GDC
CADD

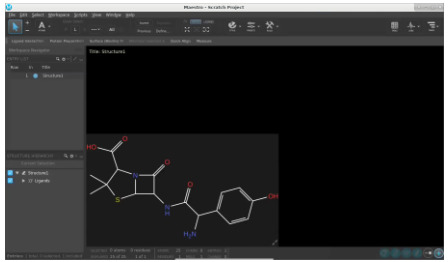
RDKit Office hours

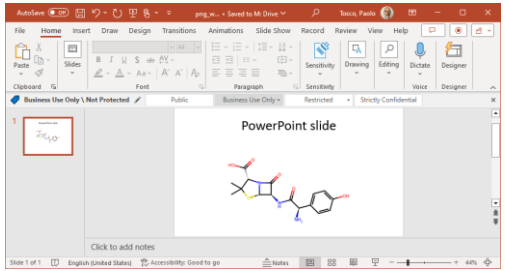
Paolo Tosco
12th RDKit UGM, Mainz
September 20, 2023

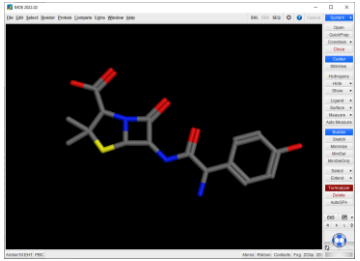
NOVARTIS | Reimagining Medicine

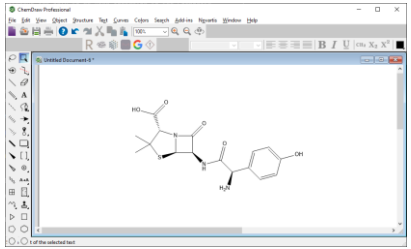
How nice would it be…

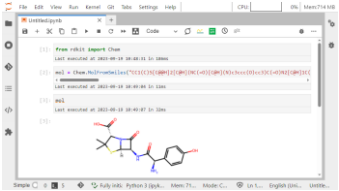
To be able to copy molecules from Office documents as rich molecule objects (*i.e.*, image + metadata) using only RDKit?

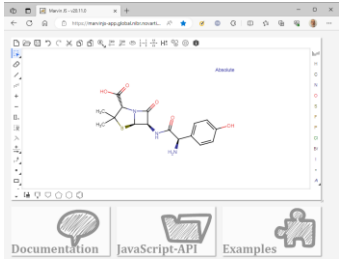






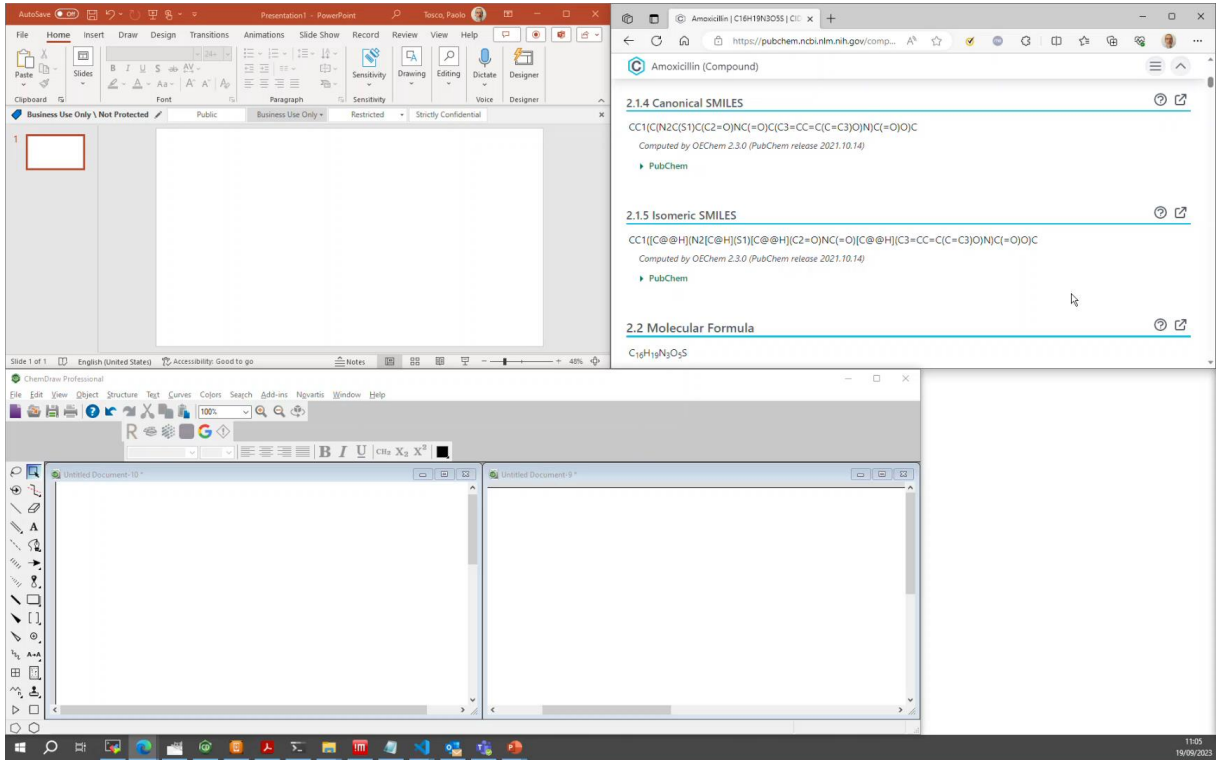






Good OLE objects

Currently this is possible using proprietary applications (e.g., ChemDraw, BIOVIA Draw...) by leveraging OLE objects

