

Jmol interactive scripting documentation version 12.0

See an error? Something missing? Please [let us know](#). For a wide variety of interactive examples, see [new.htm](#).

 Search the Database	 Index	 Examples	
[Jmol SMARTS/SMILES] *	color (scheme)	if	refresh
[Jmol command syntax]	color measures	initialize	reset
[Jmol math]	compare *	invertSelected *	restore
[Jmol parameters]	configuration	isosurface *	restrict *
[atom expressions]	connect	label	resume
[atom properties]	console	lcaoCartoon	return
[comment (#)]	continue	load *	ribbon
[export]	data	load APPEND	rocket
[fractional coordinates]	default *	load DATA *	rotate *
[functions]	define	load FILES	rotateSelected
[plane expressions]	delay	load MENU	save
[read/write ZIP files]	delete	load MODELS	script *
[status reporting]	depth	load TRAJECTORY	select
[using the clipboard]	dipole	load [property]	selectionHalo
animation	display	log *	set
axes *	dots	loop	set (antialiasing)
backbone	draw *	mapProperty *	set (bond styles) *
background	echo	measure *	set (callback)
bind *	ellipsoid	meshribbon	set (debugging) *
bondorder	else	message	set (files and scripts) *
boundingbox *	elseif	minimize *	set (highlights)
break	exit	mo *	set (labels)
calculate *	fix	model	set (language)
cartoon	font	move	set (lighting) *
case *	for	moveto	set (measure)
catch *	frame	navigate	set (misc) *
cd	frank	parallel/process *	set (navigation)
center	geoSurface	pause	set (perspective)
centerAt	getProperty	plot *	set (structure) *
color *	goto	pmesh	set (visibility) *
color (atom object)	halos	polyhedra	set echo
color (bond object)	hbonds *	print	set modelKitMode *
color (element)	help	prompt *	set picking *
color (model object)	hide	quaternion *	set pickingStyle *
color (named object)	history	quit	set userColorScheme
color (other) *	hover	ramachandran *	show *
			slab
			spacefill *
			spin
			ssbonds
			star
			step
			stereo
			strand
			structure
			struts *
			subset
			switch *
			sync
			timeout *
			trace
			translate
			translateSelected
			try *
			unbind *
			unitcell *
			var
			vector
			vibration
			while
			wireframe
			write *
			write (export)
			write (image, fra
			write (info) *
			write (model) *
			write (object)
			zap
			zoom *
			zoomto *

\* indicates new or modified in version 12.0

**[Jmol SMARTS/SMILES]**

(v. 12.0 -- new)

Jmol 12.0 supports a full implementation of [SMILES](#) and [SMARTS](#) along with powerful extensions (see <http://jmol.svn.sourceforge.net/viewvc/jmol/trunk/jmol/src/org/jmol/smls/package.html>) that allow the SMARTS syntax to be used for biomolecular substructure searching and three-dimensional conformation searching. The bioSMILES, bioSMARTS, and 3D-SMARTS syntaxes are relatively simple

extensions of the Daylight definitions, including all stereochemistry and "primitive" syntax in those definitions. Importantly, Jmol can search SMILES strings themselves (independent of any loaded structure) and find matches. This feature allows, for example, checking of student answers to questions that require the entry of structures using 2D drawing programs such as JSDraw and JME. Examples of using SMILES and SMARTS include:

<code>select search("[r4"])</code>	search for all atoms in 4-membered rings
<code>x = {*}.find("SMARTS", "[C]C=O")</code>	load the variable x with all alpha-carbons.
<code>x = {1.1}.find("[!H](t:-55,-65)CC[!H]    [!H](t:55,65)CC[!H]", true)</code>	fill array x with all non-hydrogen "gauche" interactions (torsions between -55 and -65 degrees or between 55 and 65 degrees).
<code>x = {*}.find("~-r:{*:1}")\$(*0)2-4){*:1}", true)</code>	create an array variable x that lists all RNA (~-r) loops ( {*:1}...{*:1} ) that involve from 2 to 4 un-paired bases ( [\$(*0)2-4] ).
<code>select 1.1; show SMILES</code>	Show the SMILES string for model 1.1; same as <code>print 1.1.find("SMILES")</code>
<code>print {*:A}.find("SMILES",true);</code>	display the sequence for chain A in bioSMILES format:  <pre> // ** Jmol bioSMILES 12.0.RC25    2010-07-14 15:28 1 *// // ** chain A protein 1 *// -p-GRRIQQRRGRGTSTFRAPSHRYKADLEHRKVEDGDV // * 37 *// </pre>
<code>print {selected}.find("SEQUENCE")</code>	returns the Jmol bioSMILES sequence for the specified atoms (same as <code>.find("SMILES",true)</code> ), but without the header comment. Adding an the optional second parameter TRUE adds crosslinking:  <pre> // ** chain A protein 1 *// -p-TTC:1C:2PSIVASNFNVC:3RLPGTPEAIC:3ATYTCG:2IIIPGA // ** TC:1PGDYAN // * 46 *// </pre>
<code>jmolEvaluate("''' + stringAnswer + ''' + find('"'SMILES,'" + stringKey + '"') &gt; 0")</code>	JavaScript call to a Jmol applet to find out if a given SMILES string that a student has entered using a 2D drawing object matches the key. If required stereochemistry indicated in the key is missing in the answer, the result will be FALSE. If the student's answer has unnecessary stereochemistry indicated, the result will be TRUE.

[↑ top](#) [?](#) [search](#) [i](#) [index](#)

### [Jmol command syntax]

In general, commands in Jmol start with a command word and continue with a set of parameters separated by white space and terminated by an end of line character or a semicolon. A backslash just prior to the end of a line indicates a line continuation. However, starting in Jmol 11.8, commands can "wrap" to the next line in a more JavaScript-like or Java-like fashion. In general, any unclosed {, [, or { on a line indicates that the command continues on to the next line. In addition, with [print](#) and [set](#), lines can continue just after or before a mathematical operator such as +, -, \* or /. This results in a more natural line formatting, which has strong similarities to Java and JavaScript. For example:

```
function processInfo(f, i) {
  var pdbid = (i ? f[i] : f)
  load @{"="+pdbid}
```

```
var molinfo = ({protein} == 0 ? "nonprotein"
: {protein and not *.ca} == 0 ? "alpha carbon model"
: {*}[0].model != {*}[1].model ? "multi-model"
: "OK")
```

```
if (molinfo != "OK") {
    print pdbid+" "+molinfo;
    return;
}
```

```
var chaincount = script("select protein; show chain").trim().lines.length
var helixcount = script("show structure").lines.find("HELIX").length
var sheetcount = script("show structure").lines.find("SHEET").length
print pdbid+" "+molinfo
  +" "+(".ca).size
  +" "+chaincount
  +" "+helixcount
```

```
}
+" "+sheetcount
```

See also:  
[\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [case default](#) [echo for if](#) [message reset set switch while](#)

[↑top](#) [🔍search](#) [📘index](#)

[Jmol math]

Jmol Variables  
Global and Local Scope  
Read-Only Variables  
Jmol Math Variable Types  
Jmol Quaternion Math  
Operators and Operands  
Operation Rules

Jmol 11.2 introduces a rich math environment including multiple variable types and a wide variety of [functions](#). This section of the documentation details the use of variables and the sorts of operations that are allowed with them.

Jmol Variables [back](#)

Variables may be assigned using standard mathematical expressions and used throughout Jmol in virtually any script command to substitute for parameters using the syntax @varName or @(math expression):

```
x = 0.2; wireframe @x
zoom @(x * 3);
```

Variables may be combined with ["settings"](#) to adjust parameters:

```
x = bondModeOR; set bondModeOr !x
```

or

```
if (x) {set showBoundingBox x}
```

Variables may be used to extract information from a model:

```
x = {carbon}.bonds.length.min
```

Variables may be used to introduce atom-related property data from external files into the model:

```
x = load("chargeData.txt");select 2.1;data "property_charges @x";select 2.1 and property_charges < 0.5
```

Variables may be inspected using **show x**, **show @x**, or **message @x** where **x** is a variable name. You can use [message](#) or [echo](#) to transmit this information to the user or to the web page via a JavaScript [callback](#) function or use the Jmol.js function `jmolEvaluate()`. If using the Jmol stand-alone application or the signed applet, you can write a variable to a file using the [write](#) command. Starting with Jmol 11.4, variables and math expressions can be checked using the [print](#) command.

Global and Local Scope [back](#)

Variables in Jmol have one of two scopes -- **global** or **local**. Global variables hold values that need to persist across scripts or functions. Local variables override the values previously assigned to the same name for some limited extent. In general, variables defined in Jmol are global variables. A major difference between local and global variables is that only global variables are recorded in the [state](#). It is good programming practice to define variables locally as much as possible, but there are times when global assignments are needed. Just be aware that any global variables created and not destroyed using the [reset](#) command consume memory and may slow script processing. The following rules govern the scope of local variables:

applet localization	all variables in Jmol are localized to a specific applet. Note that this is not necessarily true for <a href="#">functions</a> . Specifically, functions beginning with <b>static_</b> are common to all applets. These static functions, though able to be used by any applet, will not share variables.
script localization	var x = 30 var d = {atomno=1}.xyz.distance({atomno=2}.xyz)
Variables defined using the keyword VAR	

within a script file that is read by the <a href="#">script</a> command are localized to that script and scripts that that script calls.	
function localization  The parameters of a function and any variables defined within a function using VAR are local to that function. The Jmol script on the right will print  <b>a=1 b=2 testing now</b>	a = "testing"; c = "now" function checkX(a, b) { var c = 15; print "a="+a + " b="+ b } checkX(1,2) print "" + a + " " + c
FOR/WHILE localization  Variables defined using VAR within the context of a <a href="#">FOR</a> or <a href="#">WHILE</a> loop will be local to that loop. (Jmol 12.0) The script on the right will print  <b>5 6 7 3</b>	x = 3 for (var x = 5; x < 8; x++) { print x } print x
{ ... } localization  Variables defined using VAR can be localized to a section of a script by bracketing that section of the script with { and }. (Jmol 12.0) This script will print  <b>10 3</b>	x = 3 { var x = 10;print x } print x

Read-Only Variables [back](#)

Some variables have preset meanings, as shown in the table below. A subset of these variables can be used in math expressions. If you create your own variables, their names must not begin with an underscore. Variables starting with underscore are defined by Jmol and can be used but not set in a script. These include:

<b>_animating</b>	whether or not Jmol is currently running through the frames as a result of <b>animation on</b> or <b>animation play</b> (true or false)
<b>_applet</b>	whether or not Jmol is running as an applet (Jmol 11.6)
<b>_atomHovered</b>	the overall atom index of the atom that was most recently hovered over (or -1)
<b>_atomPicked</b>	the overall atom index of the atom that was most recently picked (or -1). Can be used, for example, with <b>select atomIndex = _atomPicked</b>
<b>_currentFileNumber</b>	the number of the currently displayed file, starting with 1 (value will be 0 if more than one file is displayed)
<b>_currentModelNumberInFile</b>	the number of the currently displayed model in its file (or 0 if more than one model is displayed)
<b>_firstFrame</b>	The first frame in the current animation frame range expressed in x.y notation (Jmol 11.4)
<b>_height</b>	the height of the applet or application window in pixels
<b>_lastFrame</b>	The last frame in the current animation frame range expressed in x.y notation (Jmol 11.4)
<b>_memory</b>	the amount of memory allocated to the applet or application
<b>_modelFile</b>	the filename of the model (or "" if more than one model is displayed)
<b>_modelName</b>	the name of the model (or "" if more than one model is displayed)
<b>_modelNumber</b>	the current model number as a <b>string</b> in file.model notation (or "file1.model1 - file2.model2" if more than one model is currently displayed)
<b>_modelTitle</b>	information from the file reader interpreted as a title
<b>_multiTouchServer</b>	indicate whether Jmol is functioning as a <a href="#">SparsH-UI</a> server. (Requires a specialized Jmol SparsH-UI driver) and has successfully connected to a multi-touch device. (Jmol 12.0)
<b>_multiTouchClient</b>	indicates if Jmol is operating as a SparsH-UI client and has connected with the SparsH-UI server (possibly itself).

<b>_pickInfo</b>	information about the last atom picked, including a description of the atom, its atom number, and its x y z coordinates. For example: <b>[GLN]25:A.O/2.1 #175 40.271 8.524 2.615</b>
<b>_signedApplet</b>	whether or not Jmol is running as an signed applet (Jmol 11.6)
<b>_spinning</b>	whether or not the model is currently spinning (true or false). The <code>_spinning</code> variable can be used, for example, to toggle spinning on and off: <b>if (_spinning);spin off;else;spin on;endif;</b>
<b>_version</b>	the version of Jmol expressed as an integer: vvrrxxx for Jmol vv.rr.xxx. For example, for Jmol 11.1.38, <code>_version</code> = 1101038
<b>_width</b>	the width of the applet or application window in pixels

**Jmol Math Variable Types**   [back](#)

Jmol math allows for several distinct variable types, some of which are common types (boolean, integer, decimal, string, serial array, associative array), and some of which are special types of particular use in molecular calculations (point, plane, quaternion, 3x3 matrix, 4x4 matrix, atom bitset, bond bitset). Array types may include any number of any combination of these data types. (Note however, that arrays of arrays are not currently supported). Examples include:

array	<pre>xlist = array(true,3,3.5,"testing",{2,3,3.4,4.5}); xlist = [true,3,3.5,"testing",{2,3,3.4,4.5}]</pre> <p>Arrays may contain a mix of any variable type. However, when they are stored internally, each element of an array is a simple text string. Arrays within arrays are stored as text strings with new-line characters between elements. The square bracket format is available starting with Jmol 11.8.</p>
associative array (Jmol 12.0)	<pre>b = {"test" : 34, "test2" : 45} b["test"] = 35 b["test3"] =["one", 3, 4, 5] print b["test3"][1]</pre> <p>one</p> <p>show b</p> <pre>b = { "test3":["one", 3, 4, 5],"test2":45,"test":35 }</pre> <pre>b := "test2" show b</pre> <pre>b = { "test3":["one", 3, 4, 5],"test":35 }</pre> <p>Associative arrays store information retrievable by string-based keys. The information may be any type, including another associative array. When an element is retrieved, a copy of that array element is retrieved.</p>
atom bitset	<pre>x = {atomno &lt; 30}; x = ({0:28 45 56:62})</pre>
bond bitset	<pre>x = {atomno &lt; 30}.bonds; x = [{0:4 6 9:12}]</pre>
boolean	<pre>isOk = TRUE; if(isOK);print "all is good";end if;</pre>
decimal	<pre>x = {atomno &lt; 10}.x * 10.0;</pre>
integer	<pre>for(var i = 1; i &lt; 10; i = i + 1); print i;end for;</pre>
3x3 matrix	<pre>m = [[1.0,2.0,3.0],[4.0,5.0,6.0],[7.0,8.0,9.0]]; m = axisAngle({1,0,0},30)%-9.</pre>
4x4 matrix	<pre>m = [[1.0,2.0,3.0,4.0],[4.0,5.0,6.0,7.0],[7.0,8.0,9.0,10.0]];</pre>
point	<pre>pt = {3,2,3,3,4};</pre>
plane	<pre>xyPlane = {0 0 1 0};</pre> <p>Planes are defined as <code>[a b c d]</code>, where the equation of the plane is <b>ax + by + cz + d = 0</b>.</p>
quaternion	<pre>q = quaternion({1 0 0}, 30); q = {0.25881904 0.0 0.0 0.9659258};</pre> <p>(see next section) Quaternions are saved internally in Jmol using the same format as for planes, as single-precision floating point</p>

	four-element vectors, with parameter order {x y z w} or {q1 q2 q3 q0}. Ordering the parameters in this way is consistent with <a href="#">Java Quaternion</a> format and allows both quaternions and planes to contain axis information in the first three parameters. The common storage format for planes and quaternions works because typical quaternion operations are not common to operations involving planes.
string	<pre>myLabel = "this is a test";</pre> <p>Starting in Jmol 11.8, strings may be enclosed in either single or double quotes.</p>

Variable types may be combined in mathematical expressions. In general, expressions are evaluated from left to right using standard operator precedence (`*`, `/`, `\` before `+`, `-`; `+`, `-` before AND/OR/NOT).

