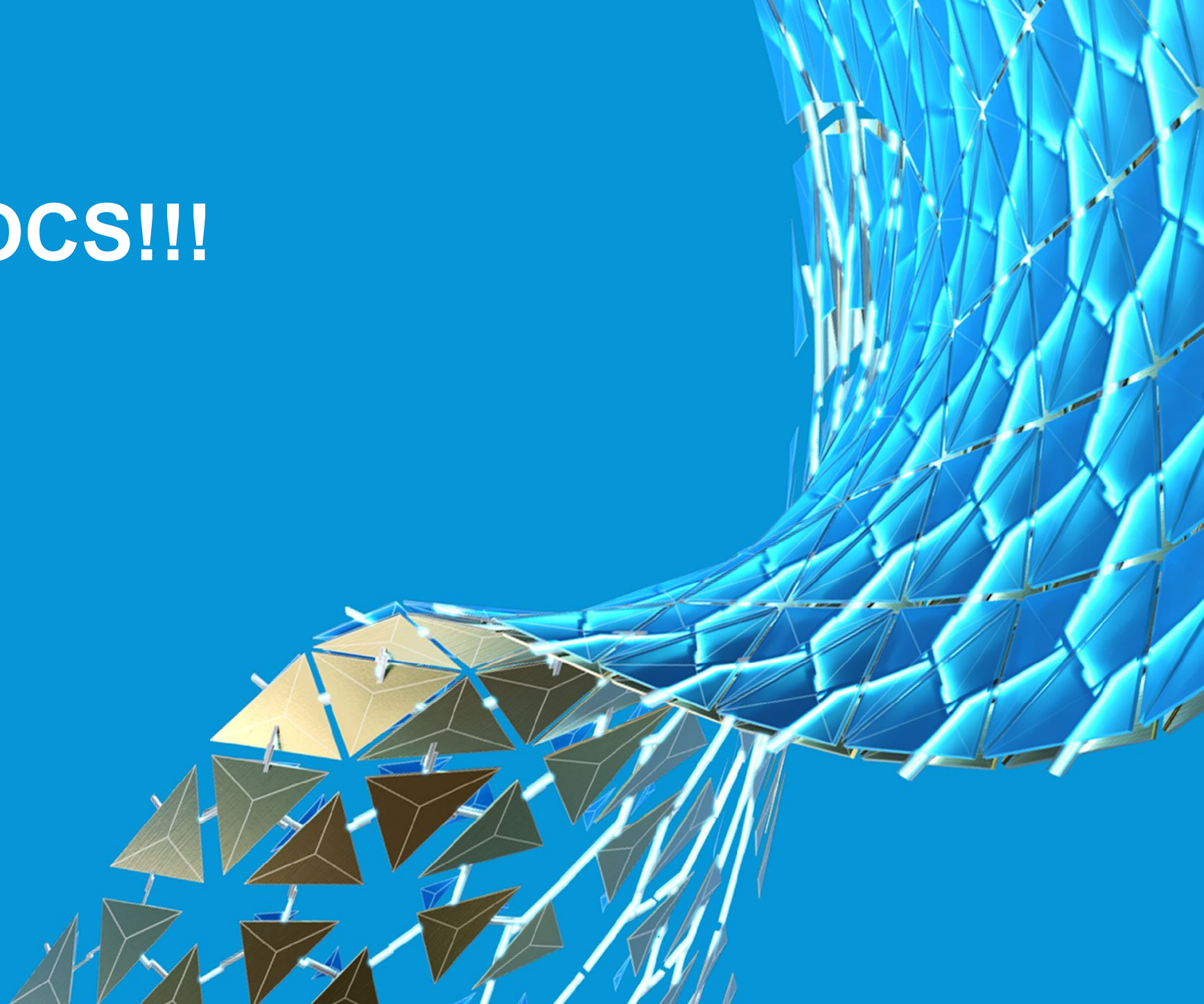


Scott Moyse 3:02 PM
epic

Getting Tipsy with Fusion 360



USE THE DOCS!!!



Using the Documentation

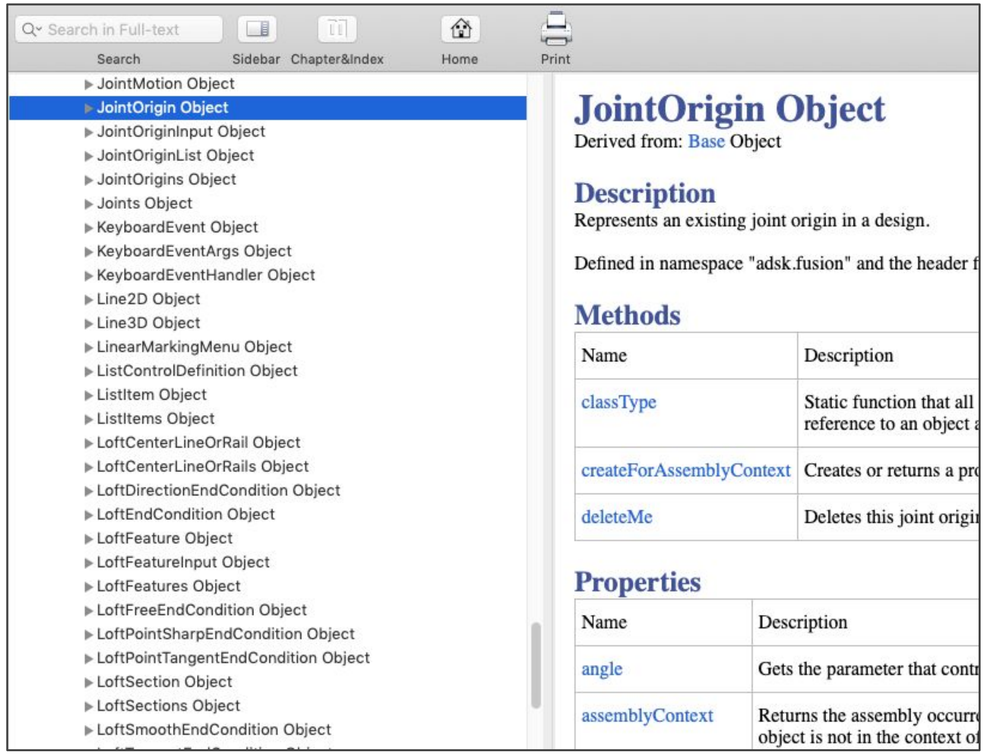
DOWNLOAD THE OFFLINE VERSION!

https://help.autodesk.com/cloudhelp/ENU/Fusion-360-API/SupportFiles/FusionAPI.chm

