FOR DISCUSSION:
InChI JNA/WASM Java+JavaScript Joint Development Proposal

Bob Hanson

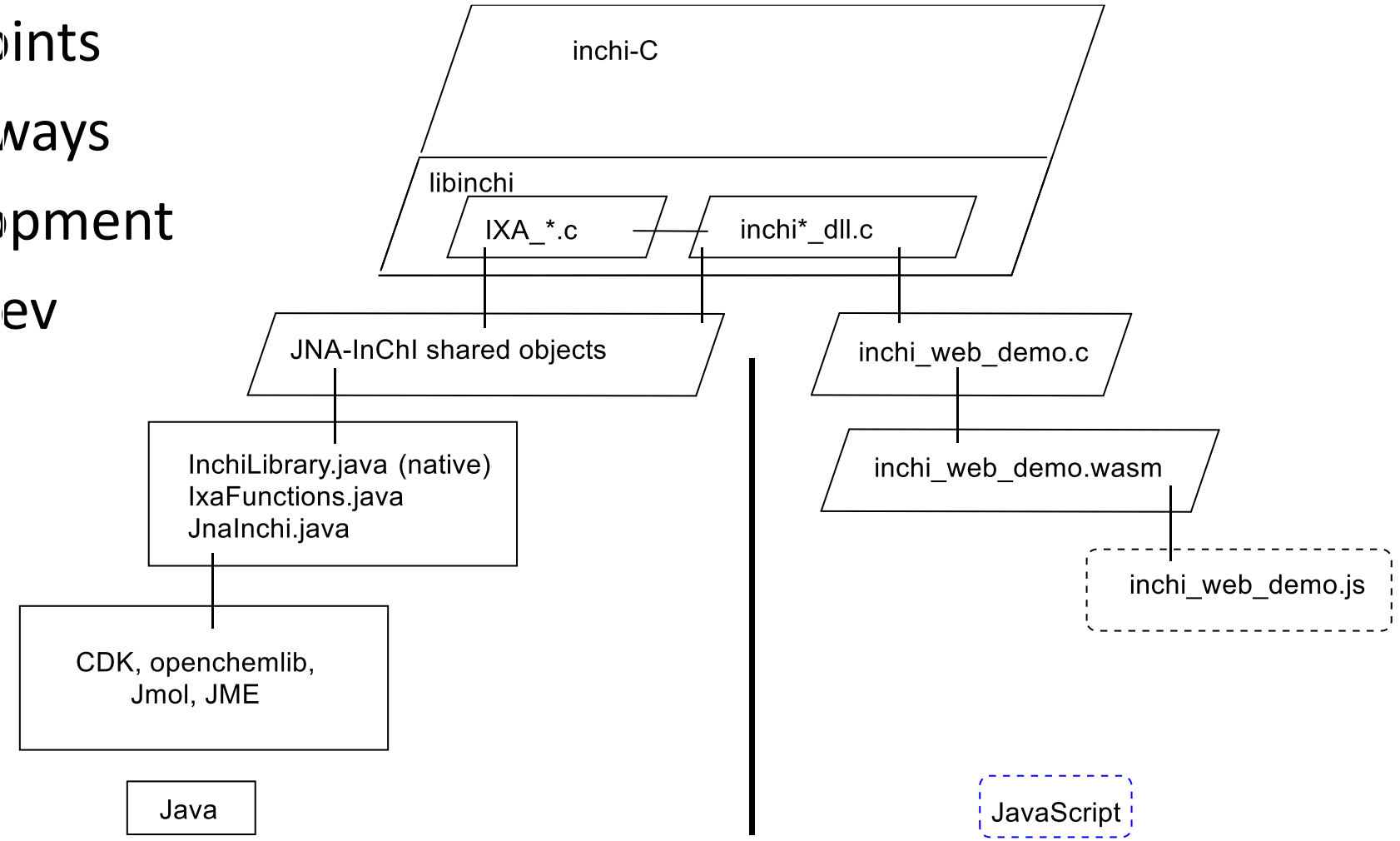
St. Olaf College

hansonr@stolaf.edu

April 2, 2025

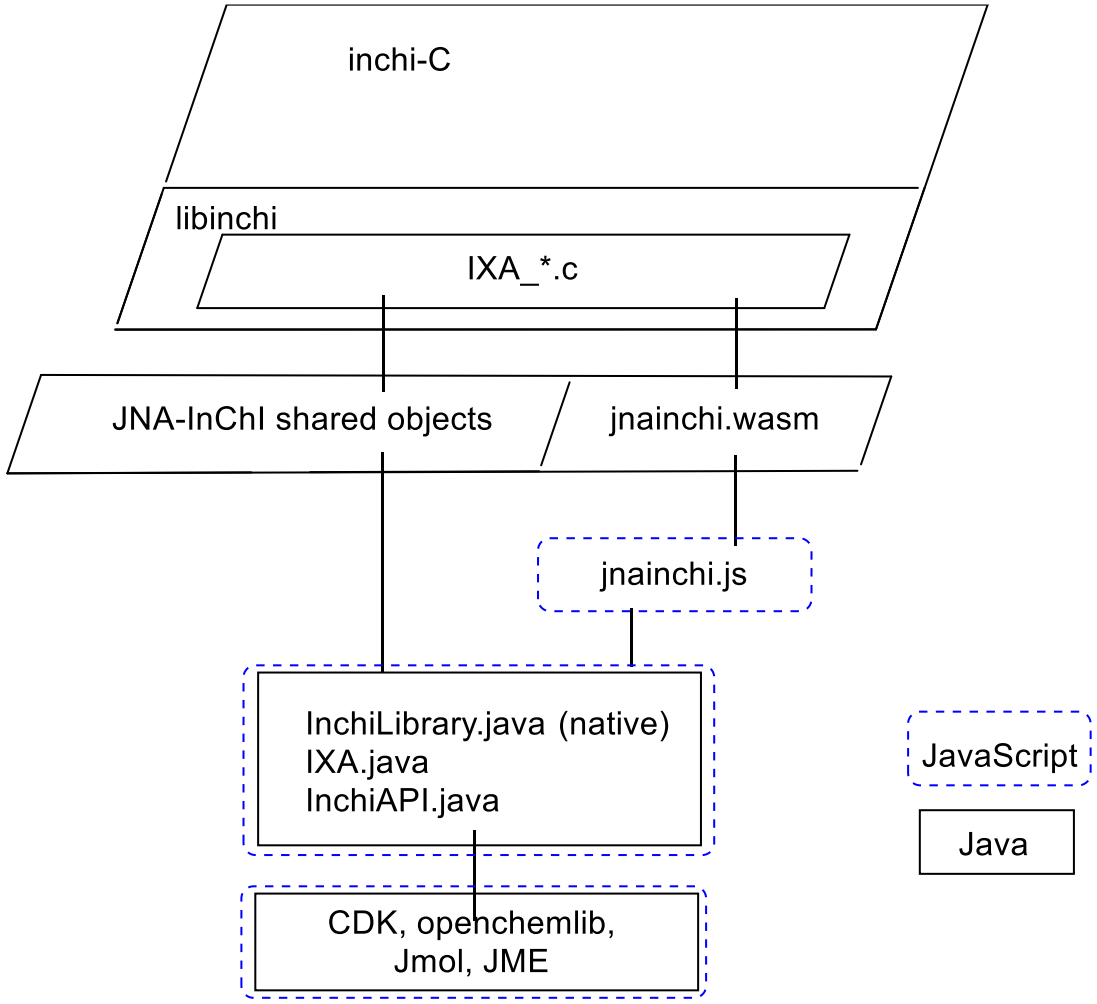
Original configuration

- Different starting points