**Jmol Quaternion Math**   [back](#)

Quaternion math was introduced in Jmol 11.5.43. All quaternions in Jmol are **unit** quaternions, which are four-dimensional vectors that can be used to define the relative rotational aspects of a protein or nucleic acid structure as well as overall molecular orientation. This means that they can be used in a variety of commands, including [navigate](#), [moveto](#), and [rotate](#) to rotate the model or selected atoms of the model around specific axes and to specific orientations. Each quaternion can be thought of as representing a unique axis and angle which will transform a reference frame (either the molecular reference frame or the window reference plane to a given orientation. If **n** is the axis and **theta** is the angle (measured in a right-handed, counter-clockwise direction), then **q(theta/2, n)** = (cos(theta/2), **n** sin(theta/2)). For storage, the vector **n** is broken out into its x, y, z components, giving a total of four numbers, (q0, q1, q2, q3), where q0 = cos(theta/2), q1 = **n<sub>x</sub>** sin(theta/2), q2 = **n<sub>y</sub>** sin(theta/2), and q3 = **n<sub>z</sub>** sin(theta/2). Jmol reports a quaternion as a four-vector with q0 listed last: {q1, q2, q3, q0} in order to be consistent with Java's {x, y, z, w} notation.

The [quaternion\(\)](#) function constructs quaternions using a variety of starting points, including the four numbers q0-q3, [quaternion\(q0, q1, q2, q3\)](#), axis-angle information, [quaternion\({0 0 0 1}, 30\)](#), and 3x3 rotation matrices, [quaternion\(mat\)](#). In addition, quaternions representing the orientation of specific amino acid residues and nucleic acid bases can be constructed automatically based on the setting of the Jmol parameter [quaternionFrame](#), and certain commands and functions such as [show rotation](#) and [script\("show rotation"\)](#) deliver quaternions. These quaternion values can be depicted visually using the [draw](#) and [plot](#) commands and can be listed or saved to a file using the [show](#) and [write](#) commands, respectively.

An interesting feature of quaternions that makes them different from the more common 3x3 rotation matrices is that they can encode rotations up to 720 degrees. For example, quaternion({0 0 1}, 30) = {0.0 0.0 0.25881904 0.9659258} (a 30-degree CCW rotation around the Z axis), while quaternion({0 0 1}, 390) = {0.0 0.0 -0.25881904 -0.9659258} (because sin(195) and cos(195) are both negative). While rotations of 0 degrees, {0 0 1 1}, and 360 degrees, {0 0 -1 -1} represent the same final state, in certain cases they can represent different processes and can thus be significantly different. In addition, quaternion differences (or "derivatives") can be quantified in ways that cannot be done for 3x3 rotation matrices. For example, the mean and standard deviation of a set of quaternions can be determined: **print [a,b,c,d,e].average, print [a,b,c,d,e].stddev**, where a-e are quaternions, providing, for example, the average helical axis for a protein helix structure, or a measure of how "ideal" that helix is.

**Operators and Operands**   [back](#)

Jmol expressions can include standard operators: `+`, `-`, `*`, `/`, `**`(exponentiation, Jmol 12.0) `%`(modulus), and `(&, &&, and)`, or `(|, ||, or)`, `not (|, not)`, and all standard comparison operators. For example:

```
twoPi = 3.14159 * 2;
minBondDistance = minBondDistance * 0.5;
```

In addition, you can set variables to be the number of atoms that match an atom expression. For example:

```
nNC = {_N and /1 and connected(_C)}.size
nAtoms = (*).size;
nCH = {_H} + {_C};
```

Since Jmol math does not include strict typecasting, it uses a relatively complex set of rules to determine the result of operations on mixed variable types. When two different variables are operated upon, the resulting variable type depends upon the operator used, the order of the variables, and sometimes the value of the variables. See [misc/operations.txt](#) for details. In general, when conversion is required for a string, point, plane, bitset, or array, Jmol will attempt to convert it to a variable type compatible with the left-hand operand prior to operating. These conversions generally involve:

string	to boolean	The strings "FALSE", "0", and decimal strings such as "0E1" and "0.0" that equal 0 are converted to FALSE; all other strings are converted to TRUE.
string	to integer	A string evaluating to an integer is converted to that integer; all other strings are converted to 0.
string	to decimal	A string evaluating to a number is converted to that number; all other strings are converted to the decimal value "not a number", or "NaN".

string	to other	Jmol saves all states as simple strings, so certain character sequences are automatically turned back into other variable types when operated upon. Jmol automatically converts:	
		{x y z}	to a point
		{x y z w}	to a plane or quaternion
		((i j k l:m ... ))	to an atom bitset
		[[i j k l:m ... ]]	to a bond bitset
point	to integer	The distance from the point to {0 0 0} rounded DOWN to the nearest integer. Note that this allows rounding a positive decimal number x down to the nearest integer (a "floor" operation) using 0 + {x 0 0}.	
point	to decimal	The distance from the point to {0 0 0}; same as x.distance({0 0 0})	
plane	to integer	The distance from the plane to {0 0 0} rounded DOWN to the nearest integer	
plane	to decimal	The distance from the plane to {0 0 0}; same as x.distance({0 0 0})	
quaternion	to decimal	cotangent(abs(theta)/2), where theta is the angle of the rotation associated with this quaternion	
bitset	to decimal or integer	The number of selected atoms or bonds in the bitset; same as x.size	
array	to decimal or integer	The number of elements in the array; same as x.size	

Operation Rules [back](#)

Rules for operations with the given types include:

+	addition	a + b produces a decimal number EXCEPT:				
		array + b	array	b added to end of a		
		a + array	array	b added to beginning of a		
		integer + b	integer	unless b is a decimal or an array		
		x + plane or quaternion	varies	{x y z} are extracted from the plane or quaternion and then added to x		
		matrix3x3 + matrix3x3	matrix3x3	sum of individual elements of the two matrices		
		matrix3x3 + point	matrix4x4	adds a translation vector to a rotation matrix to give a 4x4 matrix		
		quaternion + (decimal)x	quaternion	addition of x to the angle of rotation associated with this quaternion		
		string + b	string	unless b is an array		
		point + b	point	unless b is an array		
-	subtraction	a - b produces a decimal number EXCEPT:				
		associative array - x	associative array	removes key "x" from the associative array		
		integer - b	integer	unless b is a decimal		
		matrix3x3 - matrix3x3	matrix3x3	difference of individual elements of the two matrices		
		x - plane or quaternion	varies	{x y z} are extracted from the plane or quaternion and then subtracted from x		
		quaternion - (decimal)x	quaternion	subtraction of x to the angle of rotation associated with this quaternion		
		string - integer	integer	string a is converted to integer, then b is subtracted		
		point - b	point	subtraction of {b b b} from point a		
		*	multiplication	a * b produces a decimal number EXCEPT:		
				integer * b	integer	unless b is a decimal
point * b	point			unless b is a point (dot product, producing a decimal)		
quaternion * quaternion	quaternion			quaternion multiplication q2 * q1 results in a composite rotation resulting from first rotating by q1, then by q2.		
quaternion * (decimal)x	quaternion			multiplication of the angle associated with this quaternion by x		

/	division	a / b produces a decimal number EXCEPT:	
		integer / integer	integer
		point / b	point if b is a point, then a is scaled by the magnitude of b; thus a/a when a is a point produces a normalized vector in the direction from {0 0 0} to point a
		quaternion / (decimal)x	quaternion division of the angle associated with this quaternion by x
		quaternion / quaternion	quaternion q2 / q1 = q2 * q1 <sup>-1</sup> (absolute difference in rotation, in the molecular frame)
\	left division	a \ b produces integer division EXCEPT:	
		quaternion \ quaternion	quaternion q1 \ q2 = q1 <sup>-1</sup> * q2 (relative difference in rotation, in the q1 frame)
**	exponentiation	a**b takes a to the power of b; if both a and b are integers, the result is an integer, otherwise the result is a decimal number.	
%	modulus	a % b is fully defined only for integer b and produces an integer EXCEPT:	
		decimal % 0	integer decimal a rounded to nearest integer, with n.5 rounding to (n + 1) and -n.5 rounding to -(n + 1)
		decimal % b	string decimal a rounded to b digits after the decimal point when b > 0; decimal a rounded to b significant digits in scientific notation when b < 0
		matrix4x4 % 1	matrix3x3 extract the 3x3 rotation matrix from a 4x4 rotation/translation matrix
		matrix4x4 % 2	point extract the translational vector from a 4x4 rotation/translation matrix
		quaternion % 0	decimal extract q0
		quaternion % 1	decimal extract q1
		quaternion % 2	decimal extract q2
		quaternion % 3	decimal extract q3
		quaternion (or plane) % 4	point plane normal or quaternion axis; in the case of a plane, the vector from this point to {0 0 0} is directed toward the plane; for quaternions, the axis is defined such that the angle would be between -180 and 180 degrees.
		quaternion % -1	point extract the rotational axis ({q1 q2 q3} or {x y z}) as a point (vector from {0 0 0})
		quaternion % -2	decimal extract the angle in degrees for the rotation associated with this quaternion
		quaternion % -3	point extract the first column of the rotation matrix associated with this quaternion (what {1 0 0} is transformed to)
		quaternion % -4	point extract the second column of the rotation matrix associated with this quaternion (what {0 1 0} is transformed to)
		quaternion % -5	point extract the third column of the rotation matrix associated with this quaternion (what {0 0 1} is transformed to)
		quaternion % {x y z}	point transform {x y z} by the rotation associated with this quaternion
		string % b	string when b > 0, right-justified in a field b characters wide; when b < 0, left-justified in a field b characters wide; no effect when b = 0
		point % b	point generates the unitcell coordinate corresponding to the point, offset from {0 0 0} by {b/1 b/1 b/1}.
		bitset % b	bitset a truncated to first b items; same as a[1][b]
		array % b	array each element treated as a string and justified individually
&&    !	AND/OR/NOT	In Jmol the following are equivalent:	
		AND	& &&
		OR	
		NOT	!
		a AND/OR b as well as NOT b produce a boolean unless both a and b are bitsets, in which case the result is a bitset, or when using !a and a is a quaternion:	
		bitset AND bitset	bitset the intersection of the two bitsets

		bitset OR bitset	bitset	including all selections from both bitsets
		NOT bitset	bitset	the inverse of the bitset, based on the total atom count for an atom bitset or the total bond count for a bond bitset
		!q	quaternion	quaternion inverse, {-x -y -z w }
== !=	equal/not equal	== (or just "=") and != generally convert values to decimal values and then test these values are within 1E-6 of each other. The following cases are exceptions (Jmol 11.5.45):		
		point == point		true if the distance between the points is less than 1E-6.
		plane == plane quaternion == quaternion		true if the four-vector distance between the quaternions or planes is less than 1E-6.

See also:  
[\[Jmol command syntax\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [case default echo for if message reset set switch while](#)

[↑top](#) [🔍search](#) [📘index](#)

[Jmol parameters]

General Parameters  
Set-Only Parameters  
Deprecated Parameters  
Reserved Names

Many parameters in Jmol can be [set](#), and in Jmol 11.2 and 11.4 many (but not all) may also be checked using Jmol math.

**General Parameters** [back](#)

The following 224 parameters may be SET and also checked using Jmol math. Items without a link are either undocumented at this time or for later versions of Jmol than the one you have selected for this documentation display.

<a href="#">allowEmbeddedScripts</a>	<a href="#">allowGestures</a>	<a href="#">allowKeyStrokes</a>	<a href="#">allowModelKit</a>	<a href="#">allowMultiTouch</a>
<a href="#">allowRotateSelected</a>	<a href="#">ambientPercent</a>	<a href="#">animationFPS</a>	<a href="#">animFrameCallback</a>	<a href="#">antialiasDisplay</a>
<a href="#">antialiasImages</a>	<a href="#">antialiasTranslucent</a>	<a href="#">appendNew</a>	<a href="#">appletProxy</a>	<a href="#">applySymmetryToBonds</a>
<a href="#">atomPicking</a>	<a href="#">atomTypes</a>	<a href="#">autobond</a>	<a href="#">autoFPS</a>	<a href="#">autoLoadOrientation</a>
<a href="#">axesMode</a>	<a href="#">axesMolecular</a>	<a href="#">axesScale</a>	<a href="#">axesUnitcell</a>	<a href="#">axesWindow</a>
<a href="#">axis1Color</a>	<a href="#">axis2Color</a>	<a href="#">axis3Color</a>	<a href="#">backgroundColor</a>	<a href="#">backgroundModel</a>
<a href="#">bondModeOr</a>	<a href="#">bondPicking</a>	<a href="#">bondRadiusMilliAngstroms</a>	<a href="#">bondTolerance</a>	<a href="#">bboxColor</a>
<a href="#">cameraDepth</a>	<a href="#">cartoonBaseEdges</a>	<a href="#">cartoonRockets</a>	<a href="#">chainCaseSensitive</a>	<a href="#">colorRasmol</a>
<a href="#">currentLocalPath</a>	<a href="#">dataSeparator</a>	<a href="#">debug</a>	<a href="#">debugScript</a>	<a href="#">defaultAngleLabel</a>
<a href="#">defaultColorScheme</a>	<a href="#">defaultDirectory</a>	<a href="#">defaultDistanceLabel</a>	<a href="#">defaultDrawArrowScale</a>	<a href="#">defaultLattice</a>
<a href="#">defaultLoadScript</a>	<a href="#">defaults</a>	<a href="#">defaultTorsionLabel</a>	<a href="#">defaultTranslucent</a>	<a href="#">defaultVDW</a>
<a href="#">delayMaximumMs</a>	<a href="#">diffusePercent</a>	<a href="#">dipoleScale</a>	<a href="#">disablePopupMenu</a>	<a href="#">displayCellParameters</a>
<a href="#">dotDensity</a>	<a href="#">dotScale</a>	<a href="#">dotsSelectedOnly</a>	<a href="#">dotSurface</a>	<a href="#">dragSelected</a>
<a href="#">drawHover</a>	<a href="#">drawPicking</a>	<a href="#">dynamicMeasurements</a>	<a href="#">edsUriCutoff</a>	<a href="#">edsUriFormat</a>
<a href="#">ellipsoidArcs</a>	<a href="#">ellipsoidAxes</a>	<a href="#">ellipsoidAxisDiameter</a>	<a href="#">ellipsoidball</a>	<a href="#">ellipsoidDotCount</a>
<a href="#">ellipsoiddots</a>	<a href="#">ellipsoidfill</a>	<a href="#">exportDrivers</a>	<a href="#">fontCaching</a>	<a href="#">fontScaling</a>
<a href="#">forceAutoBond</a>	<a href="#">fractionalRelative</a>	<a href="#">greyscaleRendering</a>	<a href="#">hbondsAngleMinimum</a>	<a href="#">hbondsBackbone</a>
<a href="#">hbondsDistanceMaximum</a>	<a href="#">hbondsSolid</a>	<a href="#">helixStep</a>	<a href="#">helpPath</a>	<a href="#">hermiteLevel</a>
<a href="#">hideNameInPopup</a>	<a href="#">hideNavigationPoint</a>	<a href="#">hideNotSelected</a>	<a href="#">highResolution</a>	<a href="#">historyLevel</a>
<a href="#">hoverCallback</a>	<a href="#">hoverDelay</a>	<a href="#">hoverLabel</a>	<a href="#">imageState</a>	<a href="#">isKiosk</a>
<a href="#">isosurfacePropertySmoothing</a>	<a href="#">justifyMeasurements</a>	<a href="#">language</a>	<a href="#">loadAtomDataTolerance</a>	<a href="#">loadAtomDataTolerance</a>
<a href="#">loadFormat</a>	<a href="#">loadStructCallback</a>	<a href="#">logCommands</a>	<a href="#">logFile</a>	<a href="#">logGestures</a>
<a href="#">logLevel</a>	<a href="#">measureAllModels</a>	<a href="#">measurementLabels</a>	<a href="#">measurements</a>	<a href="#">measurementUnits</a>
<a href="#">messageCallback</a>	<a href="#">messageStyleChime</a>	<a href="#">minBondDistance</a>	<a href="#">minimizationCriterion</a>	<a href="#">minimizationRefresh</a>
<a href="#">minimizationSilent</a>	<a href="#">minimizationSteps</a>	<a href="#">modelKitMode</a>	<a href="#">mouseDragFactor</a>	<a href="#">mouseWheelFactor</a>
<a href="#">navFPS</a>	<a href="#">navigateSurface</a>	<a href="#">navigationDepth</a>	<a href="#">navigationMode</a>	<a href="#">navigationPeriodic</a>
<a href="#">navigationSlab</a>	<a href="#">navigationSpeed</a>	<a href="#">navX</a>	<a href="#">navY</a>	<a href="#">navZ</a>