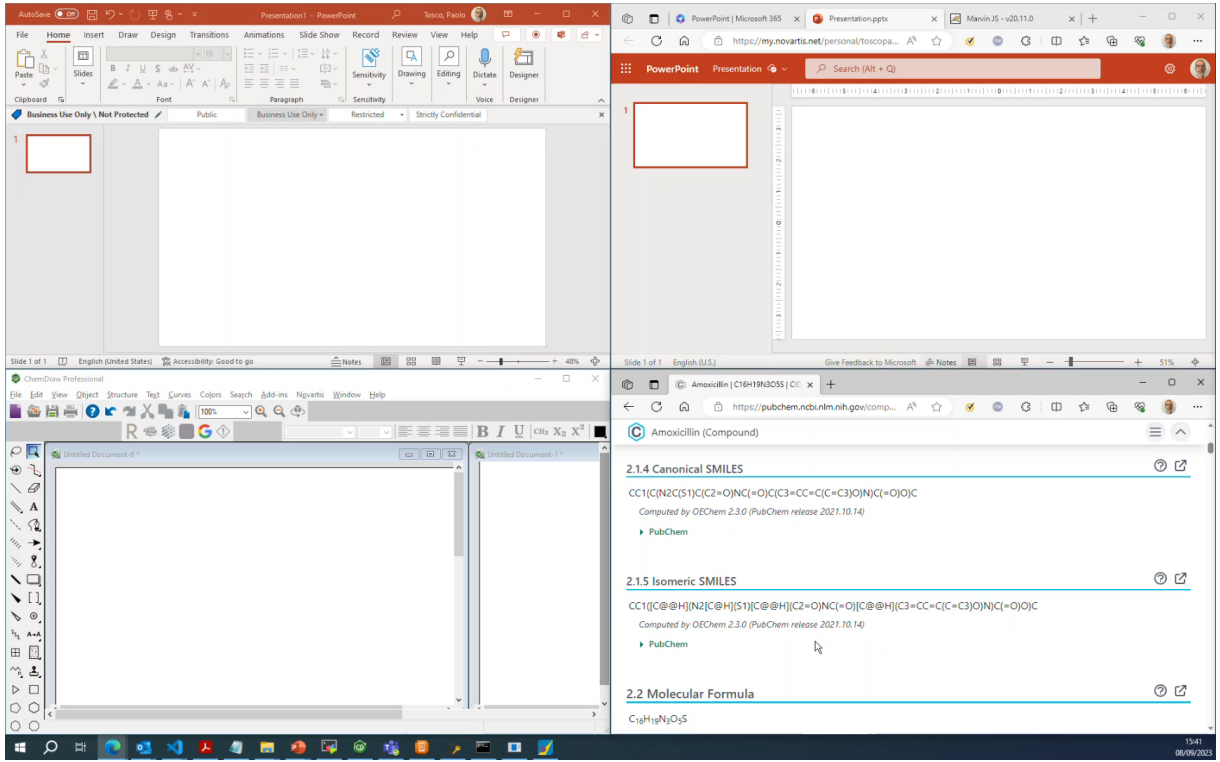


NOVARTIS | Reimagining Medicine

Not so good OLE objects

OLE objects are legacy binary blobs which:

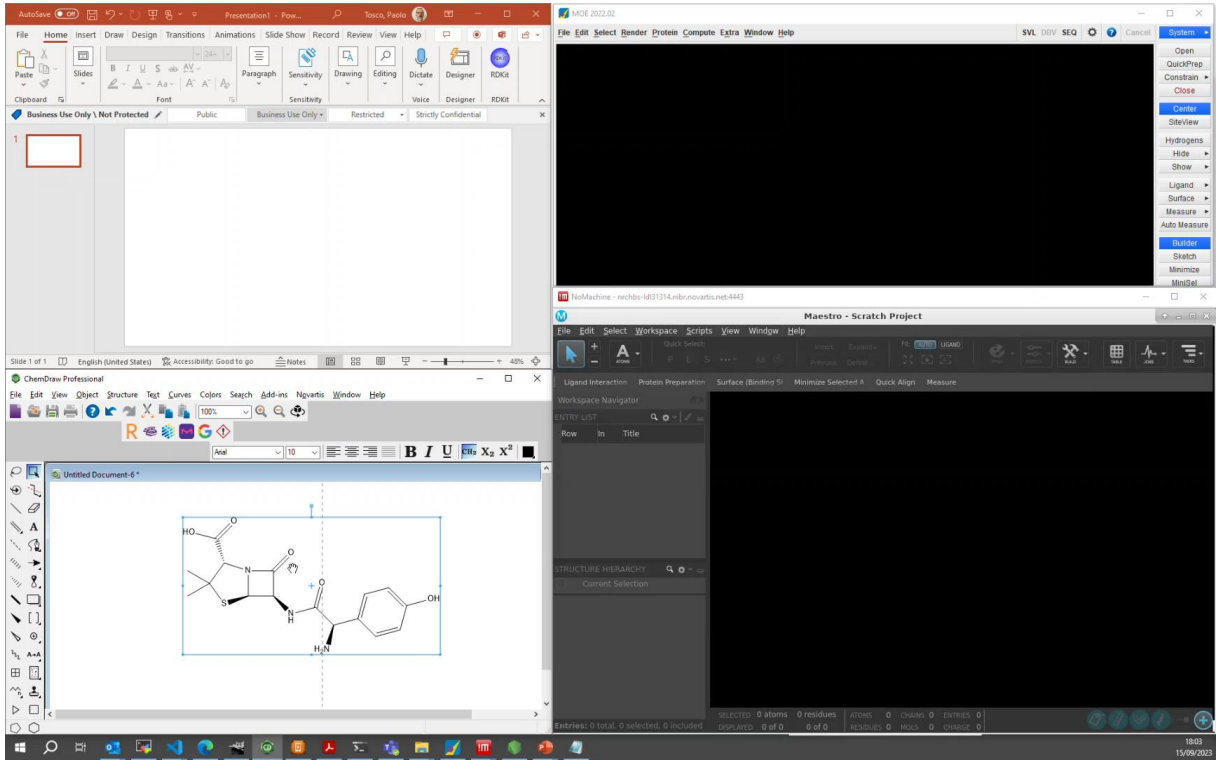
- Are hard to create programmatically
- Depend on the creator application
- Do not work in web apps (Microsoft 365, MarvinJS)