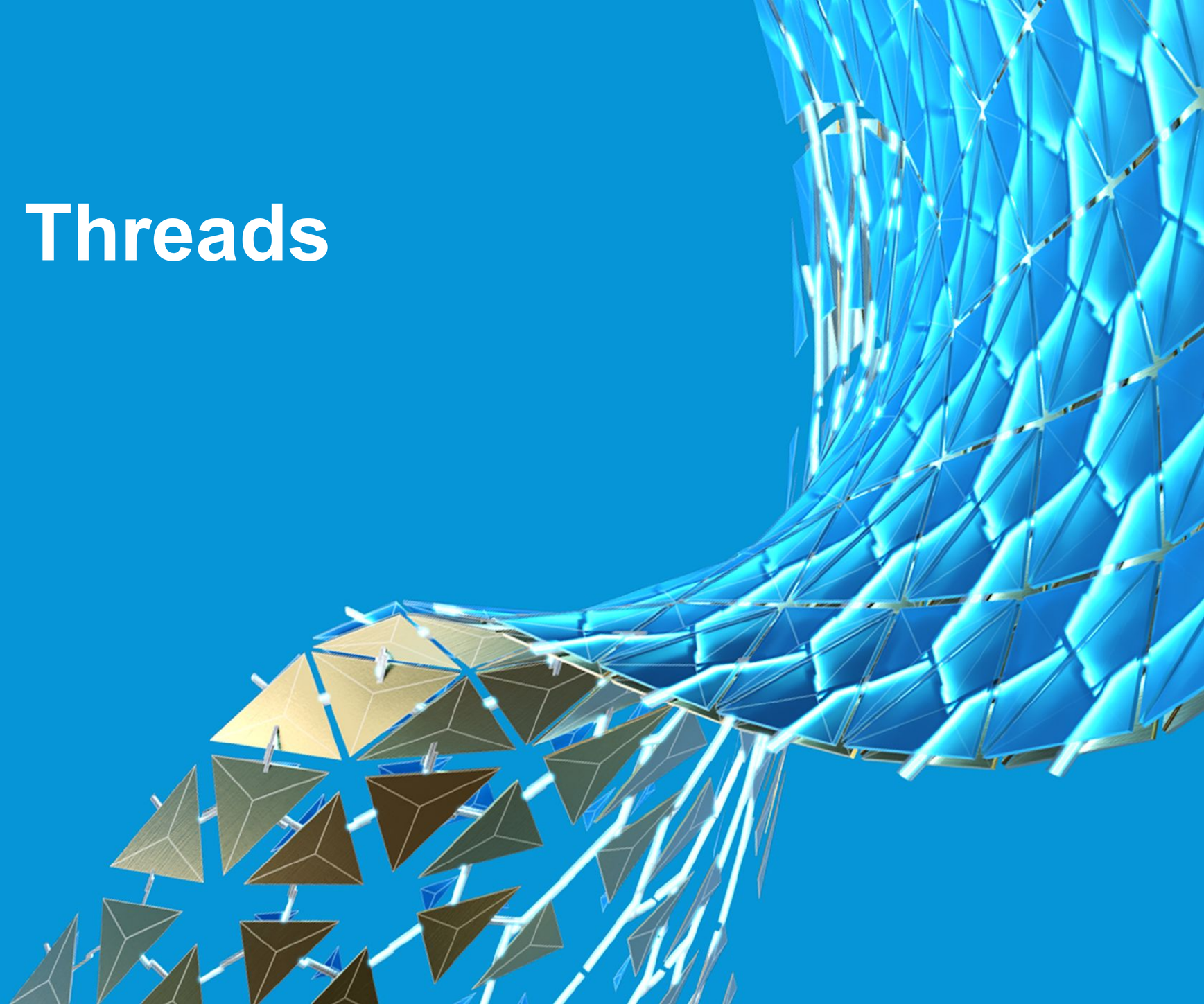
On Mac you may need to get a chm viewer. I use [CHM View](#).

You can do ANYTHING if you learn how to navigate the docs.

… and a little [Stack Overflow](#)



Background Threads



Why background threads?

IO-bound tasks, such as:

- File I/O and monitoring
- Communicating with remote servers

Tasks that would block the user interface

(that are not UI tasks)

Creating and running a background thread

Derive a class from threading.Thread, and override the `run` method

run will run once and the thread will die when it exists

use a `while` loop with a sentinel value to keep the thread alive

```
class BackgroundThread(threading.Thread):

    def __init__(self):
        super().__init__()
        self.running = True

    def run(self):
        while self.running:
            # 1) watch for file changes, or continue
            # 2) read in file if changed
            # 3) send file to remote server
```

Instantiate the class and call `start` to start the thread

```
...
thread = BackgroundThread()
thread.start()
```

To close this thread, set the sentinel running to false

`thread.join()` will block until the thread exits

```
# Close thread
thread.running = False
thread.join()
```

A word of caution

Threads in your add-in can prevent Python/Fusion from closing

Consider making your thread a daemon thread if it doesn’t require a lot of clean-up

Shared resources may require synchronized access

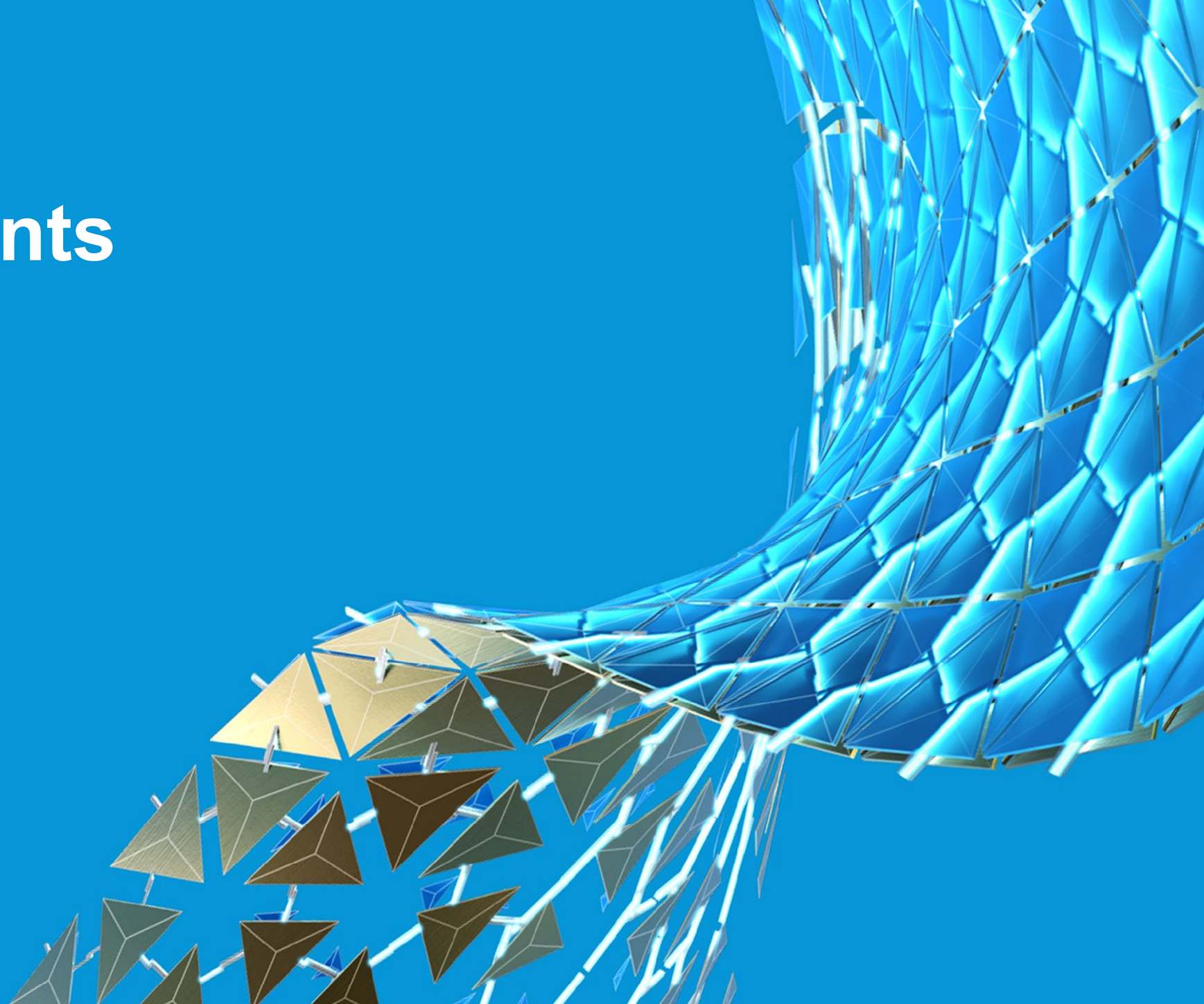
This includes UI updates!

Python’s global interface lock (GIL) limits effectiveness of threads for CPU-bound tasks

Most (if not all) Fusion API calls are not thread safe; some (UI API calls) may crash Fusion 360!

