Addition of Functionality to Jmol/JSmol Application in SASSIE Web

By Samuel Blackman, Under Joseph Curtis and Emre Brookes

Introduction: Jmol/JSmol Project

The Jmol/JSmol application is a molecular visualization program used to examine and animate structures. The Jmol/JSmol application name is interchangeable, however Jmol refers the Java version and JSmol refers the JavaScript version. JSmol is not completely JavaScript because it refers to the existing Java code used by Jmol. A JSmol applet is embedded in SASSIE Web (a program that uses atomistic models to predict scattering data) and this is what I have focused on improving.

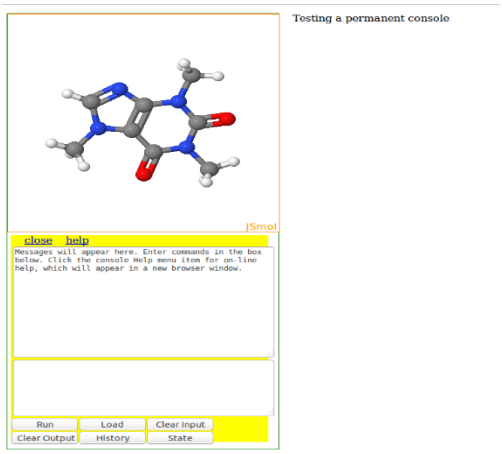
GenApp, a program that can convert standalone applications into a variety of different formats (such as a webpage or Android app) by using JSON definitions, converted the SASSIE standalone application to a web application. GenApp is relevant to the Jmol/JSmol project because any changes that are made within Jmol/JSmol and SASSIE Web must be compatible with GenApp.

The current functionalities of Jmol/JSmol include allowing the user to save the universe of molecules, save the coordinates of individual structures, load multiple structures into the same universe (whether it's all in one PDB or separate PDBs), and manipulate all structures in the universe independently. Because these features were already in place, the project then focused on improving the look and feel of the Jmol/JSmol interface such as a permanent console and a persistent right click menu as well as adding new features to the program including an atom list and a second menu that holds commands.

To be clear, all changes we are making are in JavaScript, not Java. The changes we make are visible in test web pages that load the JSmol applet, not the Jmol applet.

Permanent Console:

The goal with the permanent console was to have it load on launch of the applet. The purpose of having a console at the ready is because in order to manipulate structures precisely, commands from the Jmol/JSmol Interactive Script Documentation (http://chemapps.stolaf.edu/jmol/docs/) must be used. Some of these commands include spin, translate, color, center, and the list goes on. In the example html that the Jmol/JSmol developers sent us, the console loaded on launch, could



not be closed and was fixed in place.

Despite the developer's helpful example html, we had trouble duplicating the permanent console in other test htmls. However, in order to have the console, it's necessary to have the "script: 'console'" definition in the variable info (or myInfo) which is used to define the JSmol applet. It is useful to put make a div in html for the console to go inside, but not necessary.

```
<script type="text/javascript">
var myInfo = {
        width:'100%',
        height:'100%',
        defaultModel: '$caffeine',
        script: 'console'
};
</script>
```