- Two divergent pathways
- Independent development
- Limited JavaScript dev



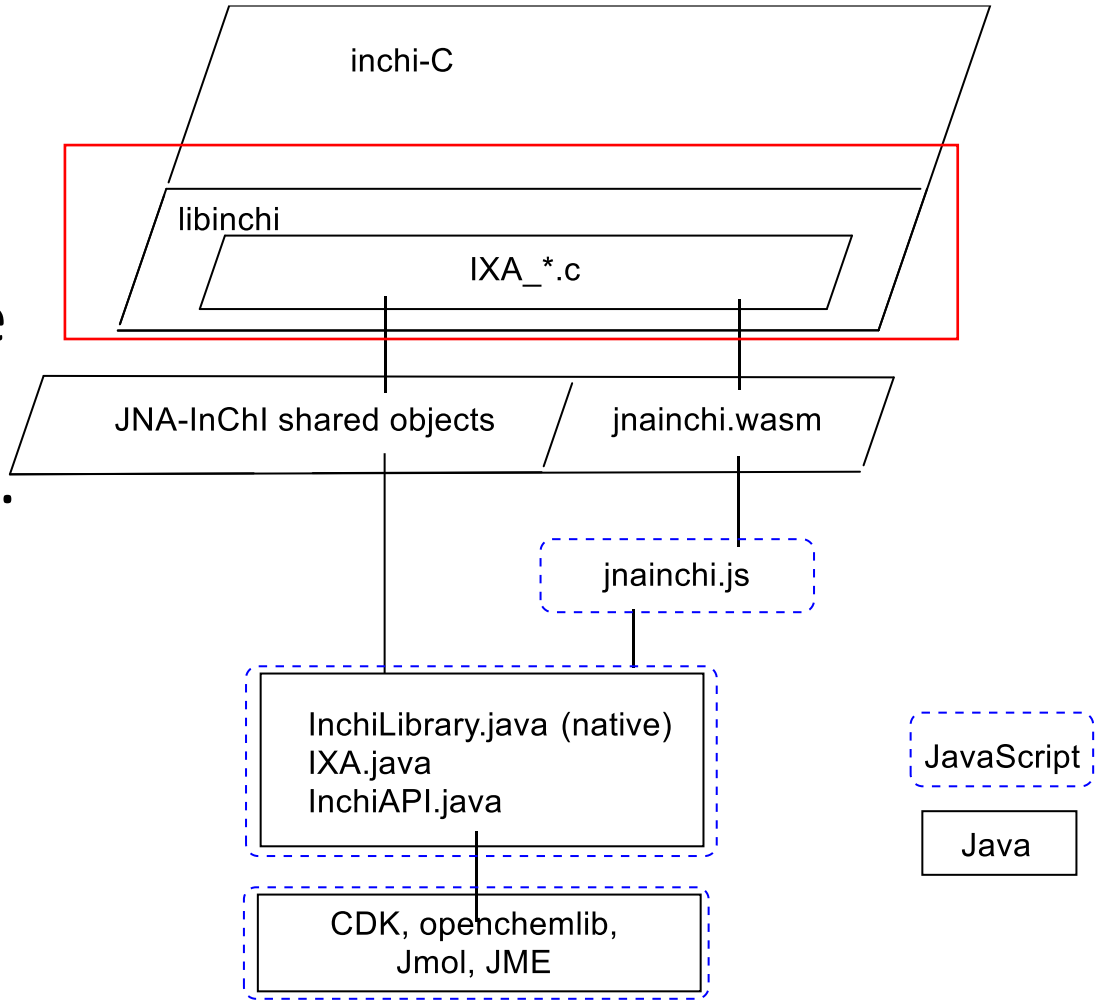
Desired configuration

- Same starting point
- Common IXA-only interface
- Simple methods only
- No significant changes in Java
- Equivalent calls in JavaScript