Not good at all OLE objects

OLE objects can only be exchanged between OLE-enabled desktop apps

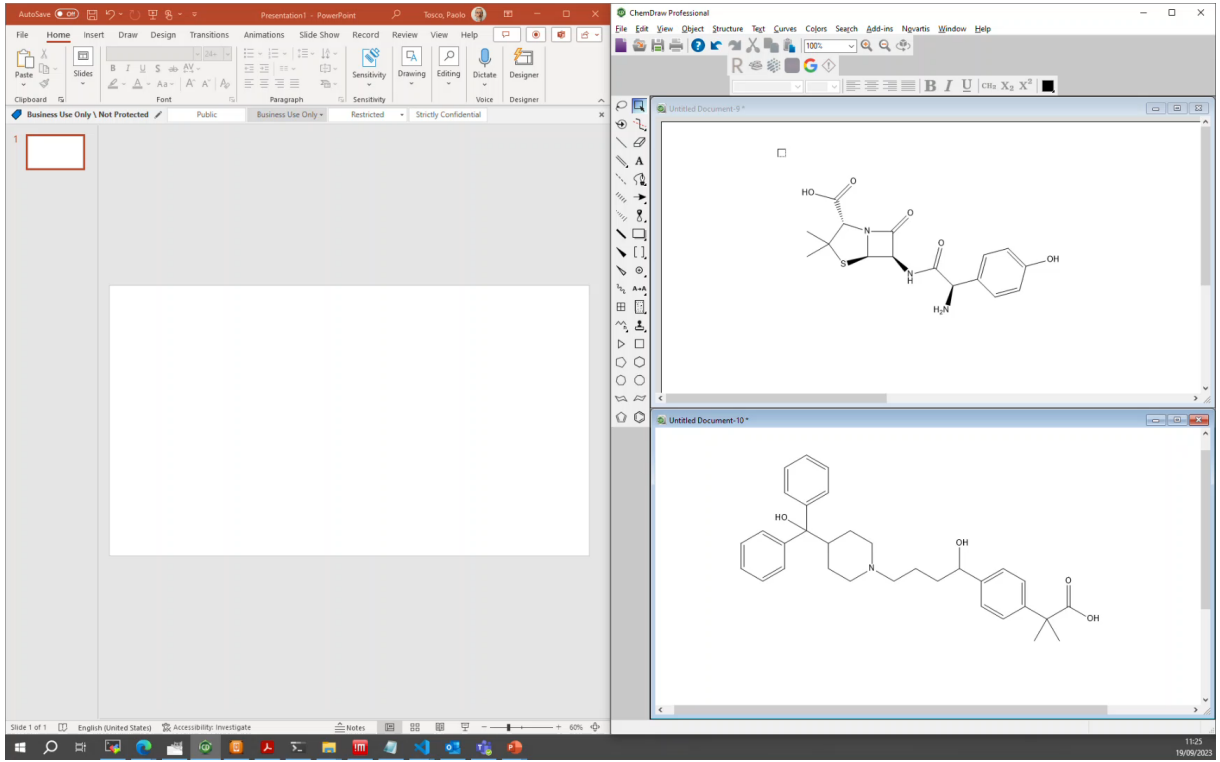
- Cannot directly paste from PowerPoint to MOE, Maestro or MarvinJS, only *via* ChemDraw



Let's forget about OLE objects

Copying multiple, separate molecules from PowerPoint to ChemDraw requires

- Either individual selections
- Or a single, multi-molecule object from the very beginning

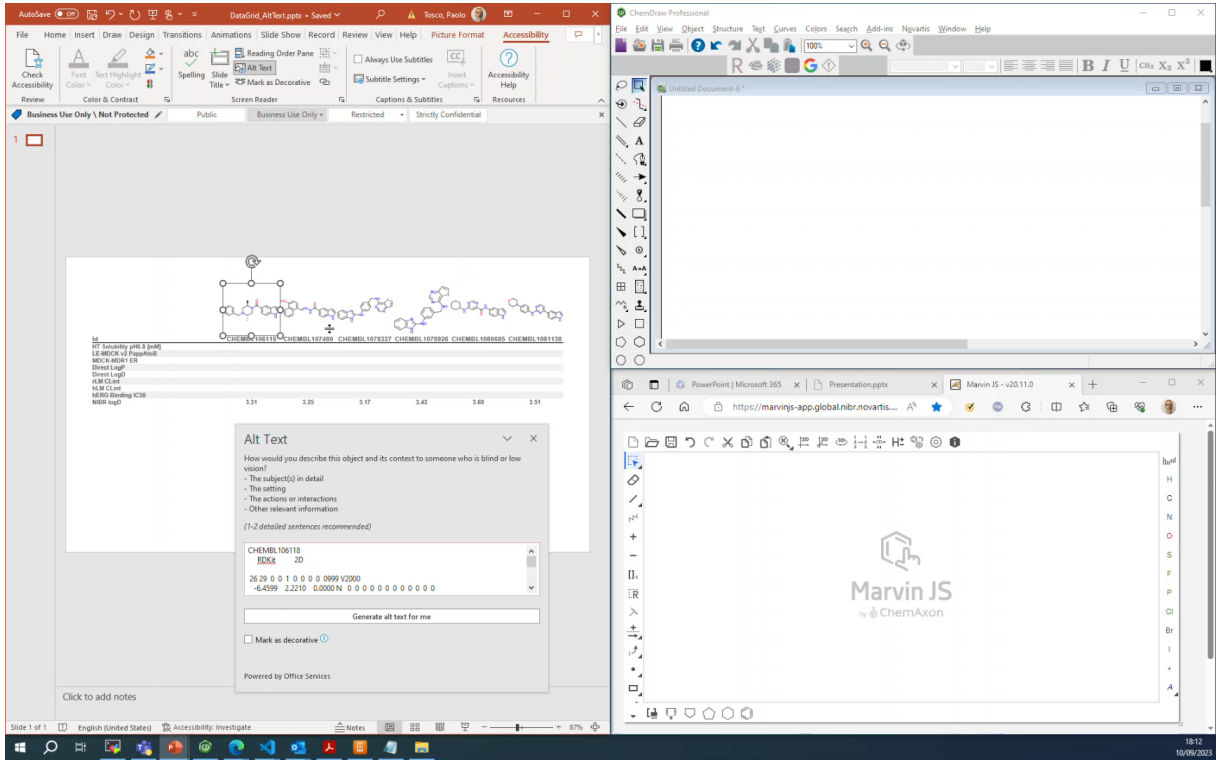


NOVARTIS | Reimagining Medicine

Poor man's image metadata

In internal web apps I abused the `Alt Text` field of the molecule's image to store the CTAB

- Not obvious
- Copying multiple molecules is cumbersome



PNG natively supports metadata

- The PNG format natively supports storing metadata in the PNG blob
- Metadata can be text or binary, compressed or uncompressed
- By default, RDKit writes molecule metadata to PNG images, and the RDKit API allows convenient access to the PNG metadata

```
[1]: import re
     import pptx
     from pptx.util import Inches
     from io import BytesIO
     import zipfile
     import json
     import base64
     from rdkit import Chem
     from rdkit.Chem.Draw import rdDepictor, rdMolDraw2D
     from IPython.display import Image
     Last executed at 2023-09-17 17:35:55 in 1.78s

[2]: amoxicillin = Chem.MolFromSmiles("CC1([C@@H](N2[C@H](S1)[C@@H](C2=O)NC(=O)[C@@H](C3=CC=C(C=C3)O)N)C(=O)O)C")
     Last executed at 2023-09-17 17:35:55 in 9ms

[3]: rdDepictor.SetPreferCoordGen(True)
     Last executed at 2023-09-17 17:35:56 in 6ms

[4]: rdDepictor.Compute2DCoords(amoxicillin)
     Last executed at 2023-09-17 17:35:56 in 69ms

[4]: 0
```

By default, RDKit writes molecule metadata (SMILES, CTAB and pickled mol) to molecule PNG strings:

```
[5]: drawer = rdMolDraw2D.MolDraw2DCairo(300, 200)
     Last executed at 2023-09-17 17:35:56 in 8ms

[6]: drawer.DrawMolecule(amoxicillin)
     drawer.FinishDrawing()
     Last executed at 2023-09-17 17:35:56 in 13ms

[7]: png = drawer.GetDrawingText()
     Last executed at 2023-09-17 17:35:57 in 18ms
```

RDKit can deal with PNG metadata

RDKit stores molecule metadata into the PNG image in different formats:

- Pickled RDKit molecule
- CXSMILES
- CTAB (optional)

Therefore, can't we just attach molecule metadata to a PNG image?

