Narrated Jmol Tutorials:

FILE ORGANIZATION:

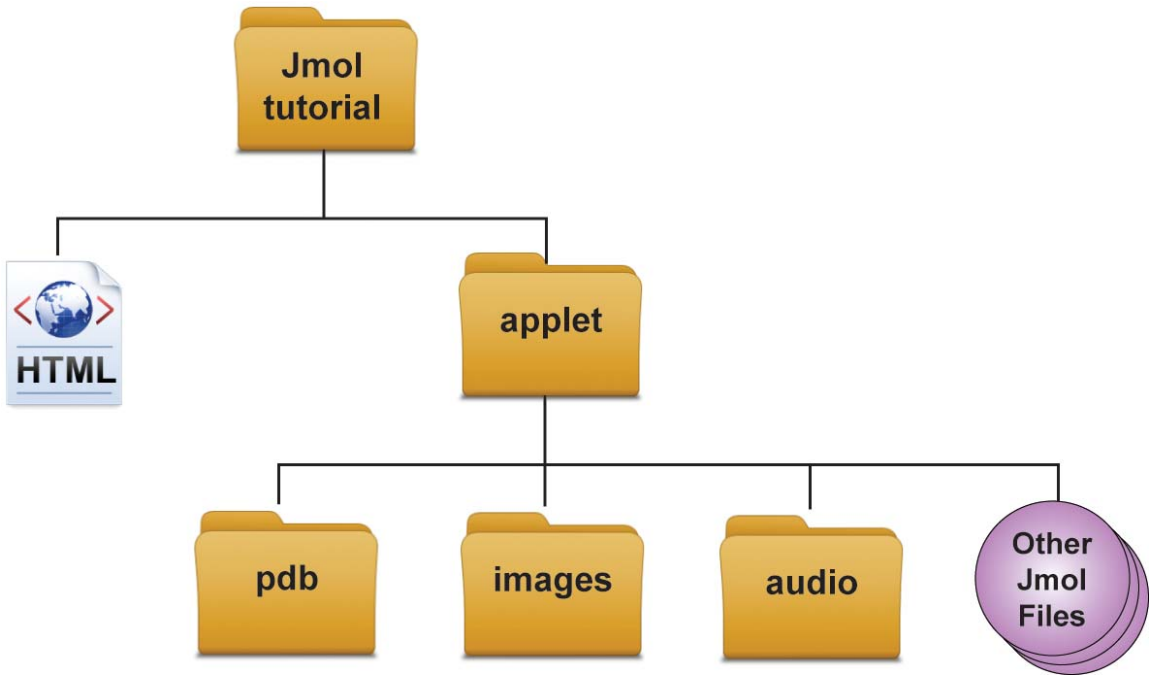
The first step in creating a narrated Jmol tutorial is understanding the basic file organization. A template of this organization can be downloaded as a .zip file from the Narrated Jmol Tutorial page on the CBM's website.

Every Jmol Tutorial requires at 2 main items:

1. The HTML document - This is the main file that holds your content. In other words, it is where you will specify what titles and text you want, what images and buttons will be displayed, and what Jmol code and audio files will be loaded.

2. The Jmol applet folder - This contains all the files necessary for Jmol to run correctly on a browser. It also includes the three folders that you will use to store your images, audio files, and pdb files that your HTML document will reference.

The graphic below represents the organization of these files:



Audio Files:

All audio files should be stored in the "audio" folder inside the "applet" folder. These can be wave (.wav) or mp3 (.mp3) audio files. Wav files are much larger and don't stream well on slow Internet connections, but they are of a higher quality. You may wish to save audio as .wav files, then use a file converter to create .mp3 files. You can then evaluate the quality of the .mp3 files to determine whether you can reduce file size without reducing sound quality.