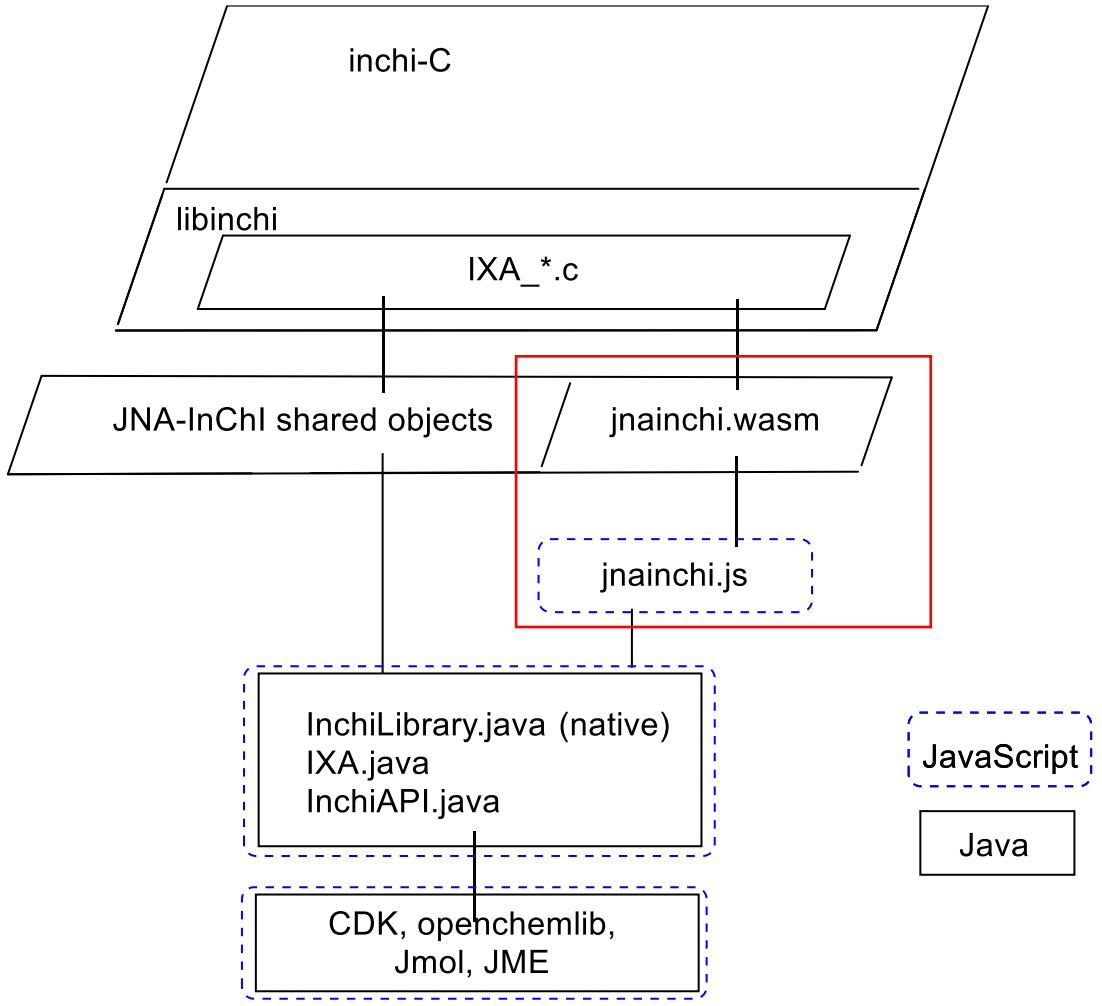
Preliminary work

- No changes were required in inchi-C.
- Two methods were added to inchilib simply to round out the options and provide a built-in way to access the InChI version. These included IXA_Builder_GetInChIVersion and IXA_MOL_ReadAuxInfo