RDKIt's Python API has convenient functions to access PNG metadata as a `dict`:

[8]:
```
png_metadata = Chem.MetadataFromPNGString(png)
```
Last executed at 2023-09-17 17:35:58 in 7ms

[9]:
```
print("\n".join(png_metadata.keys()))
```
Last executed at 2023-09-17 17:35:58 in 31ms

```
rdkitPKL rdkit 2023.03.2
MOL rdkit 2023.03.2
SMILES rdkit 2023.03.2
```

Here's amoxicillin CXSMILES with 2D coordinates; the `MOL` and `rdkitPKL` fields contain the CTAB and the pickled `Mol` object, respectively.

[10]:
```
print(png_metadata["SMILES rdkit 2023.03.2"].decode("utf-8"))
```
Last executed at 2023-09-17 17:35:58 in 7ms

```
[H][C@]12SC(C)(C)[C@H](C(=O)O)N1C(=O)[C@H]2NC(=O)[C@H](N)c1ccc(O)cc1 |(0.732924,-0.708038,;1.1392,0.205712,;1.97
86,-0.339288,;2.754,0.292912,;3.301,-0.545288,;3.645,0.748112,;2.3954,1.22631,;2.9378,2.06511,;2.483,2.95551,;3.
9364,2.01371,;1.3938,1.17111,;0.4284,1.43051,;-0.0723999,2.29491,;0.1728,0.463712,;-0.6934,-0.038088,;-1.56,0.46
0512,;-1.5616,1.46051,;-2.4254,-0.0406879,;-3.2922,0.458112,;-2.424,-1.03989,;-1.558,-1.53929,;-1.5562,-2.5392
9,;-2.4224,-3.03989,;-2.421,-4.04089,;-3.2894,-2.54209,;-3.2894,-1.54209,),wU:1.0,6.6,17.19wD:13.15|
```

The size of the PNG with metadata is 10898 bytes; let's note down this number.

[11]:
```
len(png)
```
Last executed at 2023-09-17 17:35:58 in 9ms

[11]: 10898

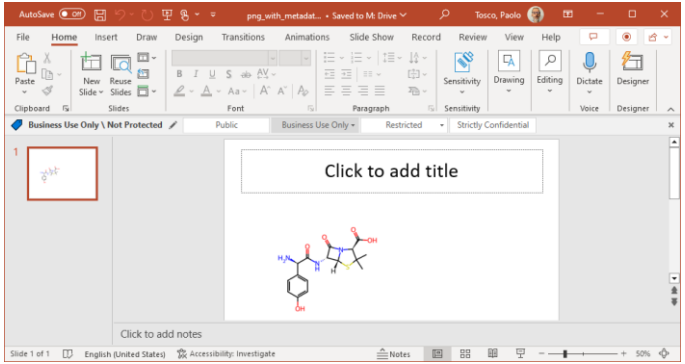
Embed RDKit PNG into PowerPoint

- We can indeed generate a PowerPoint presentation with a PNG image containing RDKit molecule metadata using the Python `pptx` module

- When we open the presentation in PowerPoint, the PNG image is there:

We can programmatically generate a PowerPoint presentation containing the PNG image of amoxicillin *and* its metadata:

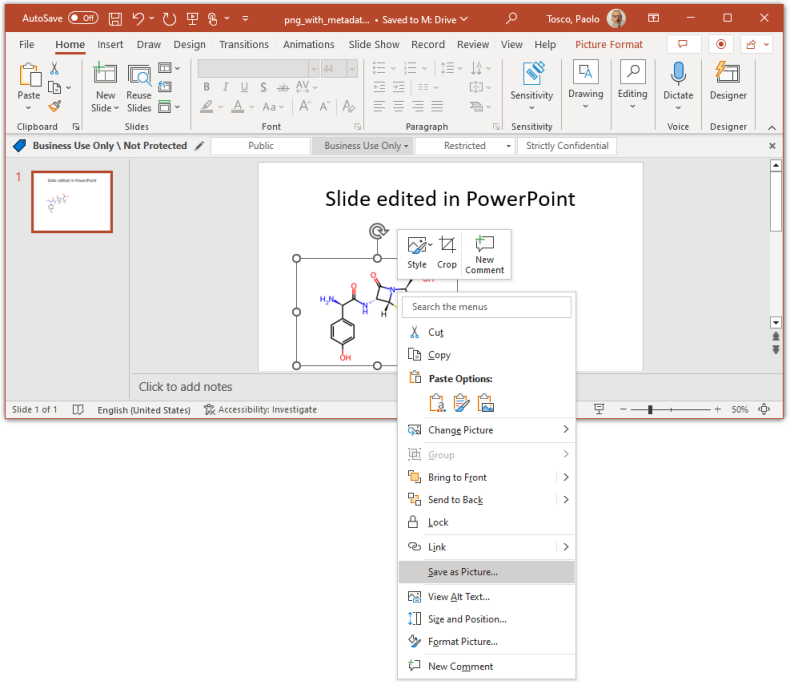
```
[12]: ppt_pres = pptx.Presentation()
Last executed at 2023-09-17 17:35:59 in 31ms

[13]: slide_layout = ppt_pres.slide_layouts[5]
      slide = ppt_pres.slides.add_slide(slide_layout)
Last executed at 2023-09-17 17:35:59 in 13ms

[14]: with BytesIO(png) as hnd:
          png_image = slide.shapes.add_picture(hnd, Inches(1), Inches(2.5))
Last executed at 2023-09-17 17:35:59 in 40ms

[15]: ppt_pres.save("/home/toscopa1/Documents/RDKit_UGM_2023/png_with_metadata.pptx")
Last executed at 2023-09-17 17:36:00 in 45ms
```



Export PNG image from PowerPoint

To test if the molecule metadata is retained and exported by PowerPoint, I did two experiments:

1. I saved the molecule image as a standalone PNG image
2. I edited the slide title and saved the presentation



Microsoft Office strips metadata upon exporting to PNG file…

- Unfortunately, Microsoft Office mangles the original PNG image on export
- in particular, all metadata is stripped off