<a href="#">pdbGetHeader</a>	<a href="#">pdbSequential</a>	<a href="#">percentVdwAtom</a>	<a href="#">perspectiveDepth</a>	<a href="#">perspectiveModel</a>
<a href="#">phongExponent</a>	<a href="#">pickCallback</a>	<a href="#">picking</a>	<a href="#">pickingSpinRate</a>	<a href="#">pickingStyle</a>
<a href="#">pickLabel</a>	<a href="#">pointGroupDistanceTolerance</a>	<a href="#">pointGroupLinearTolerance</a>	<a href="#">preserveState</a>	<a href="#">propertyAtomNumberColumnCount</a>
<a href="#">propertyAtomNumberField</a>	<a href="#">propertyColorScheme</a>	<a href="#">propertyDataColumnCount</a>	<a href="#">propertyDataField</a>	<a href="#">quaternionFrame</a>
<a href="#">rangeSelected</a>	<a href="#">refreshing</a>	<a href="#">repaintWaitMs</a>	<a href="#">resizeCallback</a>	<a href="#">ribbonAspectRatio</a>
<a href="#">ribbonBorder</a>	<a href="#">rocketBarrels</a>	<a href="#">rotationRadius</a>	<a href="#">saveProteinStructureState</a>	<a href="#">scaleAngstromsPerInch</a>
<a href="#">scriptQueue</a>	<a href="#">scriptReportingLevel</a>	<a href="#">selectAllModels</a>	<a href="#">selectHetero</a>	<a href="#">selectHydrogen</a>
<a href="#">sheetSmoothing</a>	<a href="#">showAxes</a>	<a href="#">showBoundingBox</a>	<a href="#">showFrank</a>	<a href="#">showHiddenSelectionHalos</a>
<a href="#">showHydrogens</a>	<a href="#">showKeyStrokes</a>	<a href="#">showMeasurements</a>	<a href="#">showMultipleBonds</a>	<a href="#">showNavigationPointAlways</a>
<a href="#">showScript</a>	<a href="#">showUnitcell</a>	<a href="#">slabByAtom</a>	<a href="#">slabByMolecule</a>	<a href="#">slabEnabled</a>
<a href="#">smallMoleculeMaxAtoms</a>	<a href="#">smartAromatic</a>	<a href="#">solventProbe</a>	<a href="#">solventProbeRadius</a>	<a href="#">specularExponent</a>
<a href="#">specularPercent</a>	<a href="#">specularPower</a>	<a href="#">spinFPS</a>	<a href="#">spinX</a>	<a href="#">spinY</a>
<a href="#">spinZ</a>	<a href="#">ssBondsBackbone</a>	<a href="#">stateVersion</a>	<a href="#">statusReporting</a>	<a href="#">stereoDegrees</a>
<a href="#">strandCountForMeshRibbon</a>	<a href="#">strandCountForStrands</a>	<a href="#">strutDefaultRadius</a>	<a href="#">strutLengthMaximum</a>	<a href="#">strutsMultiple</a>
<a href="#">strutSpacing</a>	<a href="#">syncMouse</a>	<a href="#">syncScript</a>	<a href="#">traceAlpha</a>	<a href="#">unitCellColor</a>
<a href="#">useMinimizationThread</a>	<a href="#">useNumberLocalization</a>	<a href="#">userColorScheme</a>	<a href="#">vectorScale</a>	<a href="#">vibrationPeriod</a>
<a href="#">vibrationScale</a>	<a href="#">visualRange</a>	<a href="#">waitForMoveTo</a>	<a href="#">windowCentered</a>	<a href="#">wireframeRotation</a>
<a href="#">zoomEnabled</a>	<a href="#">zoomLarge</a>	<a href="#">zShade</a>	<a href="#">zShadePower</a>	

**Set-Only Parameters** [back](#)

The following 10 parameters may be SET but because of their complexity or context cannot be checked using Jmol math.

<a href="#">set axesColor</a>
<a href="#">set echo</a>
<a href="#">set formalCharge</a>
<a href="#">set labelAlignment</a>
<a href="#">set labelAtom</a>
<a href="#">set labelFront</a>
<a href="#">set labelGroup</a>
<a href="#">set labelOffset</a>
<a href="#">set labelPointer</a>
<a href="#">set labelToggle</a>

**Deprecated Parameters** [back](#)

The following 47 parameters have been deprecated.

<a href="#">set ambient</a>	see <a href="#">set ambientPercent</a>
<a href="#">set axes</a>	see <a href="#">set axesMode</a>
<a href="#">set background</a>	see <a href="#">set backgroundColor</a>
<a href="#">set bond</a>	see <a href="#">set showMultipleBonds</a>
<a href="#">set bondmode</a>	see <a href="#">set bondModeOr</a>
<a href="#">set bonds</a>	see <a href="#">set showMultipleBonds</a>
<a href="#">set bbox</a>	see <a href="#">bbox</a>
<a href="#">set charge</a>	see <a href="#">set formalCharge</a>
<a href="#">set color</a>	see <a href="#">set defaultColorScheme</a>
<a href="#">set colour</a>	see <a href="#">set defaultColorScheme</a>
<a href="#">set defaultcolors</a>	see <a href="#">set defaultColorScheme</a>
<a href="#">set diffuse</a>	see <a href="#">set diffusePercent</a>
<a href="#">set display</a>	see <a href="#">selectionHalos ON/OFF</a>
<a href="#">set fontsize</a>	see <a href="#">font labels</a>
<a href="#">set frank</a>	see <a href="#">frank ON/OFF</a>
<a href="#">set hbond</a>	see <a href="#">set hbondsBackbone</a> and <a href="#">set hbondsSolid</a>
<a href="#">set hbonds</a>	see <a href="#">set hbondsBackbone</a> and <a href="#">set hbondsSolid</a>
<a href="#">set hetero</a>	see <a href="#">set selectHetero</a>
<a href="#">set history</a>	see <a href="#">history</a>
<a href="#">set hydrogen</a>	see <a href="#">set selectHydrogen</a>

set hydrogens see [set selectHydrogen](#)  
set label see [label](#) and [set \(labels\)](#)  
set labels see [label](#) and [set \(labels\)](#)  
set measure see [set measurements](#) and [set measurementLabels](#) and [set measurementUnits](#)  
set measurement see [set measurements](#) and [set measurementLabels](#) and [set measurementUnits](#)  
set measurementNumbers see [set measurementLabels](#)  
set measures see [set measurements](#) and [set measurementLabels](#) and [set measurementUnits](#)  
set monitor see [set measurements](#) and [set measurementLabels](#) and [set measurementUnits](#)  
set monitors see [set measurements](#) and [set measurementLabels](#) and [set measurementUnits](#)  
set radius see [set solventProbeRadius](#)  
set scale3d see [set scaleAngstromsPerInch](#)  
set selectionhalo see [selectionHalos ON/OFF](#)  
set selectionHalos see [selectionHalos ON/OFF](#)  
set showSelections see [set selectionHalos](#)  
set solvent see [set solventProbe](#)  
set specPercent see [set specularPercent](#)  
set specpower see [set specularPower](#) and [set specularExponent](#)  
set specular see [set specular](#) and [set specularPercent](#)  
set spin see [set spinX](#), [set spinY](#), [set spinZ](#), and [set spinFPS](#)  
set ssbond see [set ssBondsBackbone](#)  
set ssbonds see [set ssBondsBackbone](#)  
set strand see [strandCountForMeshRibbon](#) and [set strandCountForStrands](#)  
set strandCount see [strandCountForMeshRibbon](#) and [set strandCountForStrands](#)  
set strands see [strandCountForMeshRibbon](#) and [set strandCountForStrands](#)  
set timeOut see [timeout](#)  
set toggleLabel see [set labelToggle](#)  
set unitcell see [unitcell](#)

Reserved Names [back](#)

In addition to all command names, the following 7 names are reserved and should be avoided.

axesOrientationRasmol  
property  
testFlag1  
testFlag2  
testFlag3  
testFlag4  
zeroBasedXyzRasmol

See also:

[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [case default echo for if message reset set switch while](#)

[↑top](#) [🔍search](#) [📘index](#)

[atom expressions]

Atom selectors  
Functions  
RasMol biomolecular residue specifications  
Wildcards  
Atom names for other file types

An increasing number of commands, including [center](#), [connect](#), [define](#), [dipole](#), [display](#), [draw](#), [isosurface](#), [measure](#), [polyhedra](#), [restrict](#), and [select](#) take for parameters one or more expressions that represent collections of atoms in one or more models. All terms can be preceded by the keyword NOT and joined by AND, OR, or XOR.

general terms	all, bonded, clickable, none, selected, visible
---------------	---

file.model	as, for example, <b>select 3.2</b> , a specific model in a specific file. Note that <b>select 3.0</b> selects all atoms in all models of the third file of the most recent <a href="#">load</a> .																																															
subset	the currently defined <a href="#">subset</a> . Note that if a subset is currently defined, then <b>select/display all</b> is the same as <b>select/display subset</b> , <b>restrict none</b> is the same as <b>restrict not subset</b> . In addition, <b>select not subset</b> selects nothing.																																															
unitcell	atoms within the current <a href="#">unitcell</a> , which may be offset. This includes atoms on the faces and at the vertices of the unitcell.																																															
chemical elements	element_name (including "deuterium and tritium"), _Xx (an element symbol preceded by underscore, possibly with isotope number listed, such as _Cu, _Fe, _2H, _31P)																																															
isaromatic	atoms connected with the AROMATIC, AROMATICSINGLE, or AROMATICDOUBLE bond types (Jmol 11.3.29)																																															
non-protein groups	carbohydrate, dna, hetero, ions (specifically the PDB designations "PO4" and "SO4"), ligand (hetero and not solvent), nucleic, purine, pyrimidine, rna, sidechain																																															
protein residues	<table><tr><td>acidic</td><td>ASP, GLU</td></tr><tr><td>acyclic</td><td>amino and not cyclic</td></tr><tr><td>aliphatic</td><td>ALA GLY ILE LEU VAL</td></tr><tr><td>amino</td><td>all twenty standard amino acids, plus ASX, GLX, UNK</td></tr><tr><td>aromatic</td><td>HIS PHE TRP TYR (see also "isaromatic" for aromatic bonds)</td></tr><tr><td>basic</td><td>ARG, HIS, LYS</td></tr><tr><td>buried</td><td>ALA CYS ILE LEU MET PHE TRP VAL</td></tr><tr><td>charged</td><td>acidic or basic</td></tr><tr><td>cyclic</td><td>HIS PHE PRO TRP TYR</td></tr><tr><td>helix</td><td>secondary structure-related</td></tr><tr><td>hetero</td><td>PDB atoms designated as HETATM</td></tr><tr><td>hydrophobic</td><td>ALA GLY ILE LEU MET PHE PRO TRP TYR VAL</td></tr><tr><td>large</td><td>ARG GLU GLN HIS ILE LEU LYS MET PHE TRP TYR</td></tr><tr><td>medium</td><td>ASN ASP CYS PRO THR VAL</td></tr><tr><td>negative</td><td>acidic</td></tr><tr><td>neutral</td><td>amino and not (acidic or basic)</td></tr><tr><td>polar</td><td>amino and not hydrophobic</td></tr><tr><td>positive</td><td>basic</td></tr><tr><td>protein</td><td>defined as a group that contains PDB atom designations C, N, and CA</td></tr><tr><td>sheet</td><td>secondary structure-related</td></tr><tr><td>small</td><td>ALA GLY SER</td></tr><tr><td>surface</td><td>amino and not buried</td></tr><tr><td>turn</td><td>secondary structure-related</td></tr></table>		acidic	ASP, GLU	acyclic	amino and not cyclic	aliphatic	ALA GLY ILE LEU VAL	amino	all twenty standard amino acids, plus ASX, GLX, UNK	aromatic	HIS PHE TRP TYR (see also "isaromatic" for aromatic bonds)	basic	ARG, HIS, LYS	buried	ALA CYS ILE LEU MET PHE TRP VAL	charged	acidic or basic	cyclic	HIS PHE PRO TRP TYR	helix	secondary structure-related	hetero	PDB atoms designated as HETATM	hydrophobic	ALA GLY ILE LEU MET PHE PRO TRP TYR VAL	large	ARG GLU GLN HIS ILE LEU LYS MET PHE TRP TYR	medium	ASN ASP CYS PRO THR VAL	negative	acidic	neutral	amino and not (acidic or basic)	polar	amino and not hydrophobic	positive	basic	protein	defined as a group that contains PDB atom designations C, N, and CA	sheet	secondary structure-related	small	ALA GLY SER	surface	amino and not buried	turn	secondary structure-related
acidic	ASP, GLU																																															
acyclic	amino and not cyclic																																															
aliphatic	ALA GLY ILE LEU VAL																																															
amino	all twenty standard amino acids, plus ASX, GLX, UNK																																															
aromatic	HIS PHE TRP TYR (see also "isaromatic" for aromatic bonds)																																															
basic	ARG, HIS, LYS																																															
buried	ALA CYS ILE LEU MET PHE TRP VAL																																															
charged	acidic or basic																																															
cyclic	HIS PHE PRO TRP TYR																																															
helix	secondary structure-related																																															
hetero	PDB atoms designated as HETATM																																															
hydrophobic	ALA GLY ILE LEU MET PHE PRO TRP TYR VAL																																															
large	ARG GLU GLN HIS ILE LEU LYS MET PHE TRP TYR																																															
medium	ASN ASP CYS PRO THR VAL																																															
negative	acidic																																															
neutral	amino and not (acidic or basic)																																															
polar	amino and not hydrophobic																																															
positive	basic																																															
protein	defined as a group that contains PDB atom designations C, N, and CA																																															
sheet	secondary structure-related																																															
small	ALA GLY SER																																															
surface	amino and not buried																																															
turn	secondary structure-related																																															
protein-related	alpha, backbone, base, mainchain, sidechain (backbone and mainchain are synonymous), spine (*CA, *N, *C for proteins; *P, *.O3*, *.O5*, *.C3*, *.C4*, *.C5 for nucleic acids; Jmol 12.0)																																															
solvent-related	solvent, PDB "HOH", water, also the connected set of H-O-H in any model																																															

The comparison operators <, <=, =, >, >=, and != operate with may keywords. These are summarized under [atomproperties">](#).

Atom selectors [back](#)

An atom expression is simply a list of atoms. Starting with Jmol 11.2 you can select a single atom or a range of atoms from an atom expression. The way to do this is simply to suround the atom expression with parentheses and follow it with one or two numbers in brackets: **select (carbon)[3][5]**. This says, "Select the third through fifth carbon atoms." If the second selector is not present, then only a single atom is selected; the selector [0] indicates the last atom in the set, and negative numbers count back from that atom. Thus, **select (\*)[0]** selects the last atom, and **select (carbon and 2.3)[-1][0]** selects the last two carbon atoms in model 2.3. Atom selectors can be used for any expression embedded in another command. In that case an additional set of parentheses or braces is required around the whole expression: **measure {(L\_O)[1]} {(L\_O)[2]}**.

Functions [back](#)

The following functions are also supported.

CONNECTED()	allows for selection of specific atoms based on their connectivity to other atoms. The general format is:
-------------	---

	<b>connected([optional min # bonds], [optional max # bonds], [optional bond type], [optional atom expression])</b> Bond type may be any described for <a href="#">connect</a> . See <a href="#">groups.txt</a> for many examples of using connected() with <a href="#">define</a>
SUBSTRUCTURE()	atoms within a given substructure of the model. The <b>substructure()</b> function takes a quoted <a href="#">smiles string</a> for its argument. For Jmol 12.0, see also the math function <b>within("SMILES"....)</b> . (Note: aromatic ring SMILES were not supported until Jmol 12.0.)
WITHIN(setName,atomExpression)	any atom within a given set. The setName can be any one of the words <b>BOUNDBOX</b> , <b>CHAIN</b> , <b>ELEMENT</b> , <b>GROUP</b> , <b>MODEL</b> , <b>MOLECULE</b> , <b>POLYMER</b> (Jmol 12.0), <b>SITE</b> , or <b>STRUCTURE</b> , or it can be a protein or nucleic acid sequence expressed in single-letter notation surrounded by quotation marks as, for example, "GCCCTT" or "MAACYXV" (in which case the sequence is found within the expression). (SITE refers to all crystallographic sites common to the specified atom set; BOUNDBOX refers to the smallest box containing the atom set.) Additional options, including "BASEPAIR", "SMILES", and "SMARTS", are discussed below.
WITHIN(distance, withinAllModels, atomExpression)	any atom within the specified distance of any atom in the atomExpression. The optional TRUE/FALSE flag withinAllModels (by default FALSE) may be set TRUE to allow finding atoms in one model that may be within some distance of another model. (Jmol 11.4) Starting with Jmol 11.8, if the distance is negative, then the operation applies to all atoms having normalize unit cell coordinates within -distance of the designated atoms.
WITHIN(distance, {x y z})	(New in Jmol 11.1.12) any atom within the specified distance of the given fractional or Cartesian coordinate. Starting with Jmol 11.8, if the distance is negative, then the operation applies to all atoms having normalize unit cell coordinates within -distance of the designated atoms.
WITHIN(nResidues, GROUP, {atoms})	groups that are within a given number of residues of a specified group of atoms. (Jmol 12.0)
WITHIN(0,planeType, planeDesignation)	selects for any atoms within 0.01 Angstroms of a plane. If planeType is HKL, then planeDesignation is in the form {h k l}, where h, k, and l are Miller indices. If planeType is PLANE, then planeDesignation should be of the form @ {plane(a,b,c)}, where a, b, and c are atom expressions or coordinates.
WITHIN(distance,planeType, planeDesignation)	selects for atoms within the given distance in Angstroms from the plane. Positive distances are on one side; negative distances are on the other side. Experimentation may be necessary to determine which side is which for these purposes. In all cases the atoms in the plane itself (within 0.01 Angstroms of the plane on either side) are included.
WITHIN(ATOMNAME,"aa,bb,ccc")	any atom having a listed atom name (Jmol 11.8)
WITHIN(ATOMTYPE, "atomType,atomType,...")	selects for atoms of one or more atom type. Atom type is defined in certain file types, including MOL2 model files and AMBER topology files. For other file types, atom types are the same as atom names. For example, <b>select within(ATOMTYPE,"HW,OW")</b> selects all water atoms an AMBER topology file. (Jmol 11.8)
WITHIN(BASEPAIR("XY..."))	(Jmol 12.0) finds all atoms within hydrogen-bonded DNA or RNA basepairs. Any number of pairs can be indicated. For example, <b>display within(BASEPAIR,"GCAU")</b> would select only G-C and A-U pairs. (Note that the RasMol-derived predefined sets "gc" and "at" refer simply to "G or C" and "A or T", respectively, and do not relate to base pairing.)
WITHIN(BOUNDBOX)	(Jmol 11.3.62) selects all atoms within the currently defined <a href="#">boundbox</a>
WITHIN(BRANCH,{first atom}, {second atom})	selects the second atom and all atoms in the molecular branch starting with the second atom but not including the first atom. (Jmol 11.6)
WITHIN(HELIX)	Selects groups that would be selected using <b>select helix</b> but are not at either end of a helix section. (Jmol 11.8)
WITHIN(SEQUENCE,"sequence")	a protein or nucleic acid sequence expressed in single-letter notation surrounded by quotation marks as, for example, "GCCCTT" or "MAACYXV" (the entire sequence must be found; as indicated above, the keyword SEQUENCE is optional).
WITHIN(SHEET)	Selects groups that would be selected using <b>select sheet</b> but are not at either end of a sheet section. (Jmol 11.8)
WITHIN(SMARTS,"smartsString")	all atoms that conform to the given SMARTS string are found. When used as a math function, this method returns a list of all matching sets of atoms; when used in a selection context (SELECT, DISPLAY, HIDE, etc.), all matching atoms are returned. Only hydrogen atoms that are explicitly indicated as [H] are returned. Extensive details on Jmol 3D-SEARCH SMARTS capability may be found <a href="#">on-line</a> . (Jmol 12.0)
WITHIN(SMILES,"smilesString")	all atoms that conform to the given <a href="#">SMILES</a> string are found. When used as a math function, this method returns a list of all matching sets of atoms, including any indicated hydrogen atoms or hydrogen atoms required to complete the valence on an atom. When used in a selection context (SELECT, DISPLAY, HIDE, etc.), all matching atoms are returned. Note that for substructure searches, WITHIN(SMARTS,"smartsString") is recommended. (Jmol 12.0)

RasMol biomolecular residue specifications [back](#)

The general specification of atoms in PDB "residues" follows the method used in RasMol. While the order of specifiers is somewhat flexible, the following order is generally applicable:

[residueType]seqRange ^insertionCode :chainLetter .atomName %altLoc /modelNumber

[residueType]	[ALA] [G] [2E1] [L??] When used without any other specifiers it is possible <b>in some but not all</b> cases to leave off the brackets around the residue type. However, leaving off the brackets is not recommended and is <b>known to fail</b> when the residue type begins with a number.
seqRange	1 1-30 40- Note that ranges refer to physical ranges of data in the file. If residues corresponding to both the starting and ending residue numbers are not present in the file, selection returns no atoms. If residues with numbers between the starting and ending numbers are out of place in the file -- not physically between those two file positions -- they will not be included in the selection. If there is a desire to include such residues, or the selection should allow starting or ending residues to not be present, then use the <b>resno</b> comparison method instead. In this case, for example: <b>select resno &gt;=1 and resno &lt;=30 or select resno &gt;= 40</b> .
^insertionCode	^A ^B ^?
:chainLetter	:A :B :?
.atomName	.Ca .C? .? .??
%altLoc	%1 %A %?
/modelNumber	/1 /2 /* refer to the number on the MODEL record in multimodel PDB files or the sequential number of the model in the file otherwise. When multiple files are loaded, these numbers refer to the file number, indicating "all models in that file." Specific models in specific files can be specified using a decimal notation: <b>file.model</b> as, for example, <b>select *.CA/2.1</b> -- all alpha carbons in the first model of the second file listed in the <a href="#">load</a> command (Jmol 11.1).

Wildcards [back](#)

Unspecified components of the atom specification are indicated in some cases using a question mark and in others using an asterisk. The wildcard " " can be used in place of [residueType]seqRange to indicate "any." For example: **select \*.CA**. Wildcards can be used elsewhere in the specification, but it is preferred simply to not include a specifier altogether. Thus, **select [ALA].\*** is the same as **select [ALA]**. Note that in the case of PDB files and MOL2 files with residues indicated, " " may be used in the form "x" only in the case of residue names, not atom names. Thus, **select AS\*** selects aspartate and asparagine. When used for an atom, for example, with the unmediated PDB file 1bxx **select A.O?\*** the " " is not wild and selects atoms A.O1\* and A.O4\*. (In remediated PDB files, this " " becomes a single quote or "prime" character -- AO1', AO4'.) For other file types, " " can be used at the end of an atom name fragment. Asterisks cannot be used in place of insertionCode or altLoc.

Question marks are used to indicate "some character": **select \*.C??**. Note that the number of question marks is significant. "?? " only finds atoms with single-letter names such as "O" and "C"; "???" finds atoms with single-letter or double-letter names. The specification :?, ^?, and %? mean SOME chain, SOME insertion code or SOME alternate location; use :, ^, and % alone to indicate "atoms without chain indication," "atoms without insertion code," and "atoms without alternate location," respectively.

Starting with Jmol 11.8, you can use \? to match an actual ? in an atom name. For instance, if there are two atoms, one with the name "O1" and one with the name "O1?" then **select O1?** will select both atoms, but **select O1\?** will select only the second atom. You cannot use \\* to escape an actual \* in an atom name.

Atom names for other file types [back](#)

Atom names can also be used for some non-PDB file types. For example, in CIF files, the atom\_site\_label field is used for the atom name. If an atom has the label "C34" you can select it using **select \*.C34** or **select C34** or even **select C\***. Note that in this case, the wildcard " " is no problem, since non-PDB file types do not include residue names, which might conflict with atom names. Similarly, Jaguar, NWChem, Tripos MOL2, Wavefunction Odyssey, SHELX, and Wavefunction Spartan files list atoms as "H3" and "O2". Atoms for these file types can be selected using these names, and the names can be displayed in [labels](#) using the format code %a. For file types such as XYZ that do not indicate a number with the atom symbol, Jmol constructs an atom name from the element symbol and the sequential number of the atom in the file.

See also:

[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom properties\]](#) [\[functions\]](#) [\[plane expressions\]](#) [case default echo for if message reset set switch while](#)

[atom properties]

Over 60 atom properties can be selected or retrieved from model data, and many of these can be set as well. The older, more limited Rasmol notation **[group].atomName^insertion:chain%altloc** can still be used, but equally well one can combine any subsets of those using a more natural notation. For example, **select group="ARG" and atomname="CA" and chain="A"** is equivalent to **select [ARG].CA:A**. Mostly, the newer **xxxx=y** notation generalizes better in terms of additional parameters not unique to Rasmol. The capability of setting atom properties using **{atom expression}.xxxx = y** was introduced in Jmol 11.2. The **label %[xxx]** notation was introduced in

Jmol 11.8; prior to that version a more limited range of properties could be put in a **label** using % and a single- or double-character code. Within labels, %% can be used to "escape" the percent sign and have it not be a special character. The full list of atom properties is given below.

property	select xxx=y	label %[xxx]	label %x	print {*}.xxx = y	{*}.xxx = y	description
adpmax	yes	yes		yes		the maximum anisotropic displacement parameter for the selected atom
adpmin	yes	yes		yes		the minimum anisotropic displacement parameter for the selected atom
altloc	yes	yes	A	yes		PDB alternate location identifier
atomID	yes	yes		yes	yes	special atom IDs for PDB atoms assigned by Jmol
atomIndex	yes	yes	D	yes		atom 0-based index; a unique number for each atom regardless of the number of models loaded
atomName	yes	yes	a	yes	yes	atom name
atomno	yes	yes	i	yes		sequential number
atomType	yes	yes	B	yes	yes	atom type (mol2, AMBER files) or atom name (other file types) [Jmol 11.8]
atomX	yes	yes	x	yes	yes	Cartesian X coordinate (or just X [Jmol 12.0])
atomY	yes	yes	y	yes	yes	Cartesian Y coordinate (or just Y [Jmol 12.0])
atomZ	yes	yes	z	yes	yes	Cartesian Z coordinate (or just Z [Jmol 12.0])
bondcount	yes	yes		yes		covalent bond count
cell	yes					crystallographic unit cell, expressed either in lattice integer notation (111-999) or as a coordinate in ijk space, where {1 1 1} is the same as 555. The positioning is either absolute, based on the original file coordinates, or relative to the current setting of <b>unitcell</b> , as determined by <b>set fractionalRelative</b> . ANDing two cells, for example <b>select cell=555</b> and <b>cell=556</b> , selects the atoms on the common face. (Note: in the specific case of CELL, only "=" is allowed as a comparator.)
configuration	yes					Only in the context <b>{configuration=n}</b> , this option selects the set of atoms with either no ALTLOC specified or those atoms having this index into the array of altlocs within its model. So, for example, if the model has altloc "A" and "B", <b>select configuration=1</b> is equivalent to <b>select altloc=""</b> or <b>altloc="A"</b> , and <b>print {configuration=2}</b> is equivalent to <b>print {altloc=""</b> or <b>altloc="B"</b> . Configuration 0 is "all atoms in a model having configurations", and an invalid configuration number gives no atoms. (Note: in the specific case of CONFIGURATION, only "=" is allowed as a comparator.) [Jmol 11.10]
property	select xxx=y	label %[xxx]	label %x	print {*}.xxx = y	{*}.xxx = y	description
chain	yes	yes	c/s	yes		protein chain
color	yes	yes		yes	yes	the atom color
covalent	yes	yes		yes		covalent bonding radius
element	yes	yes	e	yes	yes	element symbol. The value of this parameter depends upon the context. Used with <b>select structure=x</b> , x can be either the quoted element symbol, "H", "He", "Li", etc. or atomic number. In all other contexts, the value is the element symbol. When the atom is a specific isotope, the string will contain the isotope number -- "13C", for example.
elemno	yes	yes	l (e/)	yes	yes	atomic element number
eta/theta	yes	yes		yes		Based on Carlos M. Duarte, Leven M. Wadley, and Anna Marie Pyle, RNA structure comparison, motif search and discovery using a reduced representation of RNA conformational space, <b>Nucleic Acids Research</b> , 2003, Vol. 31, No. 16 4755-4761. The parameter eta is the C4'[i]-1]-P[i]-C4'[i]-P[i+1] dihedral angle; theta is the P[i]-C4'[i]-P[i+1]-C4'[i+1] dihedral angle. Both are measured on a 0-360 degree scale because they are commonly near 180 degrees. Using the commands <b>plot PROPERTIES eta theta resno; select visible;wireframe only</b> one can create these authors' "RNA worm" graph.
file	yes	yes		yes		file number containing this atom
formalCharge	yes	yes	C	yes	yes	formal charge
format	yes	yes		yes	yes	format (label) of the atom. (Jmol 11.8)

fXYZ		yes		yes	yes	fractional XYZ coordinates
fX	yes	yes	X	yes	yes	fractional X coordinate
fY	yes	yes	Y	yes	yes	fractional Y coordinate
fZ	yes	yes	Z	yes	yes	fractional Z coordinate
fxyz		yes		yes	yes	fractional XYZ coordinates in the <b>unitcell</b> coordinate system
fux	yes	yes		yes	yes	fractional X coordinate in the unitcell coordinate system
fuy	yes	yes		yes	yes	fractional Y coordinate in the unitcell coordinate system
fuz	yes	yes		yes	yes	fractional Z coordinate in the unit cell coordinate system
group	yes	yes	n	yes		3-letter residue code
group1	yes	yes	m	yes		single-letter residue code (amino acids only)
groupID	yes	yes		yes		group ID number: A unique ID for each amino acid or nucleic acid residue in a PDB file. <div><div>0noGroup</div><div>1-5ALA, ARG, ASN, ASP, CYS</div><div>6-10GLN, GLU, GLY, HIS, ILE</div><div>11-15LEU, LYS, MET, PHE, PRO</div><div>16-20SER, THR, TRP, TYR, VAL</div><div>21-23ASX, GLX, UNK</div><div>24-29A, +A, G, +G, I, +I</div><div>30-35C, +C, T, +T, U, +U</div></div> <div>Additional unique numbers are assigned arbitrarily by Jmol and cannot be used reproducibly.</div>
property	select xxx=y	label %[xxx]	label %x	print {*}.xxx = y	{*}.xxx = y	description
groupindex	yes	yes	G	yes		overall group index (Jmol 11.6)
identify	yes	yes	U	yes		for a PDB/mmCIF file, same as [%[group]]%r: [%[chain]] %[%[altloc]]/%[model] %[%[atomno]]. For non-PDB data, same as [%[atomName]]/%[model] %[%[atomno]]
insertion	yes	yes	E	yes		protein residue insertion code
ionic	yes	yes	I	yes	yes	radius used for bonding (ionic radius when a formal charge is defined); synonymous with <b>ionicRadius</b> and settable starting in Jmol 12.0
label	yes	yes		yes	yes	current atom label (same as <b>format</b> starting with Jmol 11.8)
model	yes	yes	M	yes		model number
modelindex	yes	yes		yes		a unique number for each model, starting with 0 and spanning all models in all files (Jmol 11.8)
molecule	yes	yes	N	yes		molecule number
occupancy	yes	yes	q/Q	yes	yes	CIF file site occupancy. In SELECT command comparisons ("select occupancy < 90"), an integer n implies measurement on a 0-100 scale; also, in the context %[occupancy] or %q for a label, the reported number is a percentage. In all other cases, such as when %Q is used in a label or when a decimal number is used in a comparison, the scale is 0.0 - 1.0.
partialCharge	yes	yes	P	yes	yes	partial charge
phi	yes	yes	f	yes		protein group PHI angle for atom's residue (Jmol 11.4)
polymer	yes	yes		yes	yes	sequential polymer number in a model, starting with 1. (Jmol 12.0)
polymerLength	yes	yes	L	yes		polymer length
property_xx	yes	yes		yes	yes	a property created using the DATA command
psi	yes	yes	p	yes		protein group PSI angle for the atom's residue (Jmol 11.4)
radius	yes	yes	I	yes	yes	currently displayed radius -- In SELECT command comparisons ("select radius=n"), integer n implies Rasmol units 1/250 Angstroms; in all other cases or when a decimal number is used, the units are Angstroms.

property	select xxx=y	label %{xxx}	label %x	print {*}.xxx = y	description
resno	yes	yes	R	yes	PDB residue number, not including insertion code
selected	yes			yes	1.0 if atom is selected; 0.0 if not [Jmol 12.0]
sequence	yes	yes		yes	PDB one-character sequence code, as a string of characters, with "?" indicated where single-character codes are not available
site	yes	yes	S	yes	crystallographic site number
spacefill	yes	yes		yes	currently displayed radius
straightness	yes	yes	T	yes	quaternion-derived straightness (second derivative of the quaternion describing the orientation of the residue. This quantity will have different values depending upon the setting of <b>quaternionFrame</b> as "A" (alpha-carbon/phosphorus atom only), "C" (alpha-carbon/pyrimidine or purine base based), "P" (carbonyl-carbon peptide plane/phosphorus tetrahedron based), or "N" (amide-nitrogen based). The default is alpha-carbon based, which corresponds closely to the following combination of Ramachandran angles involving three consecutive residues i-1, i, and i+1: -psi <sub>i-1</sub> - phi <sub>i</sub> + psi <sub>i</sub> + phi <sub>i+1</sub> .
strucno	yes	yes		yes	a unique number for each helix, sheet, or turn in a model, starting with 1.
structure	yes	yes		yes	The value of this parameter depends upon the context. Used with <b>select structure=x</b> , x can be either the quoted keyword "none", "turn", "sheet", "helix", "dna", or "rna" or a respective number 0-5. In the context <b>{*}.structure</b> , the return value is a number; in the context <b>label %[structure]</b> , the return is one of the six keywords.
surfacedistance	yes	yes	u	yes	A value related to the distance of an atom to a nominal molecular surface. 0 indicates at the surface. Positive numbers are minimum distances in Angstroms from the given atom to the surface.
symop	yes			yes	symmetry operation code that generated this atom by Jmol; an integer starting with 1. This operator is only present if the file contains space group information and the file was loaded using the {i, j, k} option so as to generate symmetry-based atoms. To select only the original atoms prior to application of symmetry, you can either use "SYMOP=n", where n is the symmetry operator corresponding to "x,y,z", or you can specify instead simply "NOT symmetry" the way you might specify "NOT hydrogen". Note that atoms in special positions will have multiple operator matches. These atoms can be selected using the keyword SPECIALPOSITION. The special form <b>select SYMOP=nijk</b> selects a specific translation of atoms from the given crystallographic symmetry operation. Comparators <, <=, >, >=, and != can be used and only apply to the ijk part of the designation. The ijk are relative, not absolute. Thus, <b>symop=2555</b> selects for atoms that have been transformed by symop=2 but not subjected to any further translation. <b>select symop=1555</b> is identical to <b>select not symmetry</b> . All other ijk are relative to these selections for 555. If the model was loaded using <b>load "filename.cif" {444 666 1}</b> , where the 1 indicates that all symmetry-generated atoms are to be packed within cell 555 and then translated to fill the other 26 specified cells, then <b>select symop=3555</b> is nearly the same as <b>select symop=3</b> and <b>cell=555</b> . (The difference being that cell=555 selects for all atoms that are on any edge of the cell, while symop=3555 does not.) However, the situation is different if instead the model was loaded using <b>load "filename.cif" {444 666 0}</b> , where the 0 indicates that symmetry-generated atoms are to be placed exactly where their symmetry operator would put them (x,-y,z being different then from x, 1-y, z). In that case, <b>select symop=3555</b> is for all atoms that have been generated using symmetry operation 3 but have not had any additional translations applied to the x,y,z expression found in the CIF file. If, for example, symmetry operation 3 is -x,-y,-z, then <b>load "filename.cif" {444 666 0}</b> will place an atom originally at {1/2, 1/2, 1/2} at positions {-1/2, -1/2, -1/2} (symop=3555) and {-3/2, -3/2, -3/2} (symop=3444) and 24 other sites.
property	select xxx=y	label %{xxx}	label %x	print {*}.xxx = y	description
symmetry		yes	o	yes	list of crystallographic symmetry operators generating this atom
temperature	yes	yes	b/t	yes	temperature factor (B-factor)
unitXyz		yes		yes	unit cell XYZ coordinates

uX	yes	yes		yes	unit cell X coordinate normalized to [0,1)
uY	yes	yes		yes	unit cell Y coordinate normalized to [0,1)
uZ	yes	yes		yes	unit cell Z coordinate normalized to [0,1)
valence	yes	yes		yes	the valence of an atom (sum of bonds, where double bond counts as 2 and triple bond counts as 3)
vanderwaals	yes	yes	V	yes	van der Waals radius
vibXyz		yes	v	yes	vibration vector, or individual components as %vx %vy %vz
vibX	yes	yes		yes	vibration vector X coordinate
vibY	yes	yes		yes	vibration vector Y coordinate
vibZ	yes	yes		yes	vibration vector Z coordinate
volume	yes	yes		yes	approximate van der Waals volume for this atom. Note, {*} .volume gives an average; use {*} .volume.sum to get total volume. [Jmol 12.0]
X	yes	yes	x	yes	Cartesian X coordinate [Jmol 12.0]
Y	yes	yes	y	yes	Cartesian Y coordinate [Jmol 12.0]
Z	yes	yes	z	yes	Cartesian Z coordinate [Jmol 12.0]
xyz		yes		yes	Cartesian XYZ coordinates
			g		group index in chain
			r		PDB residue number, including insertion code
			W		PDB residue designator with x, y, z included: [%n]%r %x %y %z (Jmol 11.4)

See also:

[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[functions\]](#) [case default echo for if message reset set switch while](#)

[↑top](#) [🔍search](#) [📖index](#)

### [comment (#)]

Comments in Jmol are preceded by a number sign, '#'. In Jmol 11, the pattern /\* comment text \*/ may also be used, and in Jmol 11.2, /\*\* ... \*/ can be used as a "super-comment" to comment out whole blocks of script that might contain /\* ... \*/ as well. Starting with Jmol 11.6, // at the very beginning of a line (before any characters, including spaces) as an alternative indication of a full-line comment. In Jmol 11.4, if the character string **\*\*\*Jmol Embedded Script\*\*\*** is found within a comment or super-comment block, then ONLY the text following that string and carrying through the rest of the comment or super-comment is processed, and all other text in the file is ignored.

#	Anything following '#' up until the end of a statement is ignored by Jmol with the following three exceptions. (A statement is terminated by a semicolon ";" or a newline.)
#jx	Commands prefixed with #jx <b>will be executed by Jmol</b>
#jc	If the string '#jc' appears <b>anywhere</b> within a statement, then that <b>entire statement</b> will be assumed to be a comment and will be completely ignored by the Jmol interpreter.
state definitions	The <b>write state</b> command produces a script that uses standard-looking Jmol comments appended to some commands in order to supply necessary information for defining the exact state that was present when the command was issued. Generally this involves what atoms are to be selected or ignored, and what files and models are involved. The formatting of these extended script "comments" is very precise; these comments should not be manipulated.

Examples: [in new window using kaolin.mol](#)

# color by polarity  
color background [xffa0a0] # pink  
#jx set perspectiveDepth on; #executed only in Jmol, not Chime or Rasmol  
zoom 125 #jc; # zoom to 125% in Chime and Rasmol but not in Jmol

Chime Note:

Similar comment controls exist in Chime. Commands prefixed with `#!` will be executed in Chime but not in RasMol. Commands containing `##` will be ignored by Chime, but the portion preceding the `##` will be executed in RasMol. Thus we have:

<code>#</code>	not read by Jmol, Chime, or Rasmol
<code>#jx</code> [commands here]	Jmol execution only
<code>#!</code> [commands here]	Chime execution only
[commands here] <code>##</code> <code>#jc</code>	Rasmol execution only
[commands here] <code>#jc</code>	Chime and Rasmol only
[commands here] <code>##</code>	Jmol and Rasmol only

[↑top](#) [🔍search](#) [📖index](#)

### [export]

The Jmol application (not the applet) allows export of the currently rendered scene as files that can be read by [Maya](#), [POV-Ray](#), and VRML readers. See the [write](#) command for details.

[↑top](#) [🔍search](#) [📖index](#)

### [fractional coordinates]

Several Jmol commands, namely [center](#), [centerAt](#), [dipole](#), [draw](#), [isosurface](#), [moveTo](#), [rotate](#), [spin](#), [translateSelected](#), and [unitcell](#), accept coordinates in place of atom expressions. These coordinates are introduced using braces: {x, y, z} or {x y z}. (The commas are optional.) However, when the file data are crystallographic, and the coordinates have been derived by transformation of unit cell coordinates into cartesian coordinates, one can use the unit cell fractional coordinate system instead. The designation of a coordinate as fractional is simplicity itself: just include somewhere in one of the three coordinate values a fraction symbol, `"/`. Thus, {1/2, 0, 0} is a fractional coordinate, and it will be automatically transformed into the correct cartesian point. This allows formation of commands such as **set unitCell {1/2, 1/2, 1/2}** to move the unit cell to a new crystallographic origin (for display purposes only). Since `n/1` is `n`, one can use decimals as well, writing {0.5/1, 0, 0} instead of {1/2, 0, 0}. And since `"/1` is not particularly informative, the `"/1` can be left off to give {0.5/, 0, 0} or {0.5, 0, 0/} as sufficient indication of fractional coordinates. Note that when the [unit cell](#) is shifted, an atom's fractional coordinates also shift with it. Starting with Jmol 12.0, you can set the meaning of {1/2 1/2 1/2} to be relative to the current [unitcell](#) setting, **set fractionalRelative TRUE** or to the default absolute definition, using **set fractionalRelative FALSE**.

[↑top](#) [🔍search](#) [📖index](#)

### [functions]

`x = f(y)` functions  
coordinate transforming .xxx functions  
general .xxx functions  
[array].xxx or .propertyName.xxx modifiers  
{atomExpression}.propertyName  
.xxx(y) functions  
item-selector [i][j]  
user-defined functions

Jmol math allows for two types of functions. The first simply operate on their given parameters. For example, `x = load("myfile.dat")` loads the variable `x` with the contents of the file "myfile.dat". The second operate on the elements of a variable individually in some way and are referred to here as item selector functions. In the listings below, these functions all begin with a period character. For example, `x = {carbon}.bonds` operates so as to deliver the bond set associated with the carbon atoms; `x = "this is a test".replace("s","S")` operates on the individual characters of the string "this is a test"; `x = {oxygen}.label("%U")` assigns `x` the list of labels for the oxygen atoms in the model. Some functions can be used in both contexts. For example, `x = distance({oxygen}, {carbon})` delivers the distance from the CENTER of the oxygens to the CENTER of the carbons. `x = {oxygen}.distance(carbon)` delivers the AVERAGE distance of an oxygen atom to the CENTER of the carbons. These are subtly different. Some functions require parentheses; some do not. Basically, if a function CAN have parameters, for example, `.join()`, `.split()`, or `.label()`, then it MUST have at least empty `()`; if a function CANNOT have parameters, for example, `.atoms`, `.bonds`, or `.ident`, then it NEVER uses parentheses.

**x = f(y) functions**   [back](#)

Several of these functions are described more fully under the heading [atom expressions](#), as they can also be used in commands such as [select](#) and [display](#).

<code>x = acos(y)</code>	the arccosine of y, in degrees in the range 0 to 180 (Jmol 11.8).												
<code>x = angle(a,b,c)</code>	the a-b-c angle, where a, b, and c can be points or atom sets.												
<code>x = angle(a,b,c,d)</code>	the dihedral angle a-b-c-d is measured.												
<code>x = connected(...)</code>	See the discussion of connected() at <a href="#">atom expressions</a> .												
<code>x = cos(y)</code>	the cosine of y, where y is in degrees (Jmol 11.6)												
<code>x = cross(a,b)</code>	the cross product of two vectors of the form {x,y,z} (Jmol 11.6)												
<code>x = data({atomset},"type")</code>	creates model file data of the type MOL, PDB, or XYZ for the selected atom set.												
<code>x = data("dataset_name")</code>	places the text of the data set created using the <a href="#">data</a> command into variable x.												
<code>x = data(stringData,fieldOrColumn,columnCount,firstLine)</code>	separates stringData into lines, then reads lines starting from firstLine (1 being the first line). Data are read from free-format field or column fieldOrColumn. If the data are free-format, then set columnCount = 0, otherwise columnCount indicates the number of columns to read for each data point. The data() function returns a newline-separated list, which can be read directly into atomic properties, for example, using { <code>*</code> }. <b>partialCharge</b> = <b>data(load("mydata.mol2"),9,0,7)</b> .												
<code>x = distance(a,b)</code>	The distance from the geometric center of a to the geometric center of b, where a and b are atom expressions or coordinates.												
<code>x = file("filename")</code>	the full path to the indicated file name. Note that <b>file("??")</b> displays a file dialog, which allows the user to navigate to a different directory, and <b>file("")</b> returns the full path to the current default directory (Jmol 11.8)												
<code>x = format("sprintf format", a, b, c, ...)</code>	This function creates a string using a format similar to that used in C++ to format a set of variables into a string with special codes that start with <code>%</code> . While not an exact implementation of this format, there are strong similarities. Here a, b, and c are variable names, and "sprintf format" is the format string containing a <code>%n.mX</code> code for each variable. As for <b>labels</b> , <b>n</b> and <b>m</b> indicate column formatting and precisions. Both <b>n</b> and <b>m</b> are optional. Here, <b>X</b> indicates the variable type (Jmol 11.8), whereas in, for example, { <code>*</code> }.label("%a"), <b>X</b> represents an atom property, and the type of formatting is determined automatically from that. The options for <b>X</b> include: <table><tr><td><b>i</b> or <b>d</b></td><td>integer. Either i or d can be used synonymously.</td></tr><tr><td><b>f</b></td><td>float/decimal. A negative value for m indicates to use scientific notation with a total of -m digits. The default is full width, full precision.</td></tr><tr><td><b>e</b></td><td>exponential (scientific notation). "%8.3e" is equivalent to "%8.-3f".</td></tr><tr><td><b>p</b></td><td>point. Each of the three coordinates is formatted according to the specified width and precision. The default is %6.2p.</td></tr><tr><td><b>q</b></td><td>quaternion/plane/axisangle. Each of the four elements of the vector {x y z w} are formatted according to the specified width and precision. The default is %6.2q</td></tr><tr><td><b>s</b></td><td>string. Values for precision, m, determine maximum number of characters starting from the left (m &gt; 0) or right (m &lt; 0). So, for example, <b>print format("%0.-3s","testing")</b> prints "ing".</td></tr></table> For example: <b>calculate straightness;print format("average straightness = %4.2f", {<code>*</code>}.straightness)</b>	<b>i</b> or <b>d</b>	integer. Either i or d can be used synonymously.	<b>f</b>	float/decimal. A negative value for m indicates to use scientific notation with a total of -m digits. The default is full width, full precision.	<b>e</b>	exponential (scientific notation). "%8.3e" is equivalent to "%8.-3f".	<b>p</b>	point. Each of the three coordinates is formatted according to the specified width and precision. The default is %6.2p.	<b>q</b>	quaternion/plane/axisangle. Each of the four elements of the vector {x y z w} are formatted according to the specified width and precision. The default is %6.2q	<b>s</b>	string. Values for precision, m, determine maximum number of characters starting from the left (m > 0) or right (m < 0). So, for example, <b>print format("%0.-3s","testing")</b> prints "ing".
<b>i</b> or <b>d</b>	integer. Either i or d can be used synonymously.												
<b>f</b>	float/decimal. A negative value for m indicates to use scientific notation with a total of -m digits. The default is full width, full precision.												
<b>e</b>	exponential (scientific notation). "%8.3e" is equivalent to "%8.-3f".												
<b>p</b>	point. Each of the three coordinates is formatted according to the specified width and precision. The default is %6.2p.												
<b>q</b>	quaternion/plane/axisangle. Each of the four elements of the vector {x y z w} are formatted according to the specified width and precision. The default is %6.2q												
<b>s</b>	string. Values for precision, m, determine maximum number of characters starting from the left (m > 0) or right (m < 0). So, for example, <b>print format("%0.-3s","testing")</b> prints "ing".												
<code>x = getProperty("type",parameters...)</code>	The <a href="#">property</a> of the given type is returned as a Jmol math list. <b>print getProperty()</b> by itself giving the list of available types. Each property type has its own intrinsic structure, but in general the parameters may include an initial atom set specification followed by one or more key values and, in the case of arrays, an item selector (introduced in Jmol 11.4). For example,  <b>print getProperty("bboxInfo","center")</b> <b>x = getProperty("atomInfo",{<code>*</code>},2,"atomno=3")</b> <b>x = getProperty("bondInfo",{<code>*</code>},2,"atom1","sym")</b>												
<code>x = hkl(a,b,c)</code>	generates the plane associated with a given set of Miller plane indices. (Jmol 12.0)												
<code>x = javascript("...")</code>	returns the result of evaluating the specified JavaScript. Applet only; disallowed if on the web page _jmol.noEval = true.												

<b>x = label(...)</b>	See <b>x = format(...)</b> , above.
<b>x = load("filename")</b>	Load the data from the specified file into variable x.
<b>x = load("filename", nBytesMax)</b>	Load the data from the specified file into variable x, but no more than the specified number of bytes. (Jmol 11.8. Since this function returns "java.io.FileNotFoundException: ..." when working from a local drive, the function <b>x = load("filename", 0)</b> can be used to test for the existence of a file. If the function returns the empty string "", then the file exists; if it returns an error message, then the file does not exist. So, for example, the following code loads a PDB file from the RCSB only if not found on the local drive:  <b>if (load(pdbid + ".pdb", 0) == "") {load @ {pdbid + ".pdb"} } else {load @ {"=" + pdbid} }</b>
<b>x = measure(...)</b>	The <b>measure</b> function requires from two to four atom expressions and/or points in space and returns associated measurements in a string, one measurement per line. Additional optional parameters include minimum and maximum ranges, the designations "connected" or "notConnected", the units "nm", "nanometers", "pm", "picometers", "angstroms", "ang", or "au", and a format string in quotes similar to that used by the <b>measure</b> command. (Jmol 12.0)
<b>x = now()</b>	returns the time in milliseconds since some old date; can be used for timing scripts. <b>now(x)</b> returns the number of milliseconds since time x; <b>x = now();.....;print now(x);</b> (Jmol 12.0)
<b>x = plane(pta,ptb,ptc,ptd)</b>	creates an {x y z w} plane from the first three points, and assigns the signs of x, y, z, and w to correspond to a positive distance to ptd as measured with the distance() function. Parameters may be mathematical expressions. (starting with Jmol 11.4)
<b>x = plane(a, b, c, d)</b>	creates the four-vector {a b c d}, which represents a plane satisfying the equation <b>ax + by + cz + d = 0</b>
<b>x = plane("a b c d")</b>	creates an plane satisfying the equation <b>ax + by + cz + d = 0</b> from a string equivalent. As for all Jmol math expressions, parameters may be mathematical expressions. (Jmol 11.4)
<b>x = plane(pta,ptb,ptc)</b>	creates an {x y z w} plane through the three given points, which may themselves be mathematical expressions that evaluate to {x y z} points or atom expressions. (Jmol 11.4)
<b>x = point(a,b,c)</b>	Creates an {x y z} point. Parameters may be mathematical expressions. (starting with Jmol 11.4)
<b>x = point("{x,y,z}")</b>	creates an {x y z} point from the string equivalent. Parameters may be mathematical expressions. (starting with Jmol 11.4)
<b>x = prompt(message)</b>	displays a pop-up message box and waits for the user to press OK
<b>x = prompt(message,defaultInput)</b>	displays an input dialog allowing the user to enter some text and press OK or to cancel. If canceled, returns "null".
<b>x = prompt(message,buttonText,TRUE)</b>	displays a message box with buttons and returns the label of the button that was pressed. The parameter is a list of button labels separated by  , for example: "Yes/No" or "OK/cancel" or "Spacefill/Wireframe/Ball&Stick".
<b>x = prompt(message,buttonArray)</b>	displays a message box with buttons based on the values in the array parameter and returns an integer indicating which button was pressed (starting with 1).
<b>x = quaternion({x y z},theta)</b>	the quaternion {x y z w} associated with a rotation of <b>theta</b> degrees (counter-clockwise) around axis {x y z}. {x y z} need not be a unit vector -- Jmol will normalize it automatically. (Jmol 11.6)
<b>x = quaternion("{x y z w}")</b>	the unit quaternion {x y z w} produced by normalizing the specified quaternion. That is, where <b>theta</b> is the rotation angle, and <b>q0 = w = cos(theta/2)</b> , <b>{x y z} = sin(theta/2) * unitNormal</b> . (Jmol 11.6)
<b>x = quaternion(q0, q1, q2, q3)</b>	the unit quaternion {x y z w} produced by normalizing the specified quaternion. Note that q0 is first in the list of parameters, even though Jmol will store the quaternion in the form {q1/f, q2/f, q3/f, q0/f} where <b>f</b> is <b>sqr(q0*q0 + q1*q1 + q2*q2 + q3*q3)</b> . (Jmol 11.6)
<b>x = quaternion({m00 m10 m20}, {m01 m11 m21})</b>	the unit quaternion corresponding to the rotation matrix having the first two column vectors indicated. (Jmol 11.8)
<b>x = quaternion({center},{x-axis point}, {xy-plane point})</b>	the unit quaternion associated with a frame that has a center at the first parameter position, an x axis in the direction of the second parameter position, and a y axis in the plane of the three parameter positions. These parameters may be atom expressions. For example, <b>x = quaternion({215.CA},{215.C},{215.N})</b> creates the "standard" quaternion for residue 215. The z axis will be generated using the x and y axes and the right-hand rule. (Jmol 11.8)
<b>x = quaternion({atom})</b>	the quaternion associated with an atom (or the first atom in the atom set) based on the setting of <b>set quaternionFrame</b> .

<b>x = quaternion({atomset}, nMax)</b>	an array of quaternions, one per residue, up to nMax long (or all if nMax is <=0). Quaternions are created based on the setting of <b>set quaternionFrame</b> .
<b>x = quaternion({atom1}, {atom2})</b>	the quaternion difference of the two residues containing the specified atoms (or the first atom in each set, if applicable), created based on the setting of <b>set quaternionFrame</b> . An optional last parameter "relative" utilizes quaternion left-division. The quaternion relative difference represents the necessary rotation to get from q1 to q2 within the reference frame of q2 rather than the standard reference frame (Jmol 12.0)
<b>x = quaternion({atomset1}, {atomset2}, nMax)</b>	an array up to nMax long (or all if nMax is <=0) of quaternion differences, one per residue pair in the two sets, created based on the setting of <b>set quaternionFrame</b> . An optional last parameter "relative" utilizes quaternion left-division (Jmol 12.0)
<b>x = quaternion(quaternionArray1, quaternionArray2)</b>	an array of quaternion differences of the two array elements, taken a pair at a time. An optional last parameter "relative" utilizes quaternion left-division. (Jmol 12.0)
<b>x = script("...")</b>	returns the output from the specified script command; particularly informative script commands include <a href="#">getProperty</a> and <a href="#">show</a> .
<b>x = script("...", appletName)</b>	returns the output from the specified script command run in one or more applets. For example, <b>print script("show orientation moveto", 2)</b> will print the orientation moveto command for an applet with name "2" or, if that does not exist, "JmolApplet2". See <a href="#">script</a> for details.
<b>x = select(x;{a};b)</b>	selects atoms from the atom expression {a} based on the boolean expression b. (Jmol 11.6) Note the use of semicolons, not commas, to separate the three components of this function. The variable x is local to the function, and when it appears in the boolean expression in the form <b>x.property</b> represents a single atom of the atom expression. For example, <b>x = select(a;{*};a.distance(0 0 0) &gt; 3 and a.clemmo &gt; 18)</b> . select() functions can be nested -- just use two different variable names: <b>x = select(x;{*.ca};x.phi &lt; select(y;{*.ca}; y.resno = x.resno + 1).phi)</b> . The select() function provides a powerful selection mechanism that can utilize any Jmol math expression involving properties of an atom. (In contrast, <b>select</b> command comparisons are limited to =, <, >, <=, >=, and values are rounded to the nearest 1/100th).
<b>x = sin(y)</b>	the sine of y, where y is in degrees (Jmol 11.6)
<b>x = sqrt(y)</b>	the square root of y (Jmol 11.6)
<b>x = substructure("smiles")</b>	find atoms matching the given smiles string, which may include bond types such as = or - between atoms. Note that unspecified bond evaluates to single, not "any". (This behavior may be revised in future releases.) <a href="#">atom expressions</a>
<b>x = within(...)</b>	returns a matching atom set for a wide variety of atoms that are in some way "within" another atom set or plane or "within some distance" of any one of a set of atoms or a plane.
<b>x = within("SMILES", smilesString,...)</b>	returns any array of atom sets that match the given smiles string. Additional optional parameters include {searchSet}, {requiredAtoms}, and {notAtoms}. If provided, only atoms in {searchSet} will be checked, matches must include all {requiredAtoms} and must not include any of {notAtoms}. (Jmol 12.0)
<b>x = write(...)</b>	the output of the <b>write</b> command is loaded into variable x. The parameters are those of the write command. For example, <b>x = write("PDB")</b> or <b>x = write(quaternion, "r", "difference2")</b> . Note that <b>x = write("image")</b> is not supported.

#### coordinate transforming .xxx functions [back](#)

These functions operate on coordinates, either the geometric center of a set of atoms expressed as {atom expression} or a coordinate point expressed as {x y z}. The unit cell system used is the currently defined **unitCell** for the current model. If more than one model is visible, all coordinates are considered Cartesian coordinates, and these functions are not distinguishable.

<b>x = pt.xyz</b>	the Cartesian coordinates for the point. For example, if <b>pt = {1/2 3/2 1}</b> in an orthonormal unit cell with a = 28.0, b = 5.04, c = 6.04, then pt.xyz would equal {14.0 7.56 6.04}.
<b>x = pt.fxyz</b>	the fractional coordinates of a point. In the same case, <b>pt.fxyz</b> would be {0.5 1.5 1.0}. (Jmol 11.8)
<b>pt.fx, pt.fy, pt.fz</b>	The fractional x, y, and z coordinates, respectively.
<b>x = pt.uxyz</b>	the unit-cell normalized point in the range [0,1). In this case, <b>pt.uxyz</b> would be {0.5 0.5 0}. (Jmol 11.8)
<b>pt.ux, pt.uy, pt.uz</b>	The unit cell x, y, and z coordinates, respectively.
<b>pt.x, pt.y, pt.z</b>	The x, y, or z component of the point, regardless of the unit system.

Note that the fractional notation {1 2/2 1} is immediately converted to Cartesian coordinates, so {1/2 3/2 1} = {14.0 7.56 6.04} in this case, and {1/2 3/2 1}.y = 7.56, whereas {1/2 3/2 1}.fy = 1.5, and {1/2 3/2 1}.uy = 0.5. Note that if vibration vectors are set using, for example, **{atomno=3}.vxyz = {1/2 1/2 1/2}**, then the **frame** command should be sent so as to update the popup menu so that the **vibration** menu item is enabled (Jmol 11.8).

#### general .xxx functions [back](#)

These modifiers can be used with a number of different variable types.

<b>x = y.keys</b>	Returns the set of keys in associative array y. (Jmol 12.0)
<b>x = y.length</b>	In the case of x a set of bonds, the average length of the bonds. In all other cases, the length of the data. For example: <b>x = {carbon}.bonds.length</b> , <b>x = {*}.length</b> . To check the size of a bondset, use .size.
<b>x = y.lines</b>	splits y into a set of lines based on new-line characters, appending a new-line character onto the end if necessary so that there is one new-line character per line.
<b>x = y.size</b>	The nominal "size" of y, which depends upon its data type -- number of characters in a string, number of elements of an array, number of selected atoms or bonds in a bitset. Except for bonds, same as .length. Negative numbers indicate boolean (-1), integer (-2), decimal (-4), point (-8), or plane (-16).
<b>x = y.type</b>	the type of variable; one of "boolean", "integer", "decimal", "string", "point", "point4", "bitset", "array", "matrix3f", "matrix4f" (Jmol 11.4 and later)

#### [array].xxx or .propertyName.xxx modifiers [back](#)

Several modifiers can be added to property functions **{atoms}.y** or (starting with Jmol 11.7) arrays, such as **[1.2, 1.4, 1.6]**.

<b>.all</b>	If a property, such as <b>{selected}.vanderwaals</b> , then appending <b>.all</b> creates a list of those measures. This list can be used to transfer one property to another, as in: <b>{*}.partialCharge = {*}.temperature.all</b> , which would allow temperature data to be used for partial charges in an <b>isosurface molecular map MEP</b> command, for instance. Note that the "list" is really an array of string values.
<b>.average</b>	the average value (the default modifier).
<b>.max</b>	the maximum value, for example: <b>{*}.temperature.max</b>
<b>.min</b>	the minimum value, for example: <b>{*}.partialCharge.min</b>
<b>.stddev</b>	the standard deviation, for example: <b>print {helix}.straightness.stddev</b>
<b>.sum</b>	the sum of the values (Jmol 12.0; before this version, use .add())
<b>.sum2</b>	sum of squares

#### {atomExpression}.propertyName [back](#)

These functions operate on the individual elements of some group or listing. In addition to these are all of the atom properties described under the heading [atom expressions](#), for example, **x = {oxgyen}.temperature**, in which case they give the average value.

<b>x = y.atoms</b>	The atoms associated with a set of bonds.
<b>x = y.bonds</b>	The bonds associated with the specified atoms, using the current setting of <b>bondModeOr</b> (true, bonds having one OR the the other atom within this set; or false, bonds having BOTH atoms within the set)
<b>x = y.boundbox</b>	A Jmol math list containing the boundbox center, vector to corner, and two opposite corners associated with this atom set (Jmol 11.4)
<b>x = y.color</b>	The average color of the atoms in the set y expressed as a {r,g,b} coordinate in color space. If y is already a point, then .color converts this to a string of the form [xRRGGBB]. If the propertyColorScheme has been set, and it has been used, for example with <b>color atoms property partialcharge "rwrb" range -1 1</b> or <b>color "roygb" range 0 10</b> , then y can be a number, as in <b>x = (3).color</b> , in which case the color associated with that value is returned. Using this mechanism, a key can be generated within the Jmol applet using echo text appropriately positioned on the screen. Leaving the echo blank but coloring the background of the echo as well as the echo itself produces a horizontal bar of the desired color. (Jmol 11.4)
<b>x = y.ident</b>	a list of the standard identity labels for the elements of y, either atoms or bonds.
<b>x = y.length</b>	In the case of x a set of bonds, the average length of the bonds. In all other cases, the length of the data. For example: <b>x = {carbon}.bonds.length</b> , <b>x = {*}.length</b> . To check the size of a bondset, use .size.
<b>x = y.size</b>	The number of selected atoms or bonds in a bitset.

#### .xxx(y) functions [back](#)

These functions operate on the individual elements of some group or listing, returning a modified listing.

<b>x = data1.add(data2)</b>	Specifically for use with data set lists, adds each element of data1 to its corresponding element in data2 and returns the result. If data2 is a simple number, adds that number to each element of data1.								
<b>x = data1.add("t",data2)</b>	Specifically for use with data set lists, creates a new column separated from the previous with a tab. ("t" may be replaced with whatever separation one desires. (Jmol 11.8)								
<b>x = y.distance({atoms})</b>	the average distance from elements of y to the CENTER of {atoms}								
<b>x = y.find("s")</b>	finds the first location of "s" in y or, in the case of y being a set of lines, selects only the lines of y containing "s".								
<b>x = y.find("pattern","flags")</b>	(Jmol 11.8) Searches string or list x for a <a href="#">Java regular expression pattern</a> . The second parameter is a set of flags. This parameter must be included, even if it is the blank string "" so as to distinguish this command from the standard .find() command. Flags include: <table><tr><td><b></b></td><td>(no flags) Returns the position in the string containing the pattern, starting with 1, or 0 if the pattern is not found. For lists, returns a sublist containing the elements that match.</td></tr><tr><td><b>i</b></td><td>(case-insensitive) Match upper or lower case letters.</td></tr><tr><td><b>v</b></td><td>(reverse match) With a string, v returns "false" if the string contains the match or "true" if it does not; with lists, v returns all elements of the list that do not contain the match.</td></tr><tr><td><b>m</b></td><td>(return match) The m option allows returning only the portion of the string that matches (or, with <b>vm</b>, only the portion of the string that does NOT match). With lists, both <b>m</b> and <b>vm</b> return only elements that contain the match, but, as for strings, each element is returned as just the matching phrase or the reverse.</td></tr></table>	<b></b>	(no flags) Returns the position in the string containing the pattern, starting with 1, or 0 if the pattern is not found. For lists, returns a sublist containing the elements that match.	<b>i</b>	(case-insensitive) Match upper or lower case letters.	<b>v</b>	(reverse match) With a string, v returns "false" if the string contains the match or "true" if it does not; with lists, v returns all elements of the list that do not contain the match.	<b>m</b>	(return match) The m option allows returning only the portion of the string that matches (or, with <b>vm</b> , only the portion of the string that does NOT match). With lists, both <b>m</b> and <b>vm</b> return only elements that contain the match, but, as for strings, each element is returned as just the matching phrase or the reverse.
<b></b>	(no flags) Returns the position in the string containing the pattern, starting with 1, or 0 if the pattern is not found. For lists, returns a sublist containing the elements that match.								
<b>i</b>	(case-insensitive) Match upper or lower case letters.								
<b>v</b>	(reverse match) With a string, v returns "false" if the string contains the match or "true" if it does not; with lists, v returns all elements of the list that do not contain the match.								
<b>m</b>	(return match) The m option allows returning only the portion of the string that matches (or, with <b>vm</b> , only the portion of the string that does NOT match). With lists, both <b>m</b> and <b>vm</b> return only elements that contain the match, but, as for strings, each element is returned as just the matching phrase or the reverse.								

Note that special characters such as \S and \d must be escaped with two back-slashes, and if they are introduced via JavaScript, they will need double escaping (four back-slashes). Examples include:

<b>"this test is a Test".find("Test",")"</b>	<b>16</b>
<b>print "this test is a Test".find("Test","i")</b>	<b>6</b>
<b>print "this test is a Test".find("Test","m")</b>	<b>Test</b>
<b>print "this test is a Test".find(" a test","v")</b>	<b>true</b> (because it was <b>not</b> found)
<b>print "this test is a Test".find(" Test","ivm")</b>	<b>this is a Test</b>
<b>print "this test is a Test".find("\stest","m")</b>	<b>test</b>
<b>print "this test is a Test".find("\stest","vm")</b>	<b>this is a Test</b>
<b>print script("show spacegroup all").split().find("Hall symbol:")</b>	<b>Hall symbol: P 1 primitive Hall symbol: P 1 Hall symbol: -P 1 primitive Hall symbol: P 1 -1 Hall symbol: P 2y primitive Hall symbol: P 2y ...</b>
<b>print script("show spacegroup all").split().find("Hall symbol:").find("primitive","v")</b>	<b>Hall symbol: P 1 Hall symbol: -P 1 Hall symbol: P 2y ...</b>
<b>print script("show spacegroup all").split().find("Hall symbol:").find("primitive","v").find("Hall symbol:","vm")[1][3]</b>	<b>P 1 -P 1 P 2y</b>

<b>x = {atomExpression}.find("SEQUENCE")</b>	(Jmol 12.0) returns the Jmol <b>bioSMILES</b> sequence for the specified atoms. An optional second parameter TRUE adds crosslinking.
<b>x = smilesString.find("SMILES",pattern)</b>	(Jmol 12.0) Searches a SMILES string for at least one occurrence of <b>pattern</b> . For example: "O[C@](F)(Cl)".find("smiles","[C@](O)(F)(Cl)" would return the value 1. A return value of 0 means the pattern was not found; a return of -1 indicates there was a problem parsing one or the other of the SMILES strings. This function allows Jmol to match two SMILES strings without need of "canonicalization."

<b>x = smilesString.find("SMARTS",pattern)</b>	(Jmol 12.0) Searches a SMILES string for at least one occurrence of <b>pattern</b> , where the pattern is in the form of a SMARTS string. For example: "CCCC".find("smarts","CC") would return the value 1. A return value of 0 means the pattern was not found; a return of -1 indicates there was a problem parsing one or the other of the SMILES strings. This function allows Jmol to match two SMILES strings without need of "canonicalization."
<b>x = y.find("SMILES","MF")</b>	returns the canonical molecular formula of the SMILES string associated with the SMILES string or atom set y
<b>x = y.find("SMARTS","MF")</b>	returns the canonical MF for the SMARTS string (thus not including H atoms) of the SMILES string or atom set y.
<b>x = y.join("s")</b>	joins lines of y using the character or string "s"
<b>x = y.label("format")</b>	A list of labels for atoms or bonds using format strings. (see <a href="#">label</a> )
<b>x = data1.mul(data2)</b>	See data1.add()
<b>x = y.replace("s1","s2")</b>	replaces all occurrences of "s1" with "s2" in y. If y is a number, this function first converts the number to a string, then does the replacement.
<b>x = y.split("s")</b>	splits y into lines by replacing all occurrences of "s" with a new-line character and converting the string to a set of lines.
<b>x = data1.sub(data2)</b>	See data1.add()
<b>x = y.substring()</b>	Jmol math does not include a .substring() function. Instead, this is handled using the more general item-selector [i][j] syntax, described below.
<b>x = y.symop(op,atomOrPoint)</b>	Returns the point that is the result of the transformation of <b>atomOrPoint</b> via a crystallographic symmetry operation. The atom set y selects the unit cell and spacegroup to be used. If only one model is present, this can simply be <b>all</b> . Otherwise, it could be any atom or group of atoms from any model, for example <b>{*/1.2}</b> or <b>{atomno=1}</b> . The first parameter, <b>op</b> , is a symmetry operation. This can be the 1-based index of a symmetry operation in a file (use <b>show spacegroup</b> to get this listing) or a specific Jones-Faithful expression in quotes such as "x,1/2-y,z".
<b>x = y.symop(op,"label")</b>	This form of the .symop() function returns a set of <a href="#">draw</a> commands that describe the symmetry operation in terms of rotation axes, inversion centers, planes, and translational vectors. The <b>draw</b> objects will all have IDs starting with whatever is given for the label.
<b>x = y.trim("chars")</b>	trims any one of the characters specified from both ends of the string y, or from every line of y if y is a set of lines.
<b>x = data1.div(data2)</b>	See data1.add(). Division by 0 results in the value "NaN", meaning "not-a-number".

item-selector [i][j] [back](#)

In Jmol 11.2, specific elements of lists and strings can be selected using a square bracket notation similar to that used for [atom expressions](#). Positive numbers select for the specified element. For example, **x = "testing"[3]** selects for the third letter of the string, "s"; Zero selects for the last element: **x = "testing"[0]** selects "g". Negative numbers count from the end of the string backward, so **x = "testing"[-1]** selects "n". Two selectors select an inclusive range of items. **x = "testing"[2][4]** acts similar to a "substring()" method and selects characters 2 through 4 -- "est"; **x = "testing"[-1][0]** selects the last two characters, "ng". Similarly, for arrays, **array("this", "test", 3)[0]** selects the number 3, and **x = array("this", "test", 3); print x[1][2]** prints

this  
test

user-defined functions [back](#)

In addition, Jmol allows for user-defined functions. You can define functions to do simple tasks, include parameters to create subroutines, and add return values to create math functions. With applets, if the function name starts with "static\_", then the function will be defined for ALL applets, not just the one running the script. In general, functions are part of the state and will be saved with a state. However, functions starting with "\_" will not be saved with the state and will not be reported with **show functions**, but will be reported with **show functions " \_"**. (Functions of this sort are part of the state itself.) Examples follow.

function rotateModel { rotate {atomno=1} {atomno=2} 10 }	Without any parameters listed, any commands within the function declaration are processed with a single command word.
--	---

rotateModel	
function rotateModelX(i,j,n) { a[2] = getProperty("orientationInfo.moveTo") var x = 10 rotate {atomno=i} {atomno=j} @ {n + x} a[2] = getProperty("orientationInfo.moveTo") }	Any number of parameters may be passed into a function. The variable names are local to that function. Note that, like JavaScript, if an array is passed, it is passed "by reference," meaning that if its value is changed within the function, then that change is a global change. Variables preceded by "var" as in this example are local to that function.
a = [];rotateModelX(10,11,30, a);print a[1];	
function getDistance(i,j) { var d = ({atomno=i}.distance({atomno=j}))*100)%0 return d }	The <b>return</b> command works as one might expect, returning the value of its expression.
print "the distance is " + getDistance(3,5)	
function getXplusY { return _x.x + _x.y }	Starting with Jmol 11.8, you can define atom selector functions. The local variable <b>_x</b> will represent the selected atom within the function. Such function references may include parameters in parentheses provided qualifiers such as .min, .max, or .all are not used within the function itself.
print {atomno=3}.getXplusY print {*}.getXplusY.min print {*}.getXplusY.max print {*.CA}.getXplusY.all	

Functions must be defined prior to use. Similar to the JavaScript language, a function must be declared prior to its inclusion in a script command. You can redefine a function as many times as you wish. In addition, you can save a function's definition using **myFunctionDef = script("show function myFunction")** where function **myFunction** has been defined. This can be useful if a function definition is to be replaced and later returned to its original value using **script inline @myFunctionDef**.

See also:

[Jmol command syntax](#) [Jmol math](#) [Jmol parameters](#) [atom expressions](#) [atom properties](#) [case default echo for if message reset set switch while](#)

[top](#) [search](#) [index](#)

### [plane expressions]

Several commands in Jmol accept parameters that define planes. The commands **depth**, **draw**, **isosurface**, **mo**, and **slab** all have the PLANE option, and **depth**, **draw isosurface**, and **slab** also have the HKL option for defining planes using Miller indices. In addition, **select**, **restrict**, **display**, and **hide** as well as many other commands accept **atom expressions**, and these may include one or more WITHIN() function. The method of describing planes in Jmol is defined below:

<b>hkl</b>	Miller indices are simply indicated by enclosing them in brackets, usually as fractions: {h k l}. For example,  <b>isosurface hkl {0 1/2 1/2}</b>  ,
<b>plane</b>	A plane is defined in Jmol in one of five ways: (a) listing three points, (b) giving the parametric equation ax + by + cz + d = 0, (c) referencing an already-defined <a href="#">draw</a> or <a href="#">isosurface</a> object that describes a plane, (d) using the strings "xy", "xz", or "yz", or (e) using a simple single-variable equation such as "x=3" or "y=-2". Points may be embedded atom expressions (that is, atom expressions surrounded by parentheses), for which the center is used, Cartesian coordinates expressed as {x,y,z}, or crystallographic fractional coordinates indicated using "/" in at least one coordinate: {0,1/2,1/2}. Commas are optional. Any combination of these three point types may be used. For example:  <b>display within(0,plane,@{plane({atomno=3},{0 0 0},{0 1/2 1/2}}))</b>  The parametric equation ax + by + cz + d = 0 is expressed as {a b c d}.  Planes based on <b>draw</b> and <b>isosurface</b> objects are first defined with an ID indicated, for example:  <b>draw plane1 (atomno=1) (atomno=2) (atomno=3)</b>  After that, the reference <b>\$plane1</b> can be used anywhere a plane expression is required. For instance,  <b>select within(0,plane,\$plane1)</b>

	or
	<b>isosurface PLANE \$plane1</b>
These objects can be defined and then hidden from the user using <b>draw off</b> or <b>isosurface off</b> . For selection purposes the plane is considered of infinite extent even if it only shows up as a small triangle or quadrilateral.	

See also:  
[\[atom expressions\]](#) [isosurface](#) [IcaoCartoon](#) [mo](#) [pmesh](#) [polyhedra](#)

[read/write ZIP files]

Jmol can read and write ZIP file collections. In any file reading operation (for example, in the [isosurface](#), [load](#), or [script](#) command), Jmol can read a specific file within ZIP or JAR file collections. (The two formats are the same.) Simply indicate the path to the file within the collection using the vertical bar character as a separator: **load "myfiles.zip|files|pdb|1crn.pdb"**. A path element may be a subdirectory within the ZIP file or the name of a ZIP file contained within one of those directories: **isosurface "mysurfaces.zip|isousurfaces|vxl.zip|surface1.vxl"**. The full directory contents of a ZIP or JAR file can be read out using **getProperty fileContents "myfile.zip"**, and the contents of specific files within a zip collection can also be read this way as, for example, **getProperty fileContents "myfile.zip|JmolManifest"**. Note that variables can be assigned these contents using the `getProperty()` function: **var dir = getProperty("fileContents", "myfile.zip|JmolManifest")**. The [load](#) command specifically allows reading of multiple files from a ZIP or JAR collection, and will read files in the order specified in a file with the name "JmolManifest" contained in the compressed file collection. Starting with Jmol 12.0, the command [write xxx.jmol](#) or [write ZIPALL xxx.zip](#) will create a ZIP file collection that contains a JmolManifest file along with all of the files necessary to recreate the current state. In addition, [write ZIP xxx.zip](#) will make a collection that contains all local files necessary, but will simply point to any required remote ([http://](#), [ftp://](#)) resource.

[status reporting]

With Jmol 11.0 there is a new mechanism for reporting applet status to a web page. This mechanism uses active polling -- the web pages periodically queries the applet using `jmolGetStatus(strStatus,targetSuffix)` as to the applet's status rather than using a callback mechanism. When ON (the default condition) Jmol can be tested for one or more of the following status types by specifying one or more keywords in the `strStatus` parameter:

keyword	return
atomPicked	returns the identity of the atom picked
fileLoaded	returns the full path of the file loaded along with the simple filename and an integer value (-1) if there was an error.
fileLoadError	returns the phrase "java.io.FileNotFoundException: [filename]" (The system cannot find the file specified)" or other such error if a file error occurs.
frameChanged	returns the number of the frame being displayed and a line of information about the model
measurePending	returns what atoms are being hovered over during the middle of a measurement picking operation
measureCompleted	returns measurement information when the user completes a measurement by clicking.
measurePicked	returns information about a measurement made using <b>set picking DISTANCE or ANGLE or TORSION</b>
scriptStarted	returns a message indicating that a script has started.
scriptEcho	returns the "echo" message delivered by one of the following two events: (a) a report from the script command <a href="#">getProperty</a> or (b) the echo message from the <a href="#">echo</a> command, regardless of whether or not the message is displayed to the user using <b>echo top left</b> or not, using <b>set echo off</b> .
scriptStatus	returns the message "Script completed" if and only if a script is successfully completed as well as a variety of messages such as the number of atoms selected.
scriptError	returns a message starting with "script ERROR" or "script compiler ERROR" if a script-based error occurs.
scriptMessage	returns the message "Jmol executing script ..." when a script starts if there was no compiler error or the compiler error itself, such as "command expected : loadf  line#1", if an error occurs during script syntax checking.

scriptTerminated	returns the message "Jmol script terminated successfully" if there are no errors upon script completion or "Jmol script terminated unsuccessfully:" followed by an error message if an error occurred.
------------------	--

**set statusReporting OFF** turns off the StatusManager system.

Note: Some versions of Firefox/Java have a [known bug](#) associated with Java applet polling. To date, [callbacks](#) have proven more reliable.

[using the clipboard]

Model data that has been clipped from local sources, such as a return from [the RCSB PDB file directory](#), can be loaded into Jmol directly. From the application, simply use Edit<sup>®</sup>Paste; if using the applet, simply right-click on the applet and select Show...Console. Then paste the data into the lower (input) frame and click **Load**.

See also:  
[write](#)

animation or anim

Sets selected animation parameters or turns animation on or off. Note that there are four distinct animation types that can be employed using Jmol: (1) files may contain multiple structures that are "played" sequentially, (2) models may contain vibrational modes that can be animated, (3) certain Jmol script commands (namely [move](#), [moveTo](#), [navigate](#), [restore ORIENTATION](#) and [zoomTo](#)), can create the illusion of motion over a period of time, and (4) Jmol scripts can be run through in a predefined way, involving [loop](#) and [delay](#). The "animation" command only refers to method (1). See also [set backgroundModel](#).

animation ON/OFF {default: ON}

Turns on or off animation. With **animation ON**, the [frame](#) range is also reset to all frames. (An implicit **frame range ALL** command is executed. This functionality is for backward compatibility with Chime.) If this resetting is not desired because the frame range has been set, then **frame PLAY** or **frame PLAYREV** should be used instead of **animation ON**.

animation direction -1

Sets the animation direction to be from last frame to first frame.

animation direction +1

Sets the animation direction to be first frame to last frame.

animation fps [\[frames-per-second\]](#)

Sets the animation frames per second.

animation frame

**animation frame**, **frame**, and **model** are synonymous. See options at the [frame](#) command.

animation mode LOOP

Sets the animation mode to restart the sequence automatically when the last frame has played.

animation mode LOOP [\[time-delay1\]](#) [\[time-delay2\]](#)

Allows for a time delay at the start and end of the loop.

animation mode ONCE

Sets the animation to play once through and then stop (the default mode).

animation mode PALINDROME

Sets the animation to play forward and back endlessly.

animation mode PALINDROME [\[time-delay1\]](#) [\[time-delay2\]](#)

Allows for a time delay at the start and end of the palindrome.

where

**[frames-per-second]** is the animation rate -- (integer)

**[time-delay1]** is the time in seconds to pause on the first frame -- (integer/decimal, >=0)

**[time-delay2]** is the time in seconds to pause on the last frame -- (integer/decimal, >=0)

Examples:

See [animation.htm](#)



See also:

[frame invertSelected model move moveto rotateSelected set \(misc\) spin translate translateSelected zoom zoomto](#)

[↑top](#) [🔍search](#) [📖index](#)

## axes

(v. 11.0 -- new)

Turns on or off displayed axes, and determines their line style and line width (as a decimal number, in Angstroms).

axes ON/OFF {default: ON}

Turns axes on or off

axes (decimal)

Sets the axes diameter in Angstroms.

axes CENTER {x y z}

Sets the axes origin to the specified point, which may be an atom expression such as {*i*} or {*rna*}.

axes DOTTED

Sets the axes style to a thin dotted line.

axes (integer)

Sets the axes diameter in pixels (1 to 19).

axes LABELS "x-label" "y-label" "z-label"

Sets the labels for the positive X, Y, and Z axes. Negative-directed axes are hidden.

axes LABELS "x-label" "y-label" "z-label" "-x-label" "-y-label" "-z-label"

Sets the labels for the positive and negative X, Y, and Z axes. (Negative labels for unit cell axes are ignored.)

axes MOLECULAR

Sets the axes to be based on the molecular coordinate {0 0 0}

axes POSITION [x y] or [x y %]

Sets the axes to be positioned at a specific screen coordinate or fractional position along the horizontal and vertical screen dimensions. Must be preceded by **axes on**.

axes SCALE (decimal)

Sets the axes to align with the a, b, and c axes of the unit cell (default if fractional coordinates)

axes TICKS X|Y|Z {major,minor,subminor} FORMAT [%0.2f, ...] SCALE {scaleX, scaleY, scaleZ} | x.xx

Sets the parameters for ticks along the axes. An optional specific axis (X, Y, or Z) can be indicated. There are three levels of ticks - major, minor, and "subminor." Only the major ticks have labels. Which of these tick levels are displayed and the distance between ticks depends upon the parameter that takes the form of a point. This point may be in fractional form, {1/2 0 0}. The optional keyword FORMAT allows formatting of the labels for the major ticks. These are based on an array of strings given after the FORMAT keyword. If the array is shorter than the number of ticks, the formats in the array are repeated. Following that, the optional keyword SCALE allows setting the scale either for each axis direction independently {scaleX, scaleY, scaleZ} or overall (as a decimal number).

axes UNITCELL

Sets the axes to align with the a, b, and c axes of the unit cell (default if fractional coordinates)

axes WINDOW

Sets the axes to be based on the center of the bounding box (default if not fractional coordinates)

See also:

[boundbox measure set \(visibility\) unitcell](#)

[↑top](#) [🔍search](#) [📖index](#)

## backbone

Shows the backbone of a protein or nucleic acid macromolecule by connecting the alpha carbons. The selection of backbone units to display depends upon the currently selected atoms and the **bondmode** setting.

backbone ON/OFF {default: ON}

Turns the backbone on or off

backbone ONLY

Turns backbone rendering on and all other rendering off.

backbone [[backbone-radius](#)]

Backbone radius can be specified in angstroms using a decimal number (1.0, 2.0, etc.). Starting with Jmol 12.0, a negative number also implies **ONLY**.

where

[**backbone-radius**] is the radius of the backbone -- (decimal, <=4.0)

Examples:

See [structure.htm](#)



See also:

[background cartoon dots ellipsoid geoSurface meshribbon ribbon rocket set \(highlights\) set \(lighting\) set \(navigation\) set \(perspective\) set \(visibility\) spacefill strand trace vector wireframe](#)

[↑top](#) [🔍search](#) [📖index](#)

## background

Sets color of the background or, starting with Jmol 11.6, sets the background image. For color specifications, see [color](#).

background [[RGB-color](#)]

Sets the background color of the applet/application window.

background ECHO [[color-none-CPK](#)]

Sets the background of the most recently defined **echo** text and subsequent user-defined echo text to be the given color.

background IMAGE "filename"

Sets the background of the applet/application window to the specified image file, which can be of format JPG, PNG, or GIF. The image is stretched to fit the size of the window.

background HOVER [[color-none-CPK](#)]

Sets the background color for the pop-up label box that appear when the mouse "hovers" over an atom. "NONE" results in there being no hover backgrounds. Operates globally, not on selected atoms.

background LABELS [[color-none-CPK](#)]

Sets the background color of the atom labels that appear with the "label" command. "NONE" results in there being no label background. Operates globally, not on selected atoms.

where

[**RGB-color**] is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]

[**color-none-CPK**]

is (color name), [r, g, b], [xRRGGBB], NONE

See also:

[backbone cartoon color \(atom object\) color \(bond object\) color \(element\) color \(model object\) color \(other\) color measures dots ellipsoid geoSurface meshribbon ribbon rocket set \(highlights\) set \(lighting\) set \(navigation\) set \(perspective\) set \(visibility\) set userColorScheme show spacefill strand trace vector wireframe](#)

[↑top](#) [🔍search](#) [📖index](#)

## bind

(v. 12.0)

The **bind** and **unbind** commands allow users to customize the effects of mouse actions. When a mouse action is bound to a script, Jmol will replace `_X`, `_Y`, `_DELTA_X`, `_DELTA_Y`, `_TIME`, and `_MODE` in the script with appropriate values. The `_MODE` variable indicates the mouse action that occurred (0 pressed, 1 dragged, 2 drag-released, and 3 wheeled).

bind [\[mouse-action\]](#) [\[jmol-action\]](#)

Ties a specific mouse action to a specific Jmol action.

bind [\[mouse-action\]](#) "script"

Ties a specific mouse action to a script defined by the user.

where

**[mouse-action]** is any double-quoted combination of a control code (CTRL, ALT, or SHIFT) with a mouse button (LEFT, MIDDLE, RIGHT, or WHEEL) and a click type (SINGLE or DOUBLE)

**[jmol-action]** is one of -- `_clickFrank`, `_depth`, `_dragDrawObject`, `_dragDrawPoint`, `_dragLabel`, `_dragSelected`, `_navTranslate`, `_pickAtom`, `_pickIsosurface`, `_pickLabel`, `_pickMeasure`, `_pickNavigate`, `_pickPoint`, `_popupMenu`, `_reset`, `_rotate`, `_rotateSelected`, `_rotateZ`, `_rotateZorZoom`, `_select`, `_selectAndNot`, `_selectNone`, `_selectOr`, `_selectToggle`, `_selectToggleOr`, `_setMeasure`, `_slab`, `_slabAndDepth`, `_slideZoom`, `_spinDrawObjectCCW`, `_spinDrawObjectCW`, `_swipe`, `_translate`, or `_wheelZoom`

Examples:

```
bind "CTRL-ALT-LEFT" " _rotate";
bind "CTRL-ALT-LEFT-DOUBLE" "spin @(! _spinning)"
```

See also:

[unbind](#)

[↑top](#) [🔍search](#) [📖index](#)

## bondorder

Sets the bond order of the selected atoms or bonds. An alternative to the [connect](#) command.

bondorder 0.5, 1, 1.5, 2, 2.5, 3, 4, -1, -1.5, -2.5

Sets the bond order to the specified order. Values -1, -1.5, and -2.5 specify HBOND, PARTIALDOUBLE (reverse solid/dash of AROMATIC), and PARTIALTRIPLE2, respectively.

bondorder [\[connection-options\]](#)

Sets the bond order to the specified type. See details at [connect](#) for PARTIAL N.M (Jmol 11.4).

where

**[connection-options]** is SINGLE, DOUBLE, TRIPLE, QUADRUPLE, AROMATIC, PARTIAL, PARTIALDOUBLE, PARTIALTRIPLE, PARTIALTRIPLE2, PARTIAL N.M, UNSPECIFIED, or HBOND

See also:

[connect](#) [hbonds](#) [set \(bond styles\)](#) [set \(files and scripts\)](#) [ssbonds](#) [wireframe](#)

[↑top](#) [🔍search](#) [📖index](#)

## boundbox or boundingBox

(v. 11.4 -- expanded capability)

Turns on or off a wire-frame box that contains the model or a designated subset of the model or a designated region of space, and determines the line style and line width (as a decimal number, in Angstroms) of that box. If the atom set is not indicated, the boundbox is drawn around the entire model set (including all models in frames). A decimal number specifies the boundbox line diameter in Angstroms; DOTTED specifies a fine dotted line. For an explanation of center, corners, and vectors, see [show boundbox](#).

boundbox [\[atom-expression\]](#) {default: \*} [\[line-width-or-type\]](#) {default: ON}

Turns the boundbox on or off around the specified set of atoms. In a multi-model context, if a set of atoms is specified and that set of atoms is within a subset of the models, then the boundbox is only displayed only when the [model](#) command has made that subset of models displayable.

boundbox [\[atom-expression-or-coordinate\]](#) [\[xyz-coordinate\]](#) [\[line-width-or-type\]](#) {default: unchanged}

Sets the boundbox to be centered on the atom set or coordinate given as the first parameter and extending by a vector to a corner specified by the second parameter. If the third parameter is not given, no change in the visibility of the boundbox or its line style is made.

boundbox CORNERS [\[atom-expression-or-coordinate\]](#) [\[atom-expression-or-coordinate\]](#) [\[line-width-or-type\]](#) {default: unchanged}

Sets the boundbox based on two corners, each specified by the center off an atom set or a coordinate. If the fourth parameter is not given, no change in the visibility of the boundbox or its line style is made

boundbox TICKS X|Y|Z {major,minor,subminor} FORMAT [%0.2f, ...] SCALE {scaleX, scaleY, scaleZ} | x.xx

Sets the parameters for ticks