Audio can be recorded on a number of devices. Many smart phones and computers even have audio recording as a built in feature. If you do not have access to a recording tool, USB port microphones can be purchased for a minimal cost - they may not be the highest quality, but they'll get the job done! A useful free piece of software for recording is "Audacity" (available from http://audacity.sourceforge.net). This program is easy to learn and will allow you to crop, cut, edit and export audio files.

Note: We recommend that you use several shorter audio clips throughout your tutorial instead of one extremely long audio clip. This will allow the users to set their own pace and have more control over the content.

Note: Make sure you do not include any spaces or special characters (% $ # @ * etc. etc.) in your audio file names. For example:

Correct: "mark_audio_1.mp3" Wrong: "Mark's Great Audio!!!.mp3"

Image Files:

All image files should be stored in the "image" folder inside the "applet" folder. These can be wave Gif (.gif), Jpeg (.jpg), or Ping (.png) image files. (For a quick reference on differences among these three types of files, check out: http://blogs.sitepoint.com/gif-png-jpg-which-one-to-use/.)

Note: We recommend that you do not use huge images. The tutorial template has been set up to automatically resize images that are larger than the viewing space, but large images will still take longer to load and will slow down your browser.

Note: Make sure you do not include any spaces or special characters (% $ # @ * etc. etc.) in your image file names. For example:

Correct: "mark_is_cool.jpg" Wrong: "CBM is the best :) .jpg"

PDB Files:

All pdb files should be stored in the "pdb" folder inside the "applet" folder.

Note: If you are using a PDB file that is not from www.pdb.org, you will need to make sure that the PDB file name has only four characters. Make sure you do not include any spaces or special charecters (% $ # @ * etc. etc.) in your pdb file names. For example:

Correct: "1rtx.pdb Wrong: "This is - - - WRONG.pdb"

EDITING THE HTML:

For those who aren't used to it, HTML can be intimidating. You may open the file and see what looks like a foreign language. But don't panic! This section should explain what you see step by step - and tell what to edit and what NOT to edit.

HTML can be viewed in any text editing program. To open it in a text editing program, right click on the file and select "open with". We suggest something simple like notepad or Microsoft Wordpad. If you use Microsoft Word or another word processing program, be sure to save the file as a .txt file.

HTML code uses what are called "tags". Tags are basically a way to organize your content (in other words, organizing the text, images, and buttons you will want to display in your tutorial). These tags are always enclosed by carrots and include both an opening tag and a closing tag with a slash at the beginning:

<HTML> ← Opening Tag
</HTML> ← Closing Tag

Whatever you include in between the opening and closing tag will be including in that section of your content. For example, the HTML code below would include the words "CBM is cool" inside your HTML tag:

```
<HTML>
        CBM is cool
</HTML>
```

Note: EVERY tag must have both an opening and closing tag - if you do not have both, your file will not properly load in the browser. Tags within tags must be properly nested, or the computer won't be able to read the code. Occasionally, the opening and closing tag will be combined like this:

<HTML />

The Title Tag:

The first thing you will want to edit is the Title tag. This tag contains the name of your page and will be displayed at the top of your browser and used by search engines (like Google or Yahoo) if your page is online. The default we have in our sample is "MSOE Center for BioMolecualr Modeling - Title Goes Here!". To change this, simply edit the text between the opening title tag (<title>) and the closing title tag (</title)>

For example:

```
<title>
        CREST Team - Influenza Virus Jmol Tutorial
</title>
```

Note: This is the only thing you will need to edit or change inside the <head> tag.

Main Content

All of your main content (the text, buttons, audio and images) will fit inside the tag. This tag looks complicated, but really it is just a tag with some extra info in the opening tag.

```
<div id=" main content" >
        Your content will go here!
</div>
```

Titles and Subtitles:

Inside your main content div ( ), you can organize your text using the <h1>, <h2>, <h3> and <p> tags. Each of these four tags will make your text look slightly different so that you can visual section off information. Below are some samples of the html code followed by how it will look in your browser:

```
<h1>
        Hi there World! This is a title in h1 tags!
</h1>
```

Hi there World! This is a title in h1 tags!

```
<h2>
        This is a sub-title in h2 tags!
</h2>
```

This is a sub-title in h2 tags!

```
<h3>
        This is a sub-sub-title in h3 tags!
</h3>
```

This is a sub-sub-title in h3 tags!

```
<p>
        This is regular block text in a paragraph tag!
</p>
```

This is regular block text in a paragraph tag!