However, if we edit that presentation in PowerPoint and export the image, we will find that the metadata is stripped:

[16]:
```
with open("/home/toscopa1/Documents/RDKit_UGM_2023/png_saved_from_ppt.png", "rb") as hnd:
    png_saved_from_ppt = hnd.read()
```
Last executed at 2023-09-17 17:36:00 in 6ms

We can immediately see that something has changed in the image as its size is now different:

[17]:
```
len(png_saved_from_ppt)
```
Last executed at 2023-09-17 17:36:00 in 10ms

[17]: 18712

What is worse, metadata has been stripped:

[18]:
```
png_saved_from_ppt_metadata = Chem.MetadataFromPNGString(png_saved_from_ppt)
```
Last executed at 2023-09-17 17:36:03 in 7ms

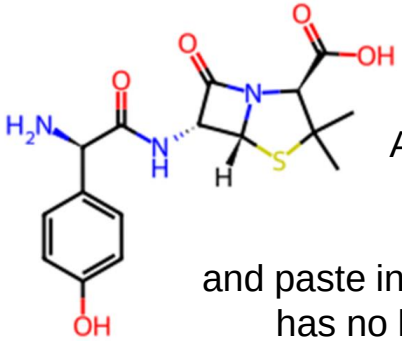
[19]:
```
print("\n".join(png_saved_from_ppt_metadata.keys()))
```
Last executed at 2023-09-17 17:36:03 in 8ms

[20]:
```
png_saved_from_ppt_metadata
```
Last executed at 2023-09-17 17:36:03 in 10ms

[20]: {}

...as well as to clipboard

If we export the image through clipboard to a Jupyter cell and analyze the exported PNG image, again we find out that the metadata has been stripped off:



Attempting to copy to clipboard and paste into Jupyter has no luck either

The PNG image is much larger in size, but metadata has gone lost

If we extract the raw `base64` metadata from the Jupyter notebook, we will find out the the image has been bloated further, but no metadata is available:

```
[21]: with open("/home/toscopa1/ipynb/OfficeStripsPNGMetaData.ipynb") as hnd:
          ipynb = json.load(hnd)

[22]: png_attachments = [cell["attachments"] for cell in ipynb["cells"]
                         if cell["cell_type"] == "markdown" and "attachments" in cell
                         and any("png" in k for k in cell["attachments"].keys())]

[23]: assert png_attachments and len(png_attachments) == 1

[24]: png_as_base64 = [v["image/png"] for v in png_attachments[0].values()]

[25]: assert png_as_base64 and len(png_as_base64) == 1

[26]: png_from_clipboard = base64.b64decode(png_as_base64[0])
```

The image size is now 42686 bytes, but no metadata is present:

```
[27]: len(png_from_clipboard)

[27]: 42686

[28]: png_from_clipboard_metadata = Chem.MetadataFromPNGString(png_from_clipboard)

[29]: print("\n".join(png_from_clipboard_metadata.keys()))

[30]: png_from_clipboard_metadata
      Last executed at 2023-09-11 10:37:58 in 10ms

[30]: {}
```

A ray of light

- Molecule metadata is not stripped altogether from the presentation
- PowerPoint preserves the metadata in the file
- It only strips it upon exporting the PNG

However, there's still a ray of light: molecule metadata is *not* stripped from the presentation.
If we open the edited presentation again with `pptx` in Jupyter, we can verify that the metadata is still there.
This means that PowerPoint preserves the metadata in the file, but strips it upon exporting the PNG to clipboard or to a file.

[31]:
```
ppt_pres = pptx.Presentation("/home/toscopa1/Documents/RDKit_UGM_2023/png_with_metadata_edited.pptx")
```
Last executed at 2023-09-17 17:36:15 in 34ms

[32]:
```
len(ppt_pres.slides[0].shapes)
```
Last executed at 2023-09-17 17:44:08 in 12ms

[32]:
```
2
```

[33]:
```
ppt_pres.slides[0].shapes[0].text
```
Last executed at 2023-09-17 17:44:09 in 11ms

[33]:
```
'Slide edited in PowerPoint'
```

[34]:
```
png_image_from_ppt_file = ppt_pres.slides[0].shapes[1].image.blob
```
Last executed at 2023-09-17 17:44:09 in 8ms

The PNG image size has not changed, and metadata is still there:

[35]:
```
len(png_image_from_ppt_file)
```
Last executed at 2023-09-17 17:44:09 in 24ms

[35]:
```
10898
```

[36]:
```
png_image_from_ppt_file_metadata = Chem.MetadataFromPNGString(png_image_from_ppt_file)
```
Last executed at 2023-09-17 17:44:12 in 8ms

[37]:
```
print("\n".join(png_image_from_ppt_file_metadata.keys()))
```
Last executed at 2023-09-17 17:44:12 in 9ms

```
rdkitPKL rdkit 2023.03.2
MOL rdkit 2023.03.2
SMILES rdkit 2023.03.2
```

PPTX is a ZIP archive

- A `.pptx` file is just a `.zip` archive with a different extension
- It contains the various parts of the presentation, *i.e.* Open XML and attachments, including the *original* PNG file with its metadata

A `.pptx` file is just a ZIP file with a different extension, and we can see that the image file is there, unaltered, with the same size it had when we generated the `.pptx` file:

[38]:
```
pptx_as_zip = zipfile.ZipFile("/home/toscopa1/Documents/RDKit_UGM_2023/png_with_metadata_edited.pptx")
```
Last executed at 2023-09-17 17:46:42 in 9ms

[39]:
```
[name for name in pptx_as_zip.namelist() if "image" in name]
```
Last executed at 2023-09-17 17:46:42 in 19ms

[39]:
```
['ppt/media/image1.png']
```

[40]:
```
pptx_as_zip.getinfo("ppt/media/image1.png")
```
Last executed at 2023-09-17 17:46:43 in 10ms

[40]:
```
<ZipInfo filename='ppt/media/image1.png' file_size=10898>
```

An RDKit Office add-in?