Custom events give you a mechanism for communicating from a background thread to the UI thread

Custom Events



Registering and cleaning up custom events

Derive a class from adsk.core.CustomEventHandler, and implement notify to handle the custom event

Register custom events using registerCustomEvent in adsk.core.Application

Add handlers to the returned CustomEvent object

Call unregisterCustomEvent to clean up the event

```
class CustomEventHandler(adsk.core.CustomEventHandler):
    def __init__(self):
        super().__init__()
    def notify(self, args):
        app = adsk.core.Application.get()
        ui  = app.userInterface
        try:
            message = args.additionalInfo
            ui.messageBox(message)
        except:
            ui.messageBox('Failed:\n{}'.format(traceback.format_exc()))
custom_event_handler = CustomEventHandler()
custom_event_id = 'CustomEvent'
def run(context):
    global custom_event_id
    app = adsk.core.Application.get()
    evt = app.registerCustomEvent(custom_event_id)
    evt.add(custom_event_handler)

def stop(context):
    app = adsk.core.Application.get()
    app.unregisterCustomEvent(custom_event_id)
```

Firing custom events

Call fireCustomEvent to fire the event:

```
...
app = adsk.core.Application.get()
app.fireCustomEvent(event_id, additional_info)
...
```

Tips and tricks

Uuids make excellent event ids:

```
import uuid
app = adsk.core.Application.get()
event_id = uuid.uuid4().hex
app.registerCustomEvent(event_id)
…
```

Calling `adsk.doEvents()` may cause pending events to be handled

`additionalInfo` can be <u>any</u> string

e.g. json, or keys into a data structure

Tips and tricks

Consider a Queue for frequently called custom events

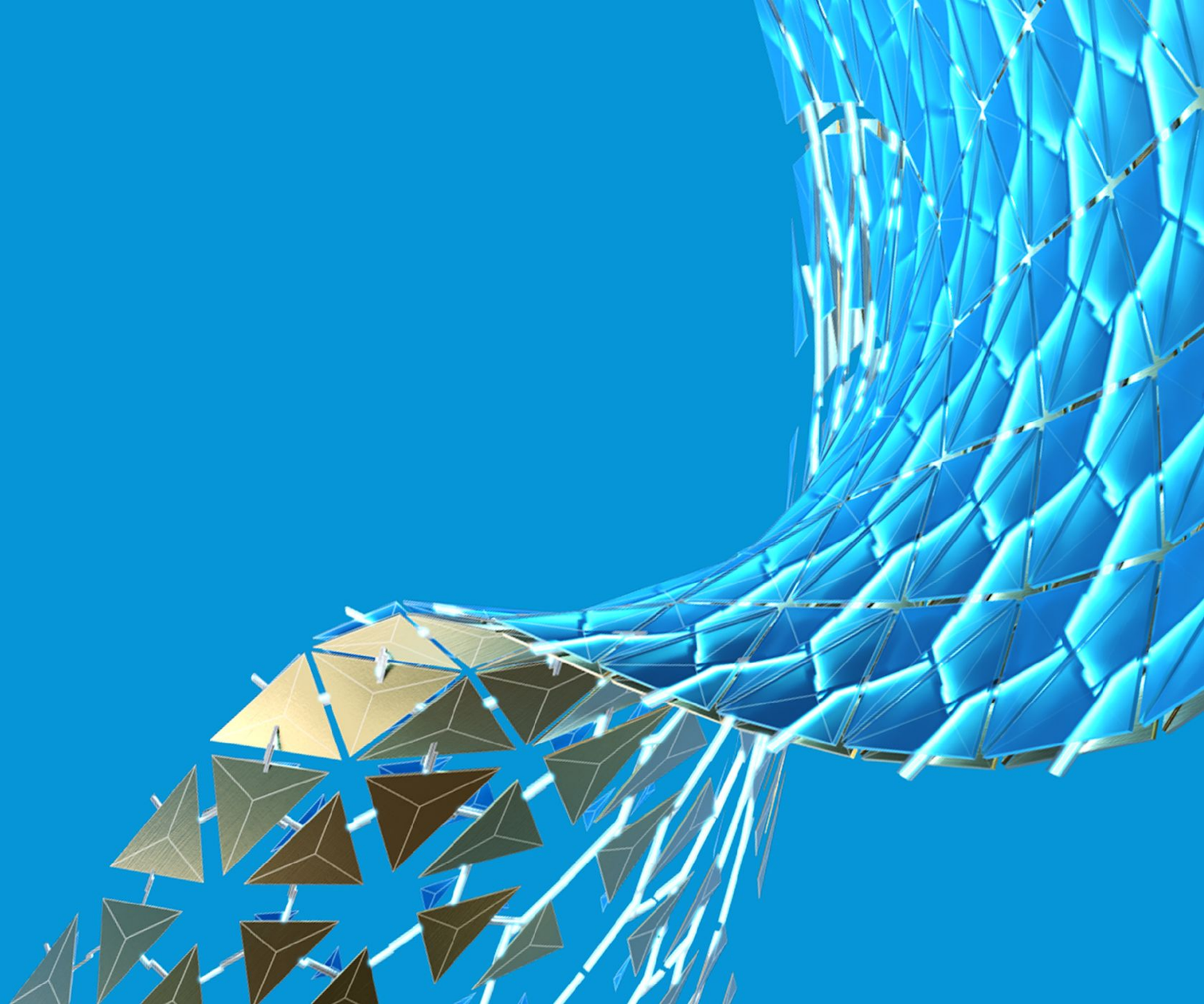
```
import queue
_queue = queue.Queue()

while not self._queue.empty():
    item = self._queue.get_nowait()

...

_queue.put_nowait(event_data)
app = adsk.core.Application.get()
app.fireCustomEvent(event_id, '')
```

Palettes



Palettes

Leverage client side libraries:

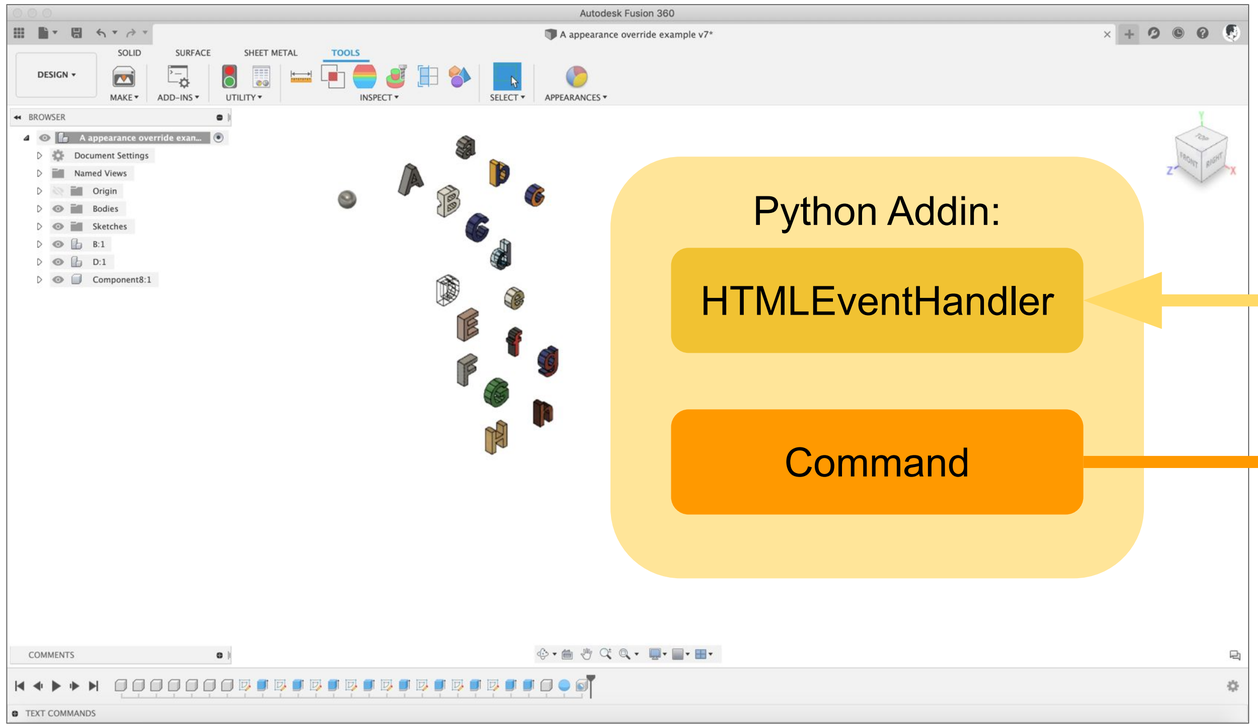
- jquery + jstree (above)
- react + material-ui + material-table (left)

Connect Directly to a web server:

- Insert components from a catalog
- Synchronize data

Loads an html page in a frame in Fusion 360

Can send and receive information from the page.