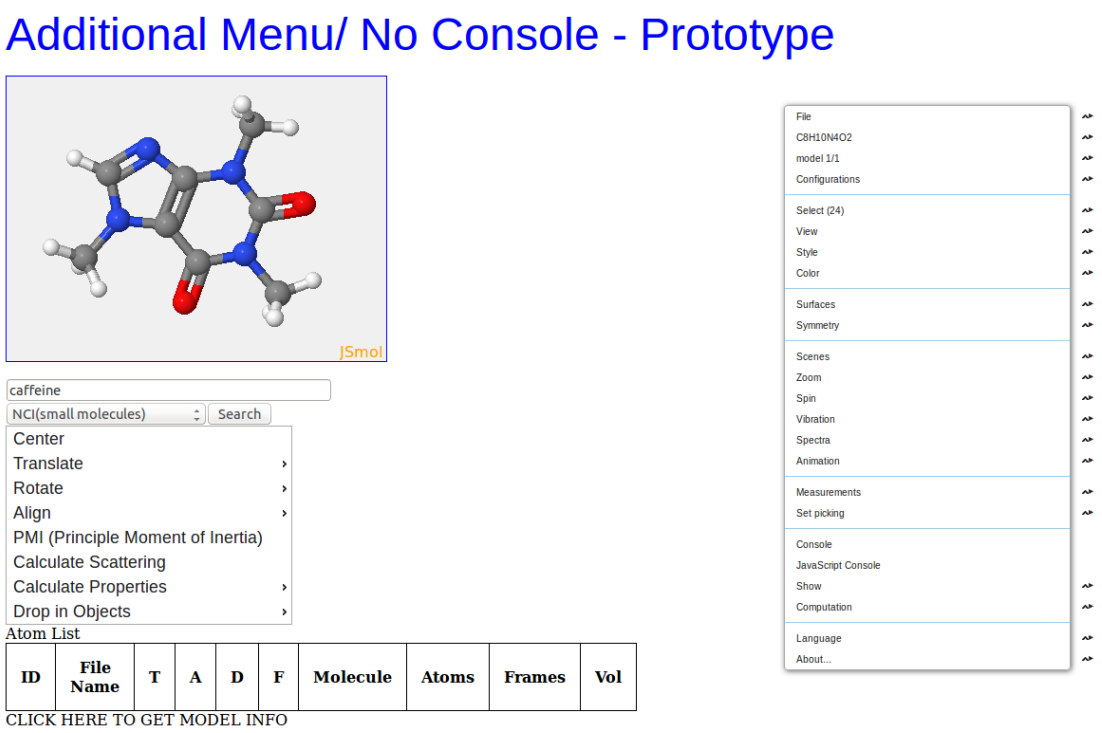
At the moment the status of the permanent console is unfinished, however we know it can be done. At this point we are just waiting on the developers of Jmol/JSmol to get back to us on this predicament; it should be a quick fix though.

Persistent Right Click Menu:

In the current version of Jmol/JSmol the menu will appear if a structure is right clicked, but if the mouse goes off the menu, the menu will disappear. This is problematic because a user might be trying to work on something, and then get sidetracked, only to find that they lost their place in the menu. We worked on persistency with the right click menu, which means that the menu will stay visible and available for usage even if a user mouses off the menu. In our meeting with Robert Hanson, the lead developer of Jmol/JSmol, he added a private method to the JSmolCore.js file "_persistentMenu" that when flagged to true, the menu should become persistent. Unfortunately, from our tests the flag does not work as advertised and we are still

```
_persistentMenu: false,
```

waiting on help from Dr. Hanson to fix the persistency.



At the moment we a several test htmls that have persistent right click menus, but for the most part the menus are still not persistent.

Atom List:

The addition of an atom list is to help users see all of the properties of the structure they loaded into JSmol and to select certain atoms to highlight and examine. The framework of the atom list has been finished, however implementation is still underway.

Atom List

ID	File Name	T	A	D	F	Molecule	Atoms	Frames	Vol

In order to make/grow the table use the code below:

This could does not dynamically grow the table, but if tweaked with a loop, it easily can. “getElementById” searches the entire html document for an Id, in this case “theTable”. After locating the table, “table.insertRow(nextRow);” is used to add another row to the table and from

```
<!-- Atom list -->
<div id="atomList">
<label> Atom List </label>
<table id="theTable" border="1" style="width:10%;">
  <tr>
        <th>ID</th>
        <th>File Name</th>
        <th>T</th>
        <th>A</th>
        <th>D</th>
        <th>F</th>
        <th>Molecule</th>
        <th>Atoms</th>
        <th>Frames</th>
        <th>Vol</th>
  </tr>
  </table>

</div>
```