- Old school VSTO add-ins only work on Windows desktop Office version
- No macOS support
- No web platform support (*i.e.*, Microsoft 365)
- DLL deployment to all clients is required

Create VSTO Add-ins for Office by using Visual Studio

Article • 04/30/2022 • 13 contributors

Feedback

In this article

In this section

Related sections

Applies to: Visual Studio Visual Studio for Mac Visual Studio Code

Important

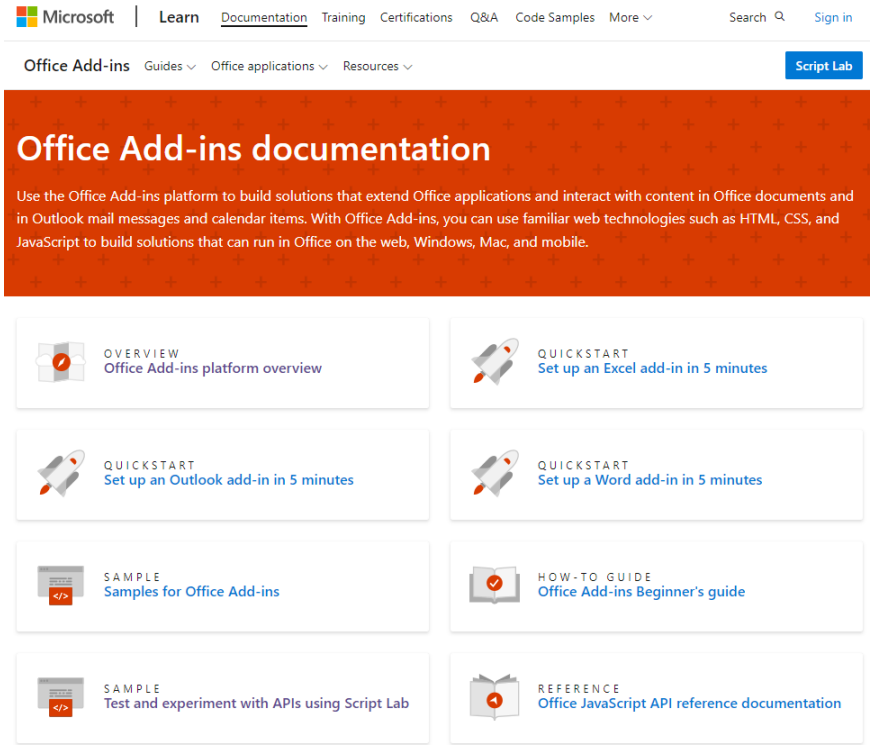
VSTO relies on the **.NET Framework**. COM add-ins can also be written with the .NET Framework. Office Add-ins cannot be created with **.NET Core and .NET 5+**, the latest versions of .NET. This is because .NET Core/.NET 5+ cannot work together with .NET Framework in the same process and may lead to add-in load failures. You can continue to use .NET Framework to write VSTO and COM add-ins for Office. Microsoft will not be updating VSTO or the COM add-in platform to use .NET Core or .NET 5+. You can take advantage of .NET Core and .NET 5+, including ASP.NET Core, to create the server side of **Office Web Add-ins**.

You can use the Microsoft Office developer tools in Visual Studio to create .NET Framework applications that extend Office. These applications are also named *Office solutions*.

The Office developer tools provide features that help you create Office solutions to suit a variety of business needs. The tools include project templates to help you create Office solutions by using Visual Basic or Visual C#, and visual designers that help you create custom user interfaces for your Office solutions.

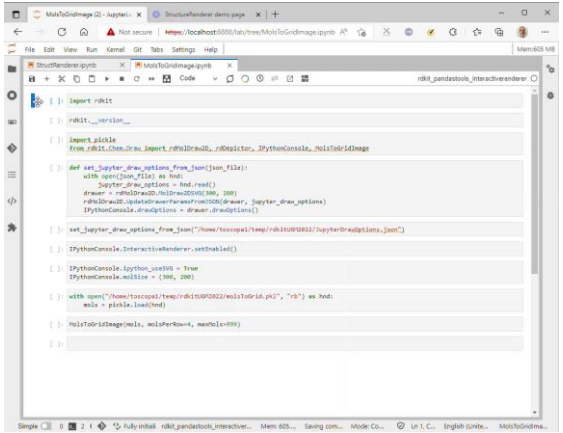
New JavaScript Office Add-ins

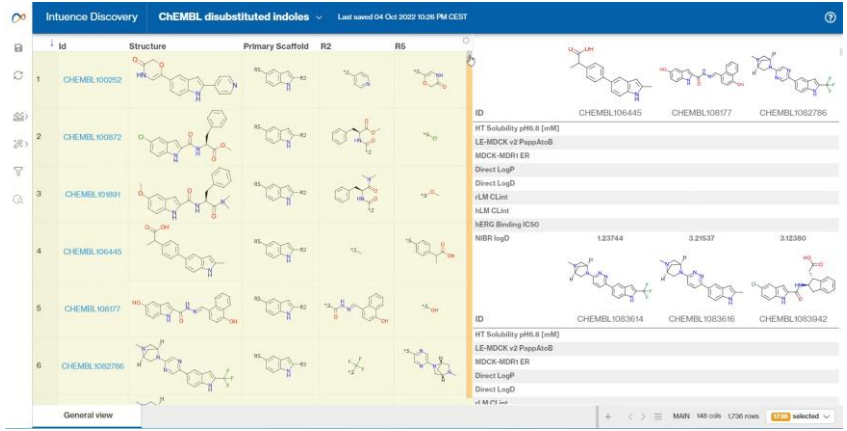
- API still being actively developed
- All platforms are supported (macOS and web)
- Easier add-in deployment through SharePoint repo



rdkit-structure-renderer

- npm package based on RDKitJS MinimalLib
- Vanilla JS package





- No dependencies on JS frameworks such as React, Angular, etc.
- Interactive rendering of 2D structures in web apps and Jupyter notebooks

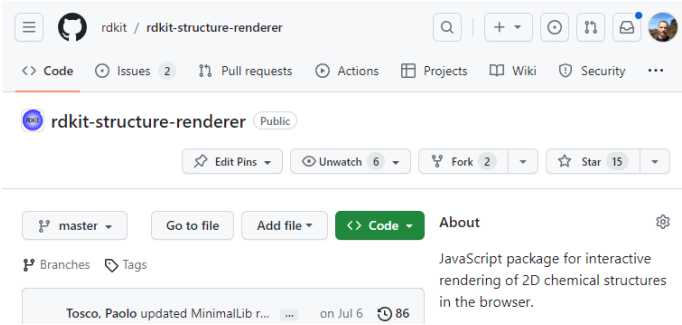
Implemented PNG metadata handling API into MinimalLib