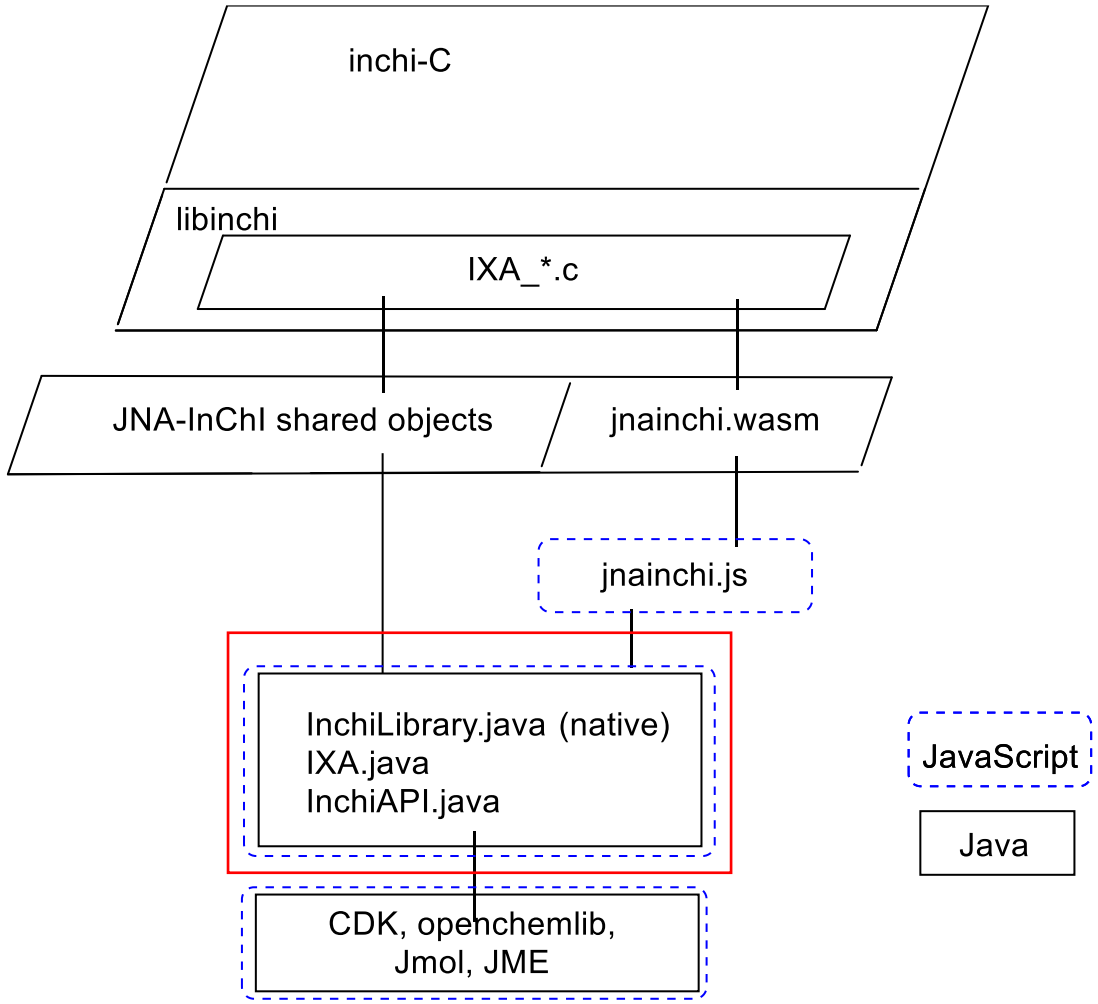
Preliminary work

- WASM makefile adds IXA methods and renames output to match JNA native registration.
- Java2script Eclipse transpiler upgraded to implement “native” methods.
- SwingJS runtime adds Clazz._loadWasm, linking Java native methods with WASM exports when native methods are registered in JavaScript.



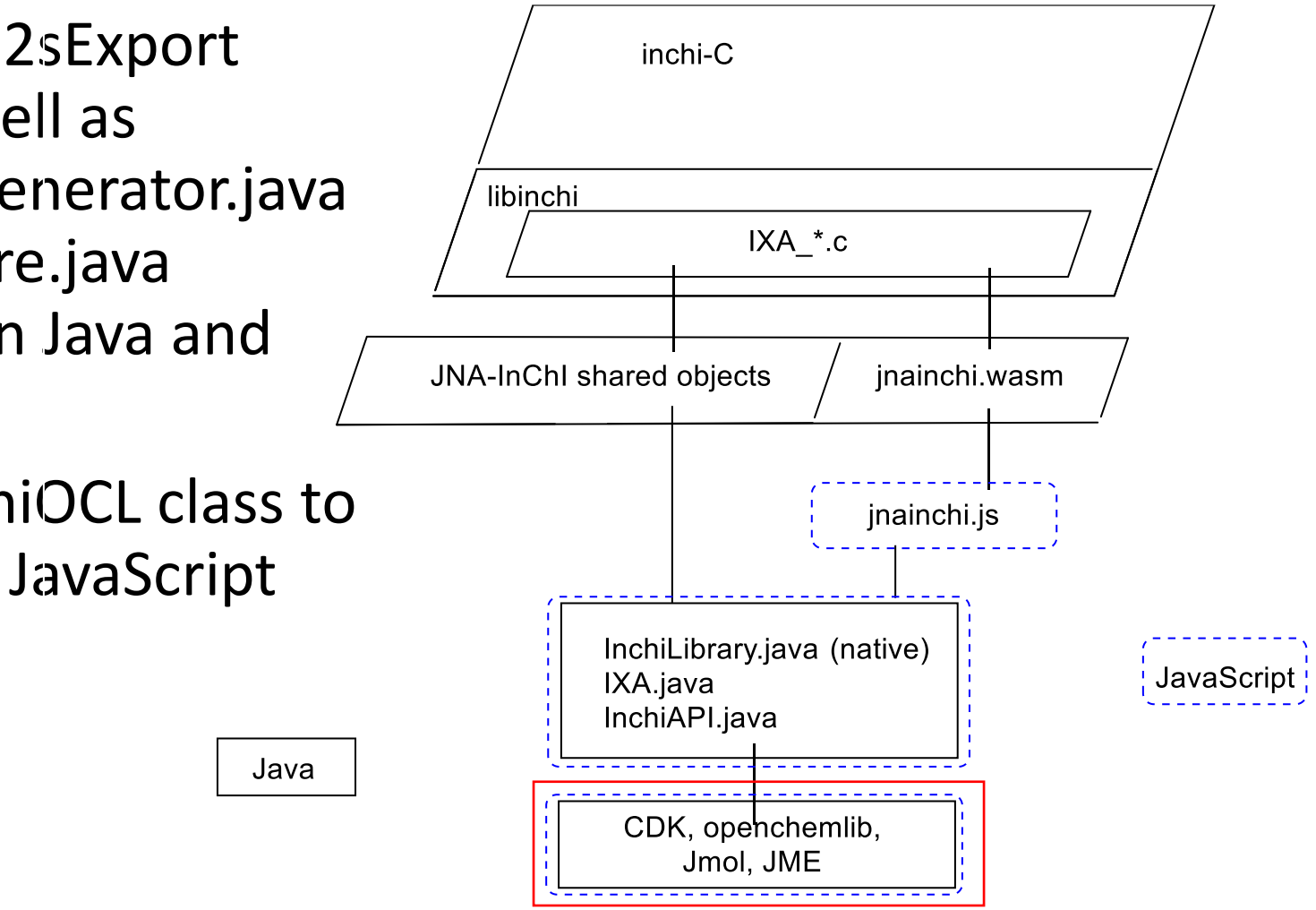
Preliminary work

- JNA-InChI adds InchiAPI.java and IXA.java, avoiding all references to C structures.
- Adds additional inchi output flags “fixamide” and “fixacid” (see slides 15 and 16). These are handled in the InChI-to-InchiInput section of InchiAPI.java.