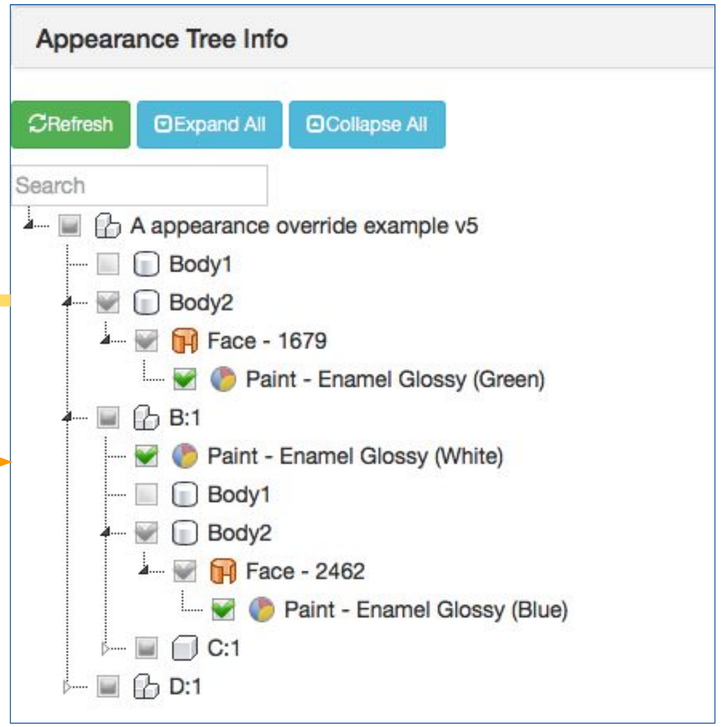
Python Addin:

HTMLEventHandler

Command

Javascript in Palette sends info
Python event handler receives it

Python Command sends info
JS event handler receives it



Palettes - Python/Add-in Side

```python
def on_html_event(self, html_args: adsk.core.HTMLEventArgs):
    ao = AppObjects()

    if html_args.action == "check_node":
        data = json.loads(html_args.data)
        adjust_material(data["node_id"], data["remove_material"], data["node_type"])

    if html_args.action == "refresh_tree":
        palette = ao.ui.palettes.itemById(self.palette_id)

        if palette:
            return_json = build_data()
            palette.sendInfoToHTML('tree_refresh', dumps(return_json))
```

This function is run when the Fusion 360 add-in recieves an event from the palette.

JSON data can be sent as a string from the corresponding javascript sending event.

Fusion can then use that data to perform some task.

```python
# When the command is clicked it will send this message to the HTML Palette
def on_execute(self, command, command_inputs, args, input_values):
    ao = AppObjects()
    palette = ao.ui.palettes.itemById(self.palette_id)
    if palette:
        return_json = build_data()
        palette.sendInfoToHTML('tree_update', dumps(return_json))
```

This is a function run by a regular command on a Fusion 360 toolbar.

Clicking the button sends an event to the palette’s javascript handler function.

Palettes - Javascript/HTML Side

JQuery, JSTree, Bootstrap

```javascript
function refresh_tree() {
    const args = {arg1: "Refresh Tree"};
    adsk.fusionSendData(
        'refresh_tree',
        JSON.stringify(args)
    )
}
```

This function is called when the user hits refresh button on web page.

Fusion is sent the json data: `args`

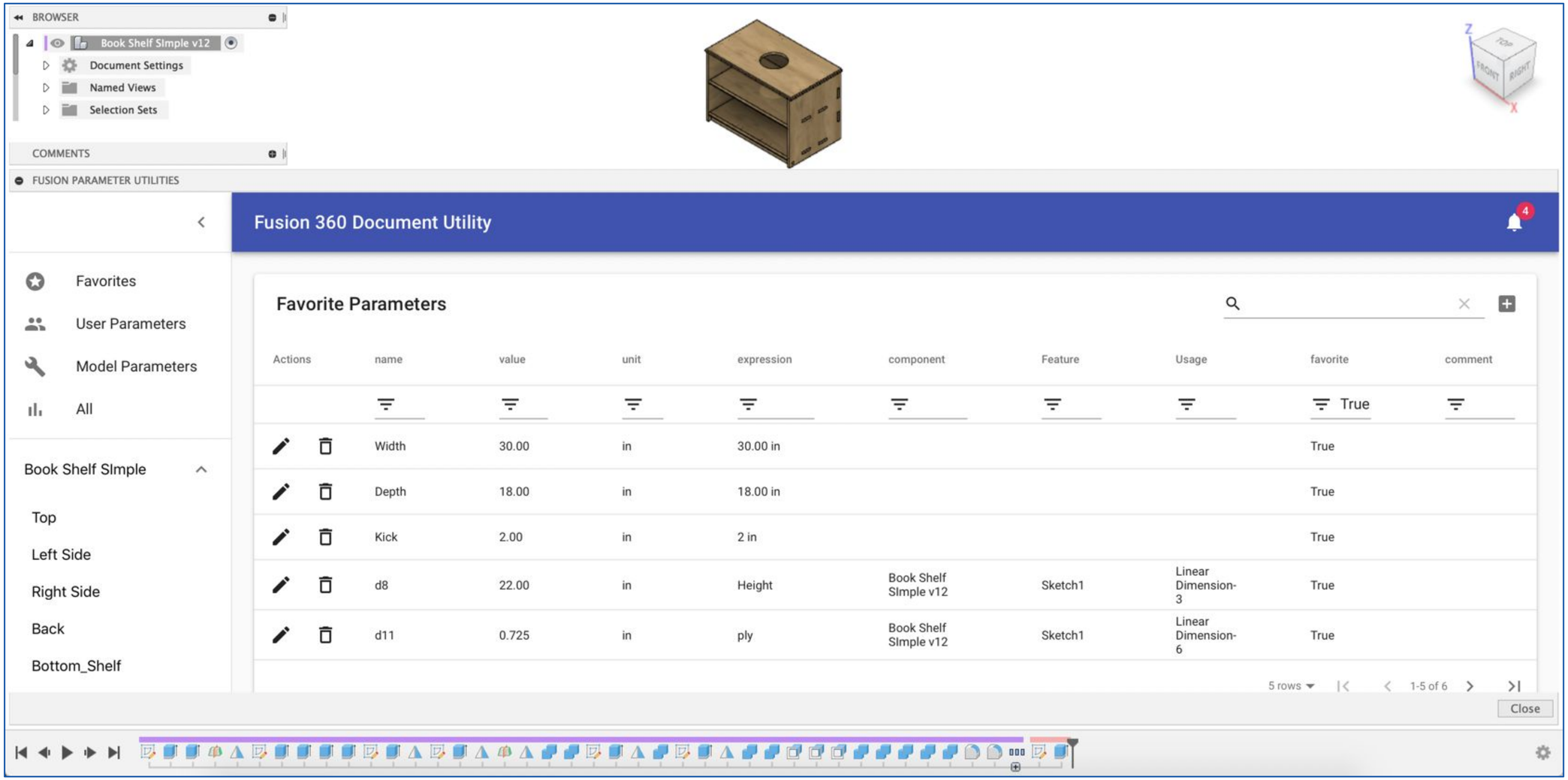
When a user hits a checkbox in the tree a different function with different args is called

```javascript
window.fusionJavaScriptHandler = {
    handle: function (action, data) {
        try {
            if (action === 'send') {
                // Update a paragraph with the data passed in.
                document.getElementById('message').innerHTML = data;
            } else if (action === 'tree_refresh') {
                if (!data || data.length === 0) return;
                let data_in = JSON.parse(data);
                $('#jstree_demo_div').jstree(true).settings.core.data = data_in.core;
                $('#jstree_demo_div').jstree(true).refresh();
```