Note: Think of these tags as ways of showing hierarchy for your content. The most important title will go in the <h1> tag. The second most important text will go in the <h2> tag. The third most important will go in the <h3> tag. And finally, the main block text or paragraph text will go in the <p> tag. You can use as many of each type as you feel you need - but always remember to have both an opening and closing tag.

Adding Images:

To add an image to your content, start by making sure your image file is in the correct "image" folder inside the "applet" folder. Then, using the code below, link your image in the HTML document:

```
<img src="applet/images/name_of_image.jpg" alt="description of image" />
```

You will change the "name_of_image.jpg" to the actual file name you want to use.

You will change the "description of image" to an accurate description of what image you are showing. This description in the "alt" property is important because it can be used by screen reading software for anyone with vision problems and is also used by search engines (like Google, Yahoo, etc.) when doing image searches.

Note: This template page will not allow images to display larger than the content area, but if you want your image to display at a certain exact size, you can add the width="x" and height="y" properties. For example:

```
<img src="applet/images/name_of_image.jpg" alt="description of image" height="200" width="100" />
```

Adding Buttons - Part 1:

To add a jmol button with sound, start by adding the button paragraph code to the main content area (div #id="main_content">) using the code below:

```
<p>
        <input type="button" name="1" value="Click to Highlight Carbon Atoms"
                onClick="carbon('sound1')"/>
        <br />
        <embed src="applet/audio/carbon.wav" autostart=false height="0" width="0"
         name="sound1" enablejavascript="true">
        </embed>
</p>
```

This looks pretty confusing, so let's break this down just a bit more. To start, the entire button code is contained within a paragraph tag . Always make sure you have an opening tag and a closing tag .

The first item inside the paragraph tag will be the <input> tag. This basically tells your browser that you want to have a button placed at this location. The "value" property that in our sample button says "Click to Highlight Carbon Atoms", is the text that will be placed on your button.

Note: it is a good idea to include something like "click here" in the button value so that the user knows that it is a button they should click on.

This will call up a function farther down in the code (I know this sounds complicated - but don't worry about the function yet - it will be explained in more detail below). You need to make sure that the "onClick", "name" and "value" properties are unique for each button you create. For example, if you have two buttons, you may want to use something like this for the two "onClick" properties:

```
<input type="button" name="1" value="Click to Highlight Carbon Atoms"
        onClick="carbon('sound1')"/>

<input type="button" name="2" value="Click to Highlight Nitrogen Atoms"
        onClick="nitrogen('sound2')"/>
```

Next is the
 tag which is here just to create a line break on the page (in other words, it is the same as hitting return when typing in a word processor).

The second main part of the button code is the <embed> tag. This will tell your browser what audio file you want to play when the button is pressed. If you do not have any audio for a button, just omit this part of the code.

The "src" property will tell your browser the name of the sound file. Remember, you should have all your sound files saved in the "audio" folder inside the "applet" folder. In this sample button, the audio file is "carbon.wav".

```
<embed src="applet/audio/carbon.wav" autostart=false height="0" width="0" name="sound1"
        enablejavascript="true">
</embed>
```

You will also have to make sure that the "name" property **in your <embed> tag** is the same as the "onClick" property **in the parentheses from the <input> tag above**. For example, if your "onClick" property has a value of sound1 within the parentheses, then your <embed> "name" property should also be sound1. (See terms in blue below.)

```
<p>
        <input type="button" name="1" value="Click to Highlight Carbon Atoms"
                onClick="carbon('sound1')"/>
        <br />
        <embed src="applet/audio/carbon.wav" autostart=false height=15 name="sound1"
                enablejavascript="true">
        </embed>
</p>
```