Preliminary work

- CDK adds SwingJS-j2sExport class CDK.java as well as streamlined InchiGenerator.java and InchiToStructure.java classes functional in Java and JavaScript
- OCL configures inchiOCL class to handle all Java and JavaScript calls to InchiAPI



Preliminary work

- Website demonstrating using both CDK and OpenChemLib in JavaScript for generating InChI from structures and structures from InChI.
- Includes InChI to structure editor, structure editor-to-InChI, CDX and CDXML file dropping, SMILES-to-InChI, MOL file-to-InChI, InChI to PNG, InChI-to-SVG, and OCL molecule-to-CDK molecule.

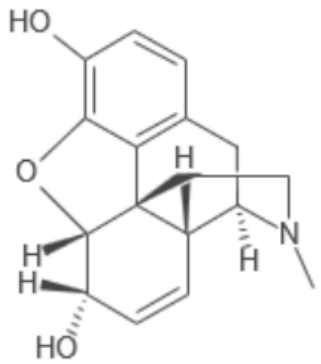
https://chemapps.stolaf.edu/inchi/cdk-web-demo/

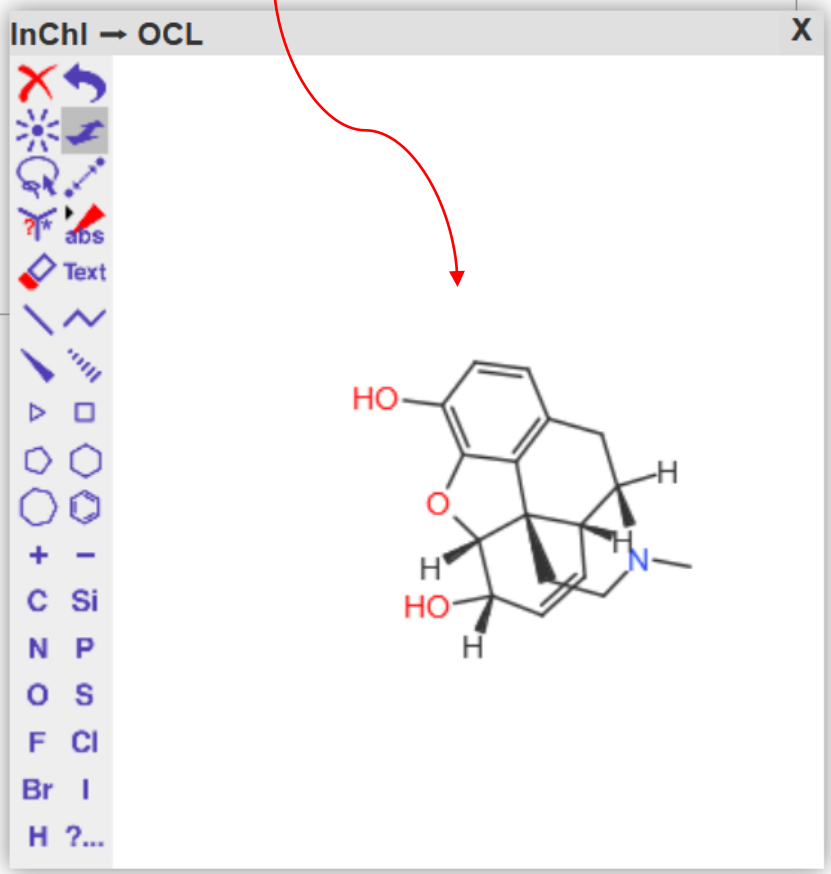
To try it out, enter an InChI such as that for morphine or its enantiomer or this taxol MOL file (with amide fixing) or this AuxInfo or provide a SMILES. Go ahead and reverse the sign of one or more of the stereochemical indicators such as "13-" to "13+". Fun, huh? And, of course, you can drop your own MOL, CDX, or CDXML file into the OCL editor, or just modify the structure that is there. It should immediately update both the InChI and the CDK images. Many thanks to John Mayfield, Daniel Lowe, Jonathan Goodman, and others for assisting us in getting all these pieces working together. Interns Josh Charlton and Swagat Malla assisted in this project. Questions? Talk to me. Bob Hanson.

```
InChI=1S/C17H19NO3/
c1-18-7-6-17-10-3-5-13(20)16(17)21-15-12(19)4-2-9(14(15)17)8-11(10)18/
h2-5,10-11,13,16,19-20H,6-8H2,1H3/t10-,11+,13-,16-,17-/m0/s1
```

InChI to CDK and OCL

InChI → CDK



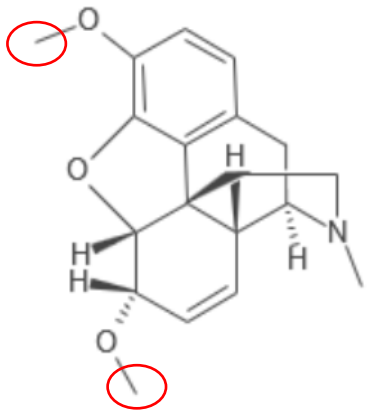


To try it out, enter an InChI such as that for morphine or its enantiomer or this taxol MOL file (with amide fixing) or this AuxInfo or provide a SMILES. Go ahead and reverse the sign of one or more of the stereochemical indicators such as "13-" to "13+". Fun, huh? And, of course, you can drop your own MOL, CDX, or CDXML file into the OCL editor, or just modify the structure that is there. It should immediately update both the InChI and the CDK images. Many thanks to John Mayfield, Daniel Lowe, Jonathan Goodman, and others for assisting us in getting all these pieces working together. Interns Josh Charlton and Swagat Malla assisted in this project. Questions? Talk to me. Bob Hanson.