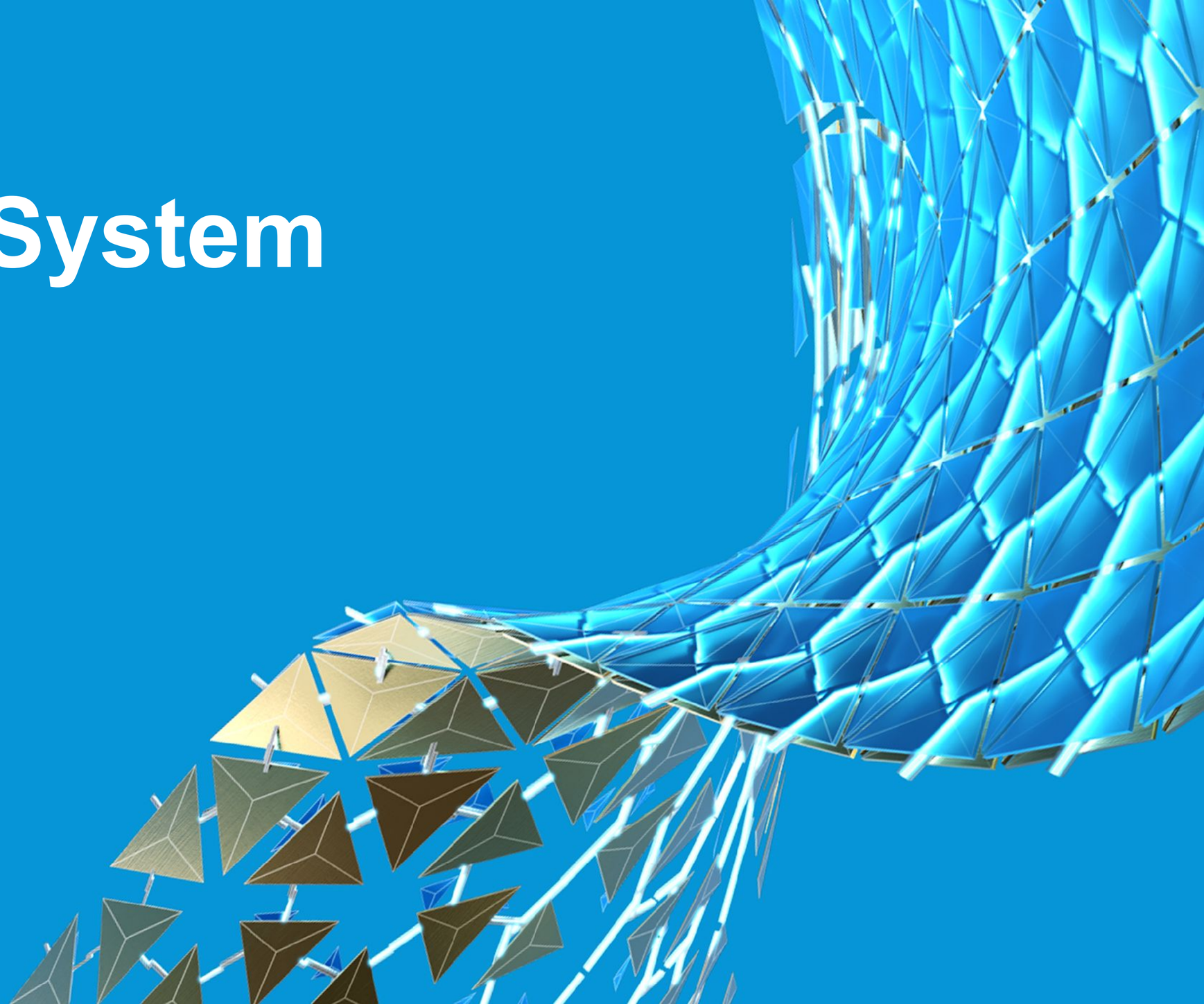
This is an event handler setup to listen for incoming messages from Fusion 360

When a “tree_refresh” message arrives, the page uses JQuery to update the JSTree status