Adding Buttons - Part 2:

We should now have the main portion of the button done, but we need to add the function that is called when the button is clicked. This function will contain a bit of code needed to start the audio file as well as a bit of code that contains the Jmol script you want the button to activate.

```
<script language="JavaScript" type="text/javascript">
        function name_of_function(soundobj) {
                var thissound= eval("document."+soundobj);
                thissound.Play();
                jmolScript("Jmol commands here","1");
        }
</script>
```

There are only two sections of this code you will need to edit for each button.

The first is the name of the function called. This should match the value you have for the "onClick" property that we discussed on the previous page. For example, if you have created a button with the code:

```
<p>
        <input type="button" name="1" value="Click to Highlight Carbon Atoms"
                onClick="carbon('sound1')"/>
        <br />
        <embed src="applet/audio/carbon.wav" autostart=false height=15 name="sound1"
                enablejavascript="true">
</p>
```

Then your function will use the name "carbon" when it is called:

```
<script language="JavaScript" type="text/javascript">
        function carbon(soundobj) {
                var thissound= eval("document."+soundobj);
                thissound.Play();
                jmolScript("jmol commands here","1");
        }
</script>
```

The second will be the actual Jmol commands that you want to activate when the button is pressed. This will use the exact same command language you have been using in the Jmol Console Window. The only difference is that instead of pressing "enter" to submit each command as you would when using the Jmol console, you will separate the various commands using semicolons (;). For example, if you want to have you button remove all display formats except a backbone of 1.0 angstroms, you would include the code:

```
jmolScript("select all; spacefill off; wireframe off; cartoon off; backbone 1.0;","1");
```

Note: There are a variety of Jmol commands that can help to liven up your narrated Jmol tutorials. These may not be commands you have used when designing a model, but they can be very useful when creating tutorials. For example:

> **The "Delay" command -** This command pauses the commands you are sending to Jmol for a certain amount of time. For example, if I wanted the molecule to spin for two seconds, then pause for two seconds, and then begin spinning again, I would use the code:
>
> ```
> jmolScript("select all; spin; delay 2; spin off; delay 2; spin;","1");
> ```

There are a variety of other useful commands that can help make these simple buttons come to life. For an excellent tutorial on animation-specific Jmol commands, visit David Marcey's Jmol tutorial at:

http://www.callutheran.edu/BioDev/omm/scripting/molmast.htm#VII

The Jmol Initiate Code:

The last thing we have to be aware of is the Jmol Initiation code. This is basically the code that tells the browser where on the screen it should launch the Jmol Display Window. All of this code should be contained within the tag and will look like this:

```
<script language="JavaScript" type="text/javascript" src="./applet/Jmol.js">
</script>
<script language="JavaScript" type="text/javascript">
	if(!navigator.javaEnabled()){
		alert('You must have Java installed and enabled in order to view Jmols.  Go to
www.java.com to install the latest version of Java.');
	}
	jmolInitialize("./applet");
	jmolApplet("100%","set messageCallback \"showmsg\" ; load ./applet/pdb/test.pdb; set frank
off; set defaultColors rasmol; background [250,240,235]; select all; color cpk; spacefill; wireframe off;
backbone off;","1");
	jmolBr();
</script>
```

The only portion of all this you should edit will be the Jmol commands that will be activated when the page is first loaded. In other words, this is where you will put the commands that open the .pdb file you want to use and determine how it will initially display.

For example, the code shown above will open the .pdb file "test.pdb" that is located in the "pdb" folder inside the "applet" folder. It will then set the colors to the default RasMol/Jmol colors and will turn the background of the Jmol a nice light, light red color. From there, it will select the entire model, color it cpk, turn the spacefill on and turn the wireframe and backbone off.