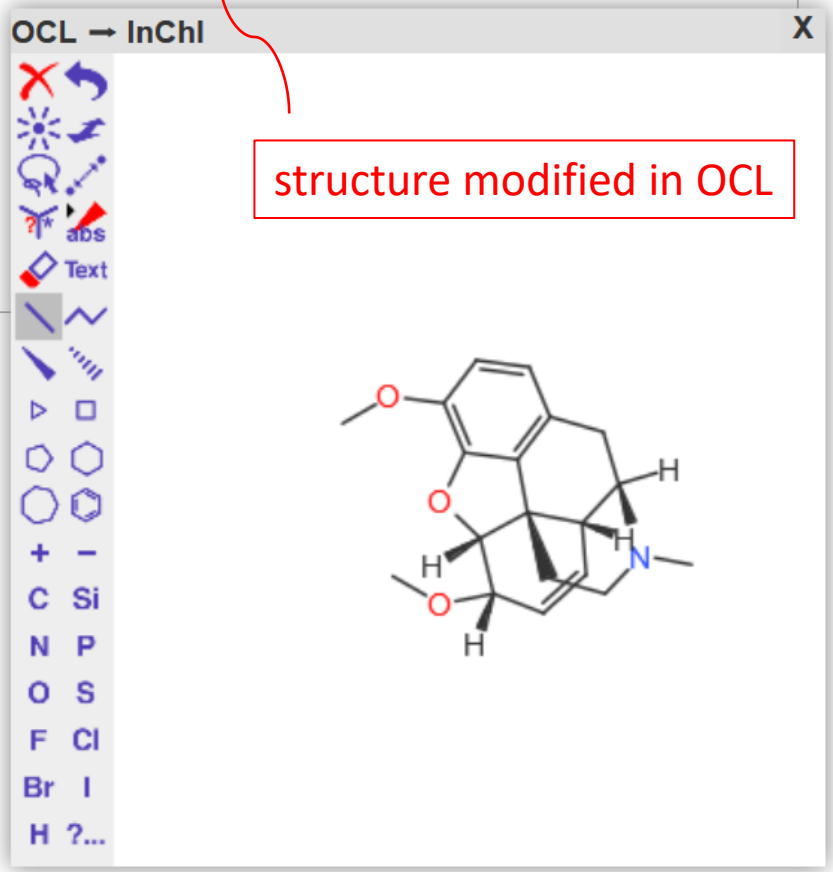
```
InChI=1S/C19H23NO3/
c1-20-9-8-19-12-5-7-15(22-3)18(19)23-17-14(21-2)6-4-11(16(17)19)10-13(12)20/
h4-7,12-13,15,18H,8-10H2,1-3H3/t12-,13+,15-,18-,19-/m0/s1
```

via InChI to CDK

InChI → CDK



OCL → InChI

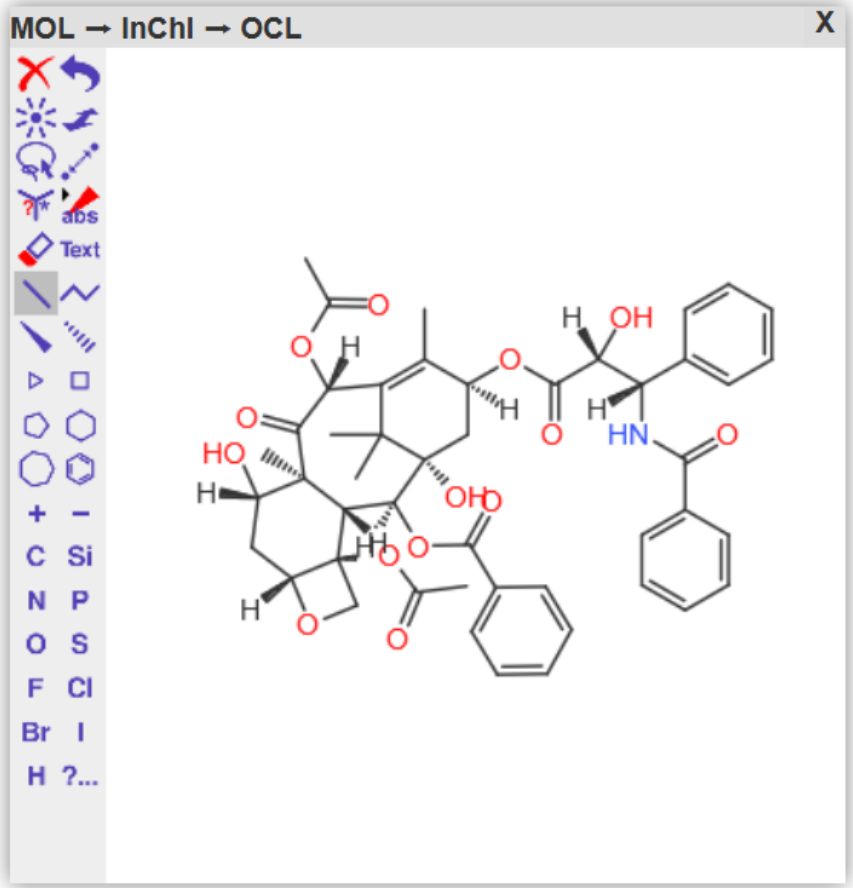


structure modified in OCL

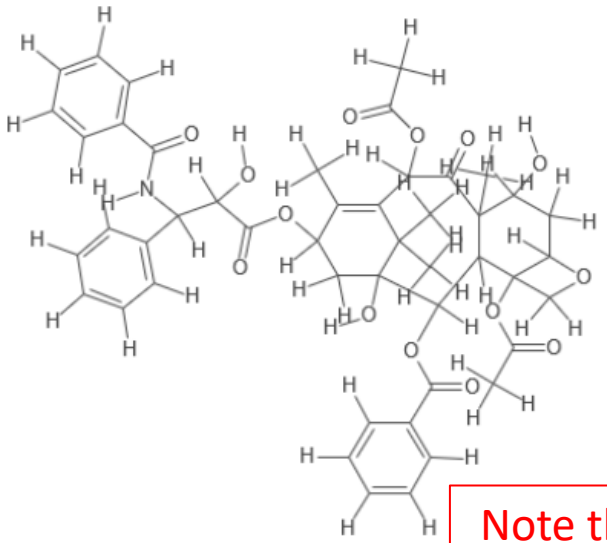
To try it out, enter an InChI such as that for morphine or its enantiomer or this taxol MOL file (with amide fixing) or this AuxInfo or provide a SMILES. Go ahead and reverse the sign of one or more of the stereochemical indicators such as "13-" to "13+". Fun, huh? And, of course, you can drop your own MOL, CDX, or CDXML file into the OCL editor, or just modify the structure that is there. It should immediately update both the InChI and the CDK images. Many thanks to John Mayfield, Daniel Lowe, Jonathan Goodman, and others for assisting us in getting all these pieces working together. Interns Josh Charlton and Swagat Malla assisted in this project. Questions? Talk to me. Bob Hanson.