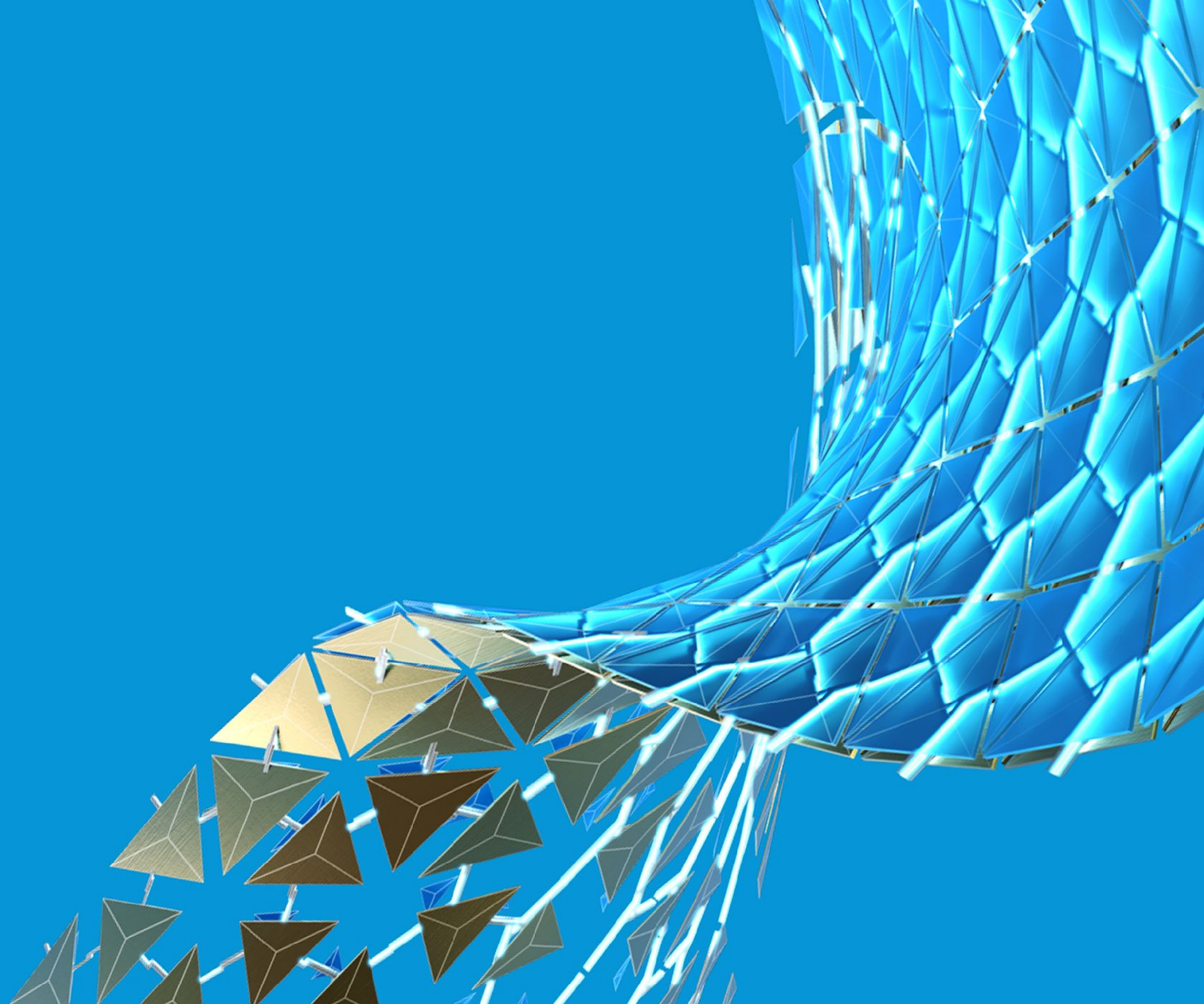
React Example



The Help System



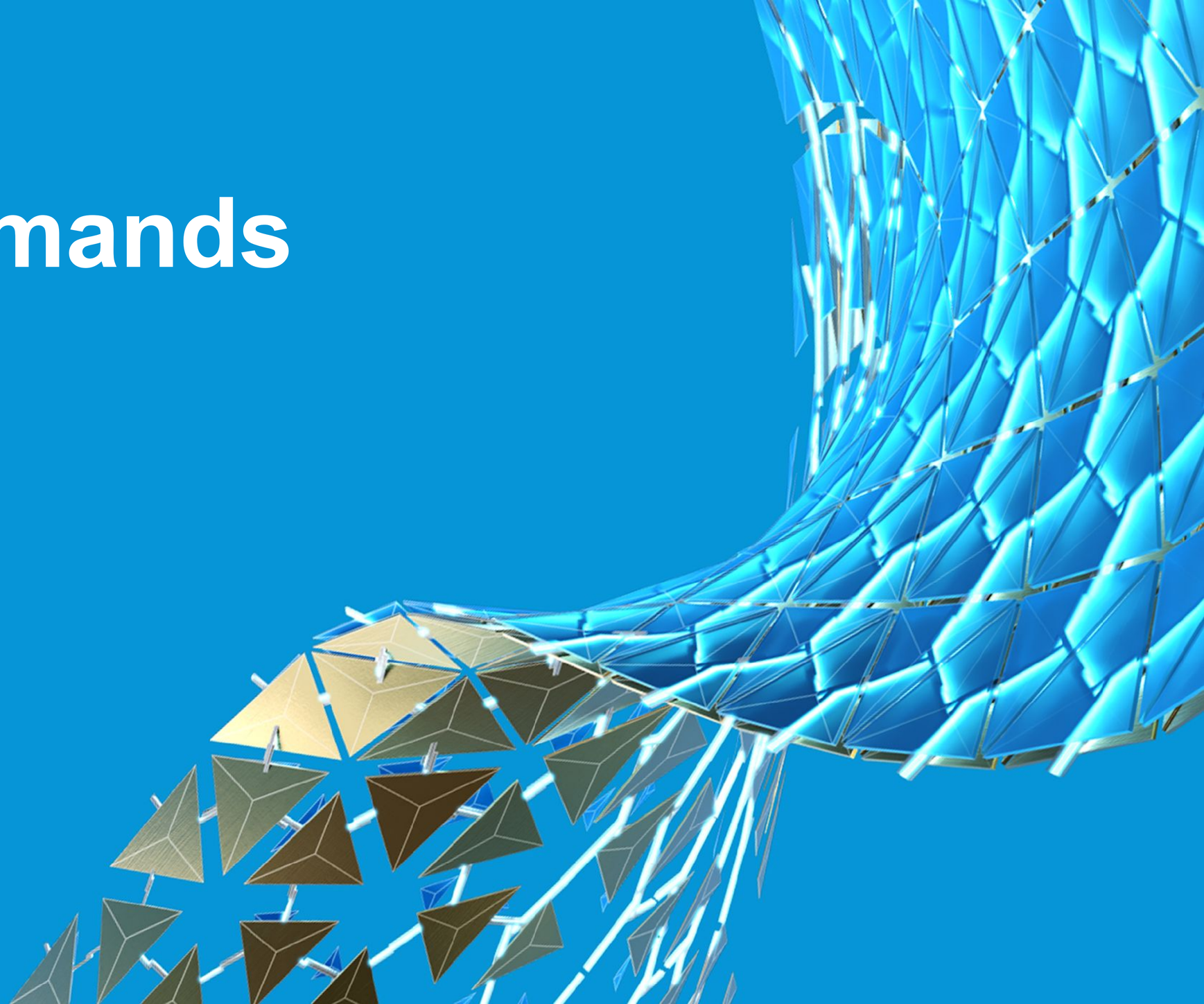
CAM API



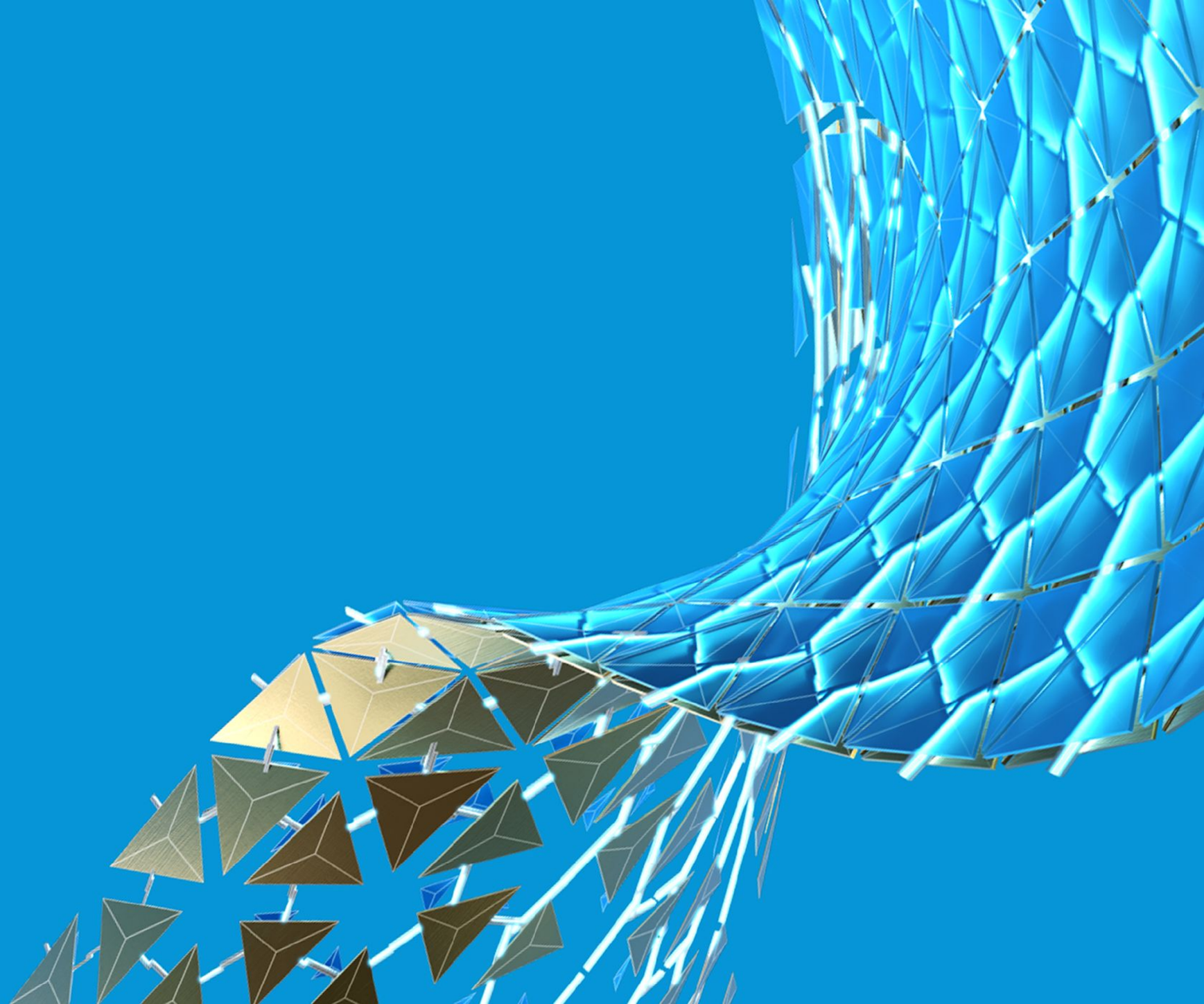
CAM API Overview

- Creating Operations from Templates
- Changing Parameters on Operations
- Updating post Properties and post processing