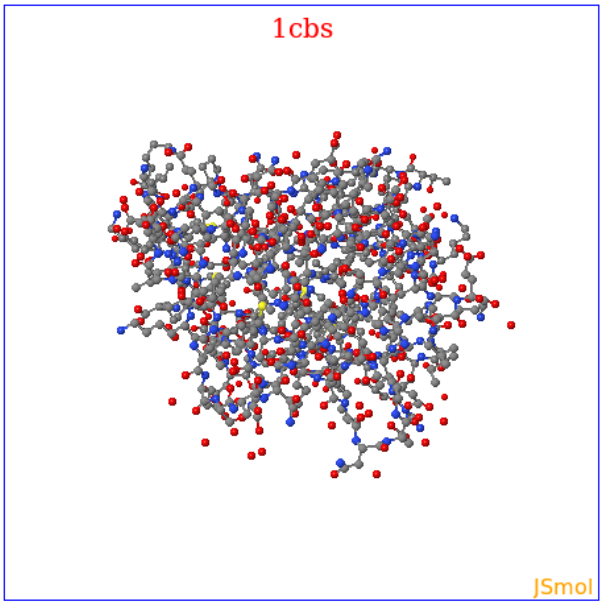
```
<!-- adds a file to the atom list -->
<script>

var nextRow = 1;

//adds the pdb file to the atom list
 function addToTable(){
    var file = document.getElementById('file').files[0];//used to get the filename
    var table = document.getElementById("theTable");

    var row = table.insertRow(nextRow);
    var ID = row.insertCell(0);
    ID.innerHTML = nextRow;
    nextRow++;

    //essentially repeat these lines of code to get the rest of the columns
    var fileName = row.insertCell(1);
    fileName.innerHTML = file.name;
 }


</script>
```

here columns can be individually added to the table.

In terms of the actual implementation of the atom list, there is still work to be done. We were able to figure out how to load a molecule’s name into the table from our own testable local file explorer. We are still trying to figure out how to load a file in the JSmol applet and have that information all go into the atom list as well. In the final days of work, we were trying to use the Jmol object’s function getProperties in order to get a molecules properties into the table from the

JSmol applet, however this has not been completed. As of now, there is a button in validation.htm (which can be found in the JSmol zip, in the repository) that will show all the properties, but it will get rid of the applet and everything else on the page.

Click on the back button on your browser to return to JSmol
.models *List[1]
.models[1].atomCount 1213
.models[1]._ipt 0
.models[1].name "1CBS"
.models[1].bondCount 1129
.models[1].file_model "1.1"
.models[1].polymerCount 1
.models[1].file "http://www.ebi.ac.uk/pdbe/entry-files/download/1cbs.cif"
.models[1].moleculeCount 102
.models[1].chainCount 1
.models[1].num 1
.models[1].vibrationVectors false
.models[1].title "1CBS"
.models[1].groupCount 238
.modelsSelected [0]
.modelSetName "1CBS"
.isTainted false
.modelSetHasVibrationVectors false
.modelSetProperties.PATH_SEPARATOR "/"
.modelSetProperties.PATH_KEY ".PATH"
.canSkipLoad true
.modelCountSelected 1
.modelIndex 0
.modelCount 1

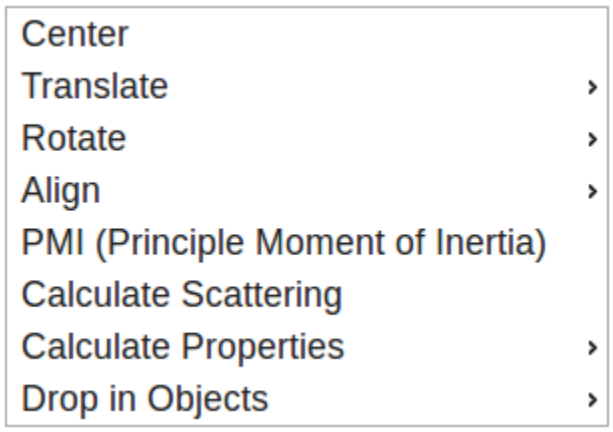


color cpk | color group | color amino | color structure
trace only | cartoon only | backbone only | ball&stick
console | help | Execute

CLICK HERE TO GET MODEL INFO

Additional Command Menu:

The additional command menu eliminates the need to learn some of the basic commands used in the console and adds some new functionalities as well. The center, translate, and rotate are also commands that can be used in the console.