MOL file to InChI, CDK, and OCL

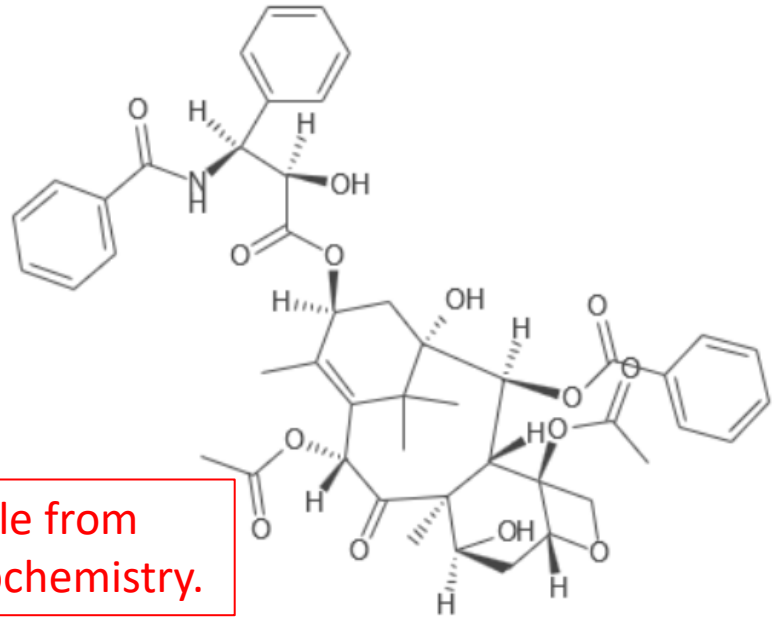
```
InChI=1S/C47H51NO14/
c1-25-31(60-43(56)36(52)35(28-16-10-7-11-17-28)48-41(54)29-18-12-8-13-19-29)23-47(5
7)40(61-42(55)30-20-14-9-15-21-30)38-45(6,32(51)22-33-46(38,24-58-33)62-27(3)50)39(
53)37(59-26(2)49)34(25)44(47,4)5/h7-21,31-33,35-38,40,51-52,57H,22-24H2,1-6H3,
(H,48,54)/t31-,32-,33+,35-,36+,37+,38-,40-,45+,46-,47+/m0/s1
```



MOL → InChI → MOL



MOL → InChI → CDK



Note that MOL file from InChI loses stereochemistry.

Issues and Solutions

One of the strongest arguments for joining forces on JNA and WASM development is that issues arising with implementing InChI, particularly InChI-to-(reasonable)structure can be solved in one place in one go. No duplication of efforts necessary. These aren't necessarily bugs. They may be simply the result of internal InChI representation choices that serve a purpose within inchi-C. The "issue" is only that some applications of InChI need to be made more chemist-friendly, with fewer surprises.

Issues and Solutions

Here I highlight three such problems we have identified:

#1 amide -> InChI -> iminol

#2 phenol-carboxylate -> InChI -> carboxylic acid-phenolate

#3 ammonium-phenol -> InChI -> loss of stereochemistry

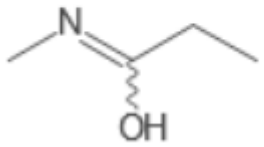
Issue #1 amide -> InChI -> iminol

This is well known. It is not a bug. It is just that inchi-C internally stores amides as iminols, and it is far more likely that a chemist would expect an amide than an iminol as the structural representation.

For example: InChI=1S/C4H9NO/c1-3-4(6)5-2/h3H2,1-2H3,(H,5,6)

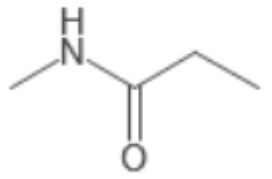
Without “fixamide”:

InChI → CDK



With “fixamide”

InChI → CDK

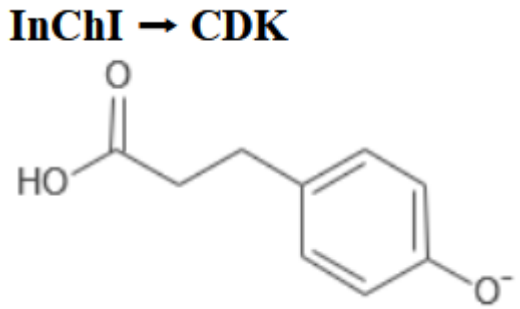


Issue #2 phenol-carboxylate -> InChI -> carboxylic acid-phenolate

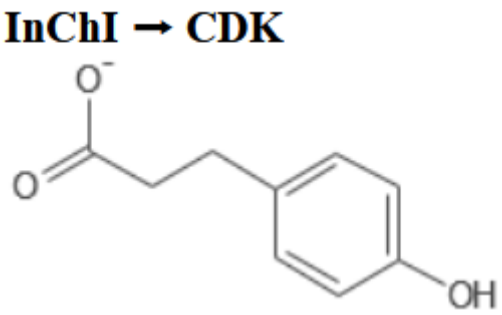
This is less well-known. Again, it is not a bug, in the sense that InChI never has guaranteed a “reasonable” internal chemical model, just one that gets the job done to create a definitive InChI. In this case, the issue is that the anion of a structure that is both a phenol and a carboxylic acid gives a model that is a phenolate, not a carboxylate.