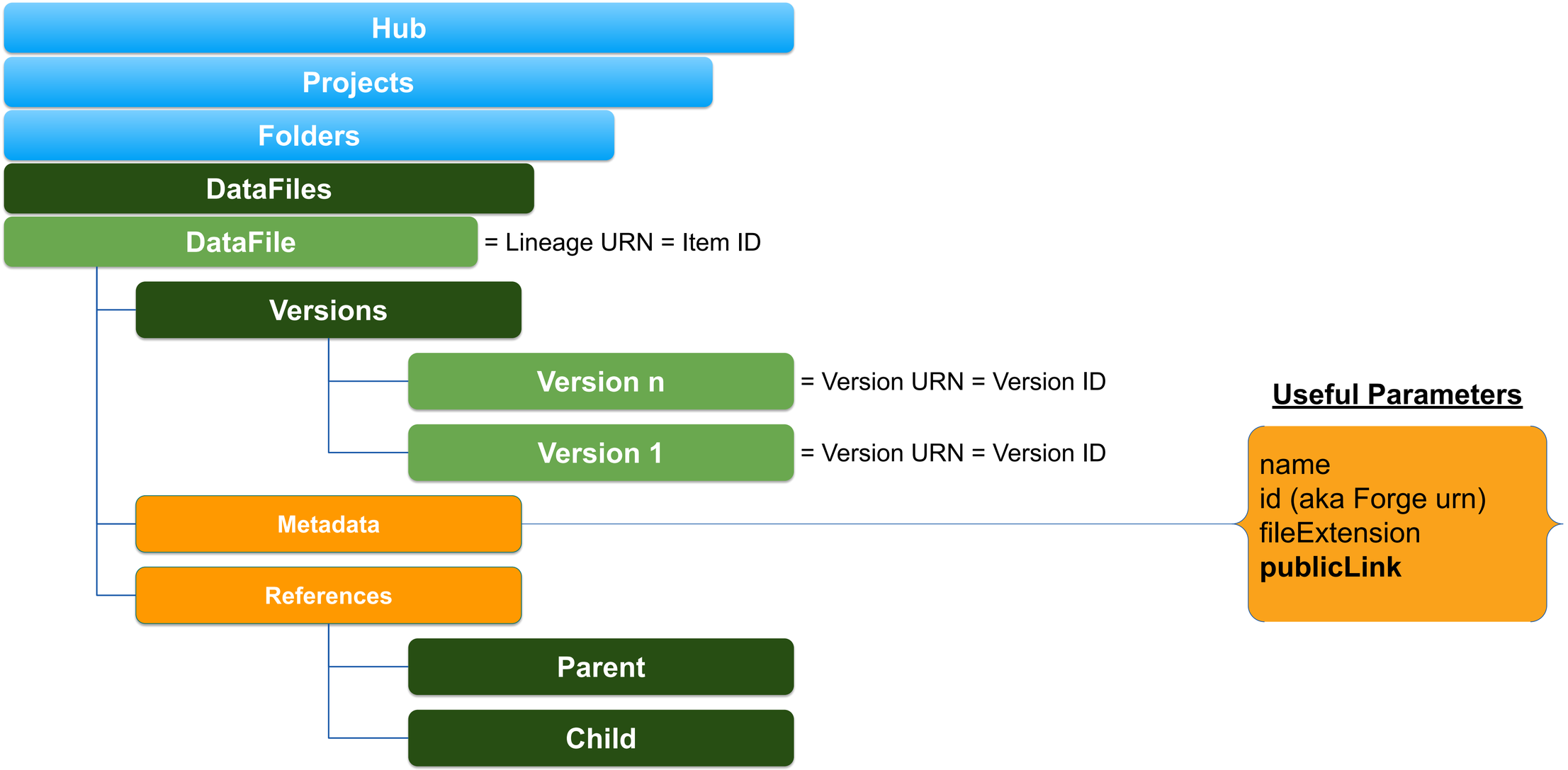
Text Commands



Data ID’s



Interrogate and Manipulate Fusion 360 Data



Data ID’s

Hub ID:	a.YnVzaW5lc3M6YXV0b2Rlc2szMDA4
Hub Name:	tapnair
Hub ID Decoded:	business:autodesk3008
Project ID:	a.YnVzaW5lc3M6YXV0b2Rlc2szMDA4IzIwMTcxMTI5MTA2MzAxMjgx
Project Name:	API Projects
Project ID Decoded:	business:autodesk3008#20171129106301281
Folder ID:	urn:adsk.wipprod:fs.folder:co.BbyeQ_9sS5quJLMo-Iju7Q
Folder Name:	API Projects
base64 Folder ID:	dXJuOmFkc2sud2lwcHJvZDpmcy5mb2xkZXI6Y28uQmJ5ZVFfOXNTNXF1SkxNby1JanU3UQ
Lineage URN:	urn:adsk.wipprod:dm.lineage:bYxmqGzrTVaOs5CowJXDnQ
Lineage Name:	Bearing Mount
base64 Lineage URN:	dXJuOmFkc2sud2lwcHJvZDpkbS5saW5lYWdlOmJZeG1xR3pyVFZhT3M1Q293SlhEblE

Data Item Lookups in a collection

DataHubs.itemById(hub_id)

DataProjects.itemById(project_id)

DataFolders.itemById(folder_id)

DataFiles.itemById(file_id)

Data Item ID’s

DataHub.id

DataProject.id

DataFolder.id

DataFile.id

DataFile.versionId

Global File Search

Data.findFileById(file_id)

FIND BY ID

Lineage URN:	urn:adsk.wipprod:dm.lineage:bYxmqGzrTVaOs5CowJXDnQ
Lineage Name:	Bearing Mount
base64 Lineage URN:	dXJuOmFkc2sud2lwcHJvZDpkbS5saW5lYWdlOmJZeG1xR3pyVFZhT3M1Q293SlhEblE

OK Cancel

Securing Python With MASSIF

The process developers must go through to securely license, protect, sell, then distribute their work, has historically been daunting, costly and complex.

Protecting Your Knowledge and Time

EASY TO USE

The Massif platform is built using the tools developers already use. We strive to automate as many of the tasks as possible.

EMPOWERING

Massif enables software developers to fairly sell, protect, and distribute their work, while seamlessly supporting their customers.