I made the menu using JQuery in JavaScript. In the first picture below, the scripting at the top of the image has to do with the look and feel of the menu (it gives it the “smoothness” look). The scripting after that allows the menu to have sub menus and allows the menu to be draggable

```
<!--gets the necessary jquery libraries, you might need to download them (but probably not)-->
<script src="https://code.jquery.com/jquery-1.10.2.js"></script>
<link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/themes/smoothness/jquery-ui.css">
<script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/jquery-ui.min.js"></script>


<!-- Makes the elements draggable and the menu collapsible -->
<script>
//Makes the menu collapsible
$(function() {
  $( "#menu" ).menu();
});

//makes the menu draggable
$(function() {
  $( "#menu" ).draggable();
});

//makes the laoding menu draggable
$(function() {
  $( "#load" ).draggable();
});

//makes the Atom list draggable
$(function() {
  $( "#atomList" ).draggable();
});

</script>
```

around the screen as well as the atom list and the file explorer we used for testing.

The picture bellow is styling done in CSS. This basically controls the sizing of the menu.

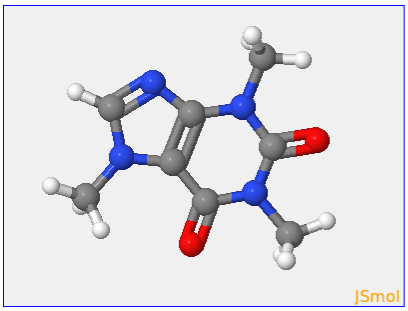
```
<!--Menu list, all items in the menu (this is where you add things -->
<ul id="menu">
  <li>Center</li>
  <li>Translate
      <ul>
      <li>X Axis</li>
      <li>Y Axis</li>
      <li>Z Axis</li>
      </ul>
  </li>
  <li>Rotate
      <ul>
      <li>X Axis</li>
      <li>Y Axis</li>
      <li>Z Axis</li>
      </ul>
 </li>
 <li>Align
      <ul>
      <li>X Axis</li>
      <li>Y Axis</li>
      <li>Z Axis</li>
      </ul>
 </li>
  <li>PMI (Principle Moment of Inertia)</li>
  <li>Calculate Scattering</li>
  <li>Calculate Properties
    <ul>
      <li>Properties
        <ul>
          <li>Atoms</li>
          <li>Bonds</li>
          <li>Bond Angles</li>
        </ul>
      </li>
      <li>Calculate</li>
    </ul>
  </li>
  <li>Drop in Objects
      <ul>
      <li>Sphere</li>
      <li>Plane</li>
      <li>Prism</li>
      </ul>
</li>
</ul>
```

The image above displays the actual contents of the menu, which was created in html. Adding additional menu options should not be a problem, just repeat the format within the <ul> tag (unordered list). The additional menu is still unfinished, in that the menu has not been implemented into the applet, nor do any of the buttons actually do anything.

Conclusion:

We have made great progress in learning about Jmol/JSmol and its abilities as well as making progress in our additions to the application. The pictures below demonstrate two of the prototypes of that bring all the features together.