- `rdkit-structure-renderer` already supports generating PNG images from molecules through the HTML5 `canvas` API
- What is missing is an API to embed molecule metadata into PNG images
- I added the following functions/methods to the MinimalLib JS API:
  - `get_mol_from_png_blob(png_blob, details_json)`
  - `get_mols_from_png_blob(png_blob, details_json)`
  - `JSMol.add_to_png_blob(png_blob, details_json)`
  - `JSMol.get_coords()`
  - `JSMol.combine_with(other)`
- I also added the same functionality to the MinimalLib CFFI API

rdkit-office-addin

```
PPTX_Presentation
├── [Content_Types].xml
├── docProps
│   ├── app.xml
│   └── core.xml
├── ppt
│   ├── charts
│   │   └── _rels
│   ├── embeddings
│   ├── media
│   │   ├── image1.png
│   │   └── image2.png
│   ├── notesMasters
│   │   ├── notesMaster1.xml
│   │   └── _rels
│   │       └── notesMaster1.xml.rels
│   ├── notesSlides
│   │   ├── notesSlide1.xml
│   │   ├── notesSlide2.xml
│   │   └── _rels
│   │       ├── notesSlide1.xml.rels
│   │       └── notesSlide2.xml.rels
│   ├── presentation.xml
│   ├── presProps.xml
│   ├── _rels
│   │   └── presentation.xml.rels
│   ├── slideLayouts
│   │   ├── _rels
│   │   │   └── slideLayout1.xml.rels
│   │   └── slideLayout1.xml
│   ├── slideMasters
│   │   ├── _rels
│   │   │   └── slideMaster1.xml.rels
│   │   └── slideMaster1.xml
│   ├── slides
│   │   ├── _rels
│   │   │   ├── slide1.xml.rels
│   │   │   └── slide2.xml.rels
│   │   ├── slide1.xml
│   │   └── slide2.xml
```

npm package depending on `rdkit-structure-renderer`



When a selection is to be copied to clipboard, the add-in looks for `/ppt/media/image###.png` objects in the Open XML tree

RDKit metadata are extracted and put on the clipboard as plain text by the JS Office add-in

```
function getOOXml(doc, platform);
async function getSelectedImages(slideId);
function getMolSizeAngstrom(xyzArray);
function extractMolFromPngBlob(rdkitModule, data);
function getMolAndCentroidFromImage(rdkitModule, {
left, top, width, height, data }, molToImageRatio);
async function getCombinedMolFromImages(images)
async function getSelectedImagesWithData(event);
async function getSmilesArrayFromImages(images);
async function onCopyCommon(event);
async function onCopySmiles(event, sep, term);
```

Open XML SDK

- Currently Microsoft only offers an Open XML SDK for .NET applications

About the Open XML SDK 2.5 for Office

Article • 03/05/2022 • 5 contributors

Feedback

Open XML is an open standard for word-processing documents, presentations, and spreadsheets that can be freely implemented by multiple applications on different platforms. Open XML is designed to faithfully represent existing word-processing documents, presentations, and spreadsheets that are encoded in binary formats defined by Microsoft Office applications. The reason for Open XML is simple: billions of documents now exist but, unfortunately, the information in those documents is tightly coupled with the programs that created them. The purpose of the Open XML standard is to de-couple documents created by Microsoft Office applications so that they can be manipulated by other applications independent of proprietary formats and without the loss of data.

System requirements

The Open XML SDK 2.5 has the following system requirements:

Supported operating systems: Windows 8 Preview, Windows 7, Windows Server 2003 Service Pack 2, Windows Server 2008 R2, Windows Server 2008 Service Pack 2, Windows Vista Service Pack 2, Windows XP Service Pack 3

System prerequisites: .NET Framework version 4.0, Up to 300 MB of available disk space

Open XML SDK for JavaScript

- Fortunately, there is a 10-year-old open-source JavaScript Open XML SDK by Eric White
- I found a GitHub version that can be `npm` installed
- I forked it and did some work to modernize it a bit
- I will publish it on GitHub

Eric White's Blog

Open XML, SharePoint, and Office

Open XML SDK for JavaScript

The Open XML SDK for JavaScript is a light-weight JavaScript API that enables you to create, modify, or query Open XML documents. It is useful in the following scenarios:

- Client-side Open XML applications that run in the browser
- Server-side applications using Node.js
- Windows 8 "Windows Store" applications
- Apps for Office Client 2013
- Apps for SharePoint 2013

openxml

openxmlsdkjs - Open XML SDK for JavaScript

The original author is Eric White. I just created a module in Node for it.

RDKit Office Add-In demo

ChEMBL disubstituted indoles

Not secure | https://grok.local.nibr.nova...

Intuence Discovery | ChEMBL disubstituted indoles | Analysis last saved one hour ago | Data last refreshed one hour ago

	Id	Structure
1	CHEMBL106118	
2	CHEMBL107480	
3	CHEMBL1080685	
4	CHEMBL108177	

ID
HT Solubility pH6.8 [mM]
LE-MDCK v2 PappAtoB
MDCK-MDR1 ER
Direct LogP
Direct LogD
rLM CLint
hLM CLint
hERG Binding IC50
NIBR logD

General view

MAIN 152 cols 750 rows SELECT

Marvin JS - v20.11.0 | Untitled.ipynb (2) - JupyterLab

https://marvinjs-app.global.nibr.novartis....

Marvin JS
by ChemAxon

ChemDraw Professional - [Untitled Document-8 *]

File Edit View Object Structure Text Curves Colors Search Add-ins Novartis Window Help

100%

11:21
18/09/2023

ChEMBL disubstituted indoles

lab?reset?to... (2) - JupyterLab

Not secure https://localhost:8888/lab?...

File Edit View Run Kernel Git Tabs Settings Help

CPU: 20% Mem:443 MB

OfficeStripsPNGMetaData.ipy

Untitled.ipynb

Code

rdkit_master

[1]:
```
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole, InteractiveRenderer, MolsToGridImage
```

Last executed at 2023-09-18 12:13:24 in 197ms

[2]:
```
InteractiveRenderer.setEnabled()
```

Last executed at 2023-09-18 12:13:24 in 22ms

Interactive molecule rendering is enabled in this Jupyter Notebook.

[ ]:
```
Chem.MolFromSmiles("")
```

[ ]:

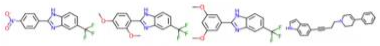
Simple 0 0 0 6 Fully initializ rdkit_maste... Mem: 442... Mode: Com... Ln 1, C... English (United... Untitled.i...

AutoSave Off DataGrid_... Last Modified: 44m ago Tosco, Paolo

File Home Insert Draw Design Transitions Animations Slide Show Record Review View Help

Paste Clipboard Slides Font Paragraph Sensitivity Drawing Editing Dictate Voice Designer RDKit

Business Use Only \ Not Protected Public Business Use Only Restricted Strictly Confidential



Id	CHEMBL1091258	CHEMBL1091651	CHEMBL1091652	CHEMBL109199
HT Solubility pH6.8 [mM]				
LE-MDCK v2 PappAtoB				
MDCK-MDR1 ER				
Direct LogP				
Direct LogD				
rLM CLint				
hLM CLint				
hERG Binding IC50				
NIBR logD	3.48	3.50	3.65	4.11

Slide 2 of 2 English (United States) Accessibility: Investigate Notes 100%

Marvin JS - v20.11.0

Amoxicillin | C16H19N3O5S | CID

https://marvinjs-app.global.nibr.novartis...