For example: InChI=1S/C9H10O3/c10-8-4-1-7(2-5-8)3-6-9(11)12/h1-2,4-5,10H,3,6H2,(H,11,12)/p-1

Without “fixacid”:



With “fixacid”

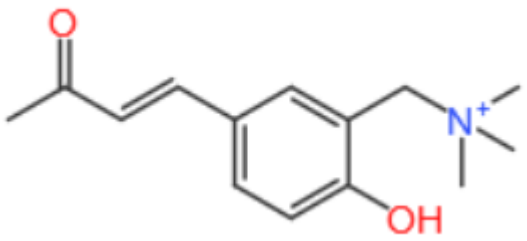


Issue #3 ammonium-phenol loss of stereochemistry

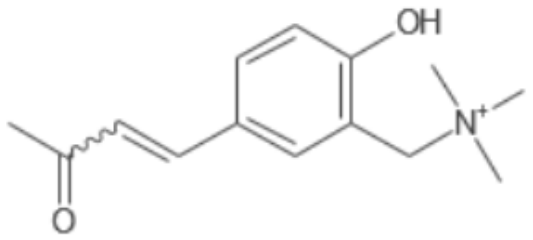
This is a rather odd case that is a bug, because it is a loss of information in the InChI relative to the input structure. It is mentioned in the Tech Manual as “should be corrected”. No fix for this is known at this time, however. For example:

InChI=1S/C14H19NO2/c1-11(16)5-6-12-7-8-14(17)13(9-12)10-15(2,3)4/h5-9H,10H2,1-4H3/p+1

OpenChemLib input:



CDK output from InChI:



Coding Examples

The following examples use SwingJS. As such, the Java methods map directly to JavaScript functions. All of the functions **in bold** are written in Java. They are accessed in both Java and JavaScript. For JavaScript apps not using SwingJS, we would provide IXA calls using Emscripten methods *cwrap* and *ccall*. (And the apps would be responsible for anything else). But in SwingJS that is already taken care of by the JavaScript version of the native library registration process.

Classes swingjs.CDK and swingjs.OCL are imported in Java as CDK and OCL. In the JavaScript they are script-local references to `var CDK = swingjs.CDK` and `var OCL = swingjs.OCL`. These two classes provide equivalent calls in CDK and OCL in both Java and JavaScript for generally useful InChI-related methods. IXA in Java refers (currently) to `io.github.dan2097.jnainchi.IXA`, and in JavaScript refers to that as well, but is also aliased in JavaScript to `J2S.wasm.jnainchi.IXA`.

Coding examples (InChI-to-image, Java)

```
BufferedImage inchiToImage(String inchi) {
  try {
    Pointer hStatus = IXA.IXA_STATUS_Create();
    Pointer hMolecule =  IXA.IXA_MOL_Create(hStatus);
    void IXA.IXA_MOL_ReadInChI(hStatus, hMolecule, inchi);
    InchiInput input = CDK.getInchiInputFromMoleculeHandle(hStatus, hMolecule, “fixamide”);
    IAtomContainer mol = CDK.getCDKMoleculeFromInchiInput(input);
    BufferedImage image = (mol.getAtomCount() == 0 ? null : CDK.getImageFromCDKMolecule(mol));
    return image;
  } catch (e) {
    throw e;
  } finally {
    IXA.IXA_STATUS_Destroy(hStatus);
    IXA.IXA_MOL_Destroy(null, hMolecule);
  }
}
```

Note that all of the IXA method parameters and returns are simple variable types or com.sun.jna.Pointer objects. These are easily handled in JavaScript.

Coding examples (InChI-to-image, JavaScript)

```
function inchiToImage(inchi, moreOptions) {
  try {
    var hStatus = IXA.IXA_STATUS_Create();
    var hMolecule =  IXA.IXA_MOL_Create(hStatus);
    IXA.IXA_MOL_ReadInChI(hStatus, hMolecule, inchi);
    var input = CDK.getInchiInputFromMoleculeHandle(hStatus, hMolecule, “fixamide”);
    var mol = CDK.getCDKMoleculeFromInchiInput(input);
    var image = (mol.getAtomCount$() == 0 ? null : CDK.getImageFromCDKMolecule(mol));
    return (image == null ? "" : CDK.getDataURIForImage(image);
} catch (e) {
    throw e;
  } finally {
    IXA.IXA_STATUS_Destroy(hStatus);
    IXA.IXA_MOL_Destroy(null, hMolecule);
  }
}
```

For JavaScript, in the end we to turn the image into a DataURI in order to display it; this is not necessary in Java.

Coding examples (SwingJS CDK to OCL molecule)

Java:

```
StereoMolecule cdkToOCLMolecule (IAtomContainer mol) {
   InchiInput input = CDK.getInchiInputFromCDKMolecule(mol);
   return OCL.getOCLMoleculeFromInchiInput(input);
}
```

JavaScript (SwingJS):

```
cdkToOCLMolecule = function(mol) {
   var input = CDK.getInchiInputFromCDKMolecule(mol);
   return OCL.getOCLMoleculeFromInchiInput(input);
}
```

These two versions are essentially identical. In both Java and JavaScript. Converting simple structures from CDK to and from OCL is just a matter of passing them through InchiInput. One need not actually create the InChI string. No string parsing is required, and no stereochemistry is lost.

Moving Forward

- Obviously needs more discussion!
- Submit PR requests for InChI, JNA, OCL, and CDK
- Proposing two new InChI repositories:
  - INCHI-API-JNA
    - Focus on Java, with simple IXA methods only
  - INCHI-API-WASM
    - Focus on JavaScript, using simple IXA methods only
  - Coordinated development
- None of this requires the use of SwingJS; we only use that because it also gives us access to OpenChemLib and the CDK in JavaScript.