Additional Menu/ No Console - Prototype



caffeine

NCI(small molecules) Search

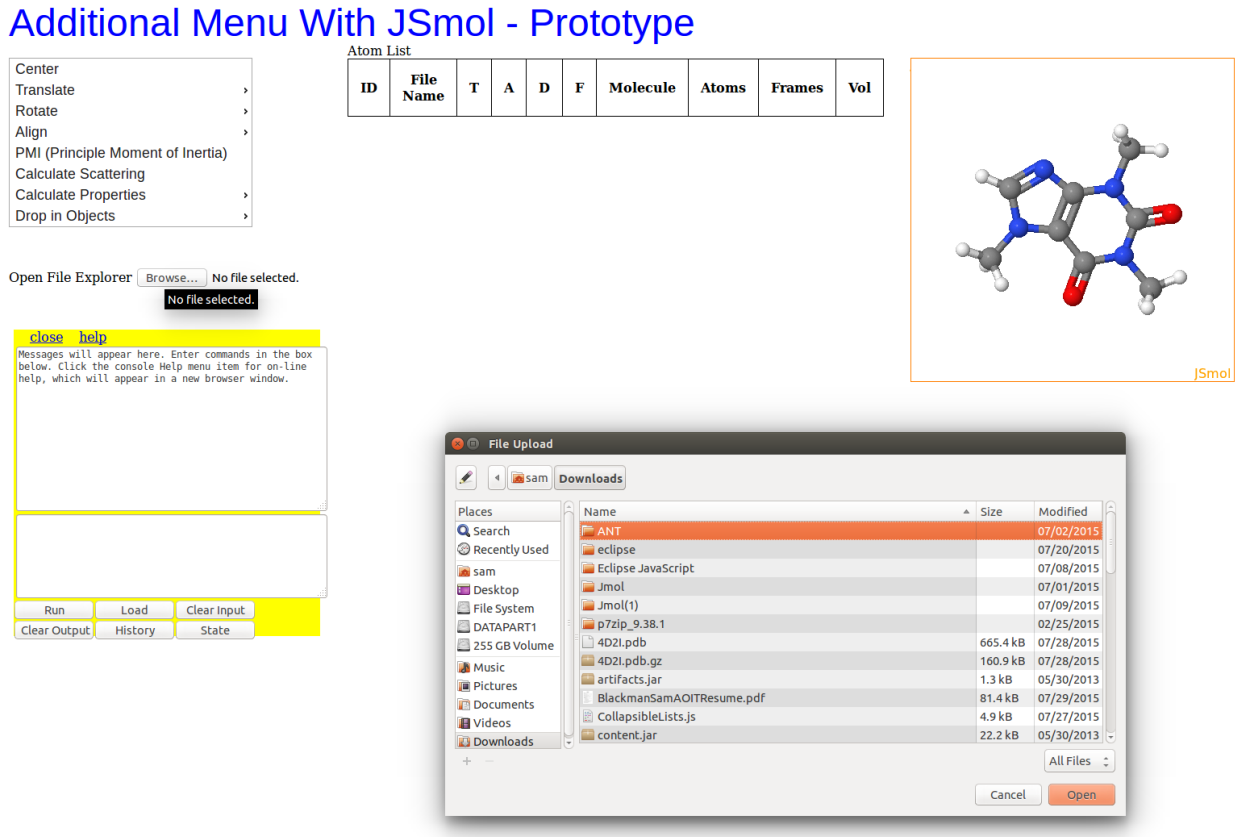
Center
Translate ›
Rotate ›
Align ›
PMI (Principle Moment of Inertia)
Calculate Scattering
Calculate Properties ›
Drop in Objects ›

Atom List

ID	File Name	T	A	D	F	Molecule	Atoms	Frames	Vol

CLICK HERE TO GET MODEL INFO

File
C8H10N4O2
model 1/1
Configurations
Select (24)
View
Style
Color
Surfaces
Symmetry
Scenes
Zoom
Spin
Vibration
Spectra
Animation
Measurements
Set picking
Console
JavaScript Console
Show
Computation
Language
About...



Just a few notes, the Jmol/JSmol code documentation is rather incomplete. Also, the JSmol code is highly modularized, which is good, but makes the code hard to comprehend without developer help. In addition, due to unresponsive developers, the ability to add all of these new functionalities into JSmol has been hindered.

References:

1. Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org/
2. Emre H. Brookes, Nadeem Anjum, Joseph E. Curtis, Suresh Marru, Raminder Singh, and Marlon Pierce.2015. The GenApp framework integrated with Airavata for managed compute resource submissions.Concurrency and Computation: Practice and Experience. Concurrency Computat.: Pract. Exper. 1532-0634. 10.1002/cpe.3519. CASE tools. Science Gateway. http://dx.doi.org/10.1002/cpe.3519
3. Dr. Joseph Curtis
4. Dr. Emre Brookes
5. Dr. Robert Hanson