CONNECTIVITY

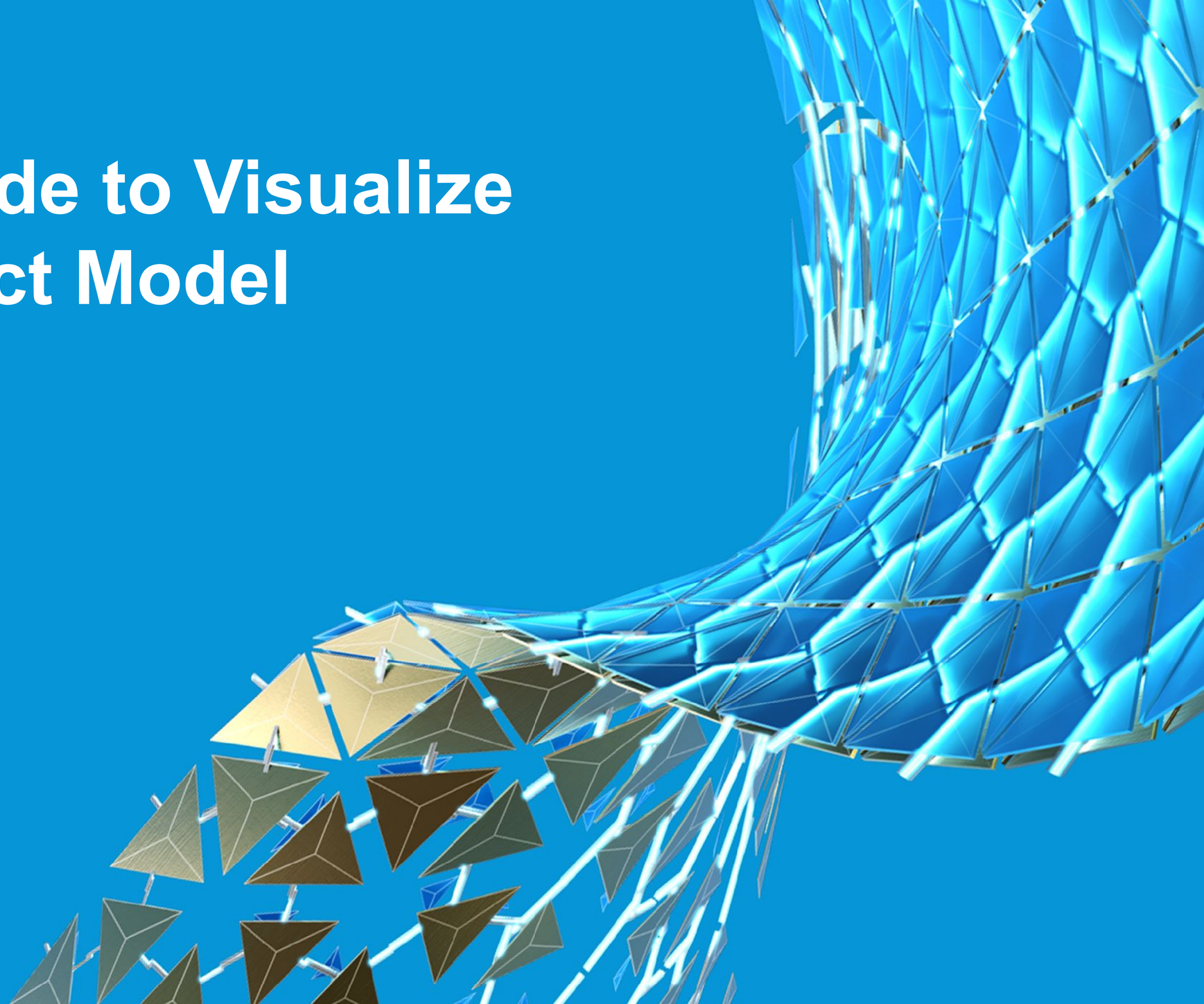
We bring users closer to the developers of the tools they love and deliver the tools developers love to create to more customers than ever before.

EXPERIENTIAL

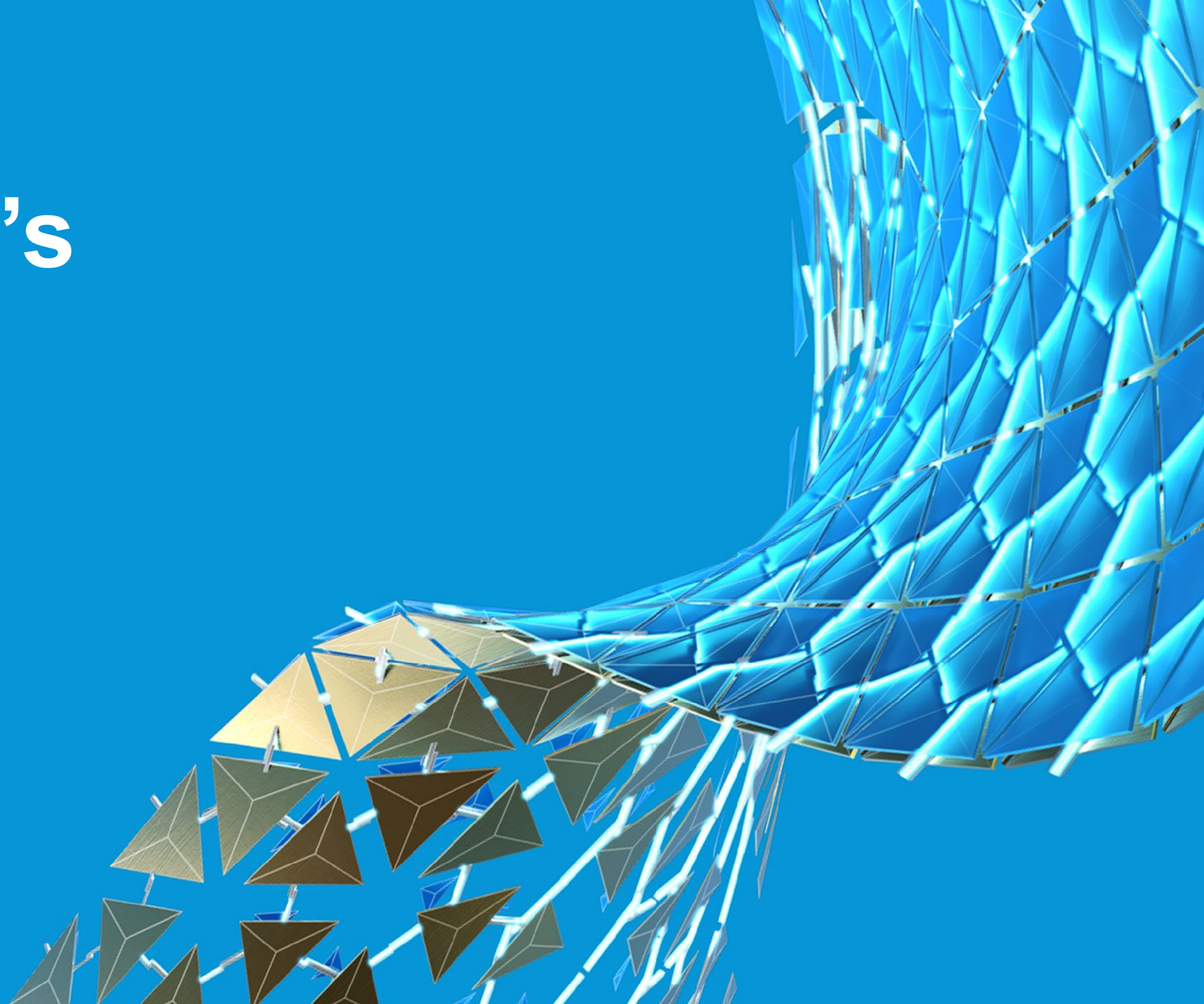
Massif exists to reduce friction in the software economy.

MASSIF.DEV

Using VS Code to Visualize the API Object Model



Forge API’s



Connecting to Forge API's

REST API's to work with data in a Fusion 360 Hub

Authorization… [read the docs](#)…

Leverage the data ID's obtained from Fusion 360 API:

- **DataHub.id**
- **DataProject.id**
- **DataFolder.id**
- **DataFile.id**
- **DataFile.versionId**

Use [Postman](#) to check your API calls!