Marvin JS by ChemAxon

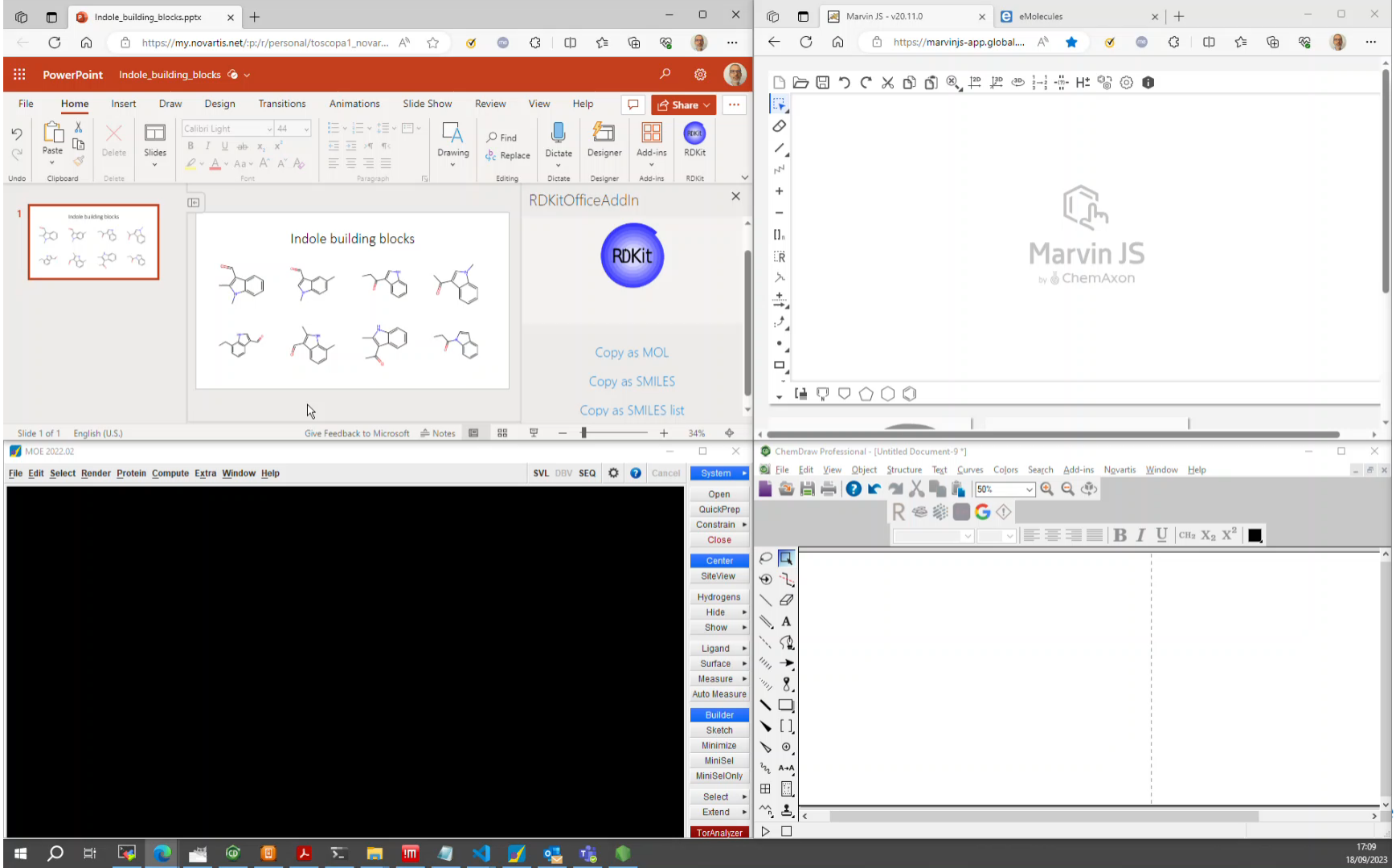
H C N O S F P Cl Br I A

ChemDraw Professional - [Untitled Document-8 *]

File Edit View Object Structure Text Curves Colors Search Add-ins Novartis Window Help

50%

12:14 18/09/2023


Indole_building_blocks.pptx
https://my.novartis.net/:p:/r/personal/toscopa1_novar...
PowerPoint
Indole_building_blocks
File
Home
Insert
Draw
Design
Transitions
Animations
Slide Show
Review
View
Help
Share
Paste
Delete
Slides
Drawing
Find
Replace
Dictate
Designer
Add-ins
RDKit
Indole building blocks
RDKitOfficeAddIn
RDKit
Copy as MOL
Copy as SMILES
Copy as SMILES list
Slide 1 of 1
English (U.S.)
Give Feedback to Microsoft
Notes
34%
Marvin JS - v20.11.0
eMolecules
https://marvinjs-app.global....
Marvin JS
by ChemAxon
MOE 2022.02
File Edit Select Render Protein Compute Extra Window Help
SVL DBV SEQ
Cancel
System
Open
QuickPrep
Constrain
Close
Center
SiteView
Hydrogens
Hide
Show
Ligand
Surface
Measure
Auto Measure
Builder
Sketch
Minimize
MiniSel
MiniSelOnly
Select
Extend
TorAnalyzer
ChemDraw Professional - [Untitled Document-9 *]
File Edit View Object Structure Text Curves Colors Search Add-ins Novartis Window Help
50%
17:09
18/09/2023

Summary

- Implemented MinimalLib API to
  - Read/write molecule metadata from/to PNG images
  - Extract coordinates as a JavaScript array
  - Combine molecules
- Realized a JavaScript Office Add-In to enable copying structure metadata from Office (desktop and web) to chemistry-aware applications
- The only dependencies are `rdkit-structure-renderer` and the Open XML SDK for Javascript

Outlook

- Polish the UI
- Complete testing
- Explore different options for deploying the add-in
- Release as open-source package on GitHub

Acknowledgments

- Biomedical Research
  - Nico Pulver
  - Riccardo Vianello
  - Andreas Liistro
  - Nik Clare
  - Grégori Gerebtzoff
  - Nik Stiefl
  - Daniil Tkachev
  - Ekaterina Stepanova
  - [...]
- RDKit
  - Greg Landrum
  - Ricardo Rodriguez
  - Dan Nealschneider
  - David Cosgrove
  - Brian Kelley
  - Community
  - [...]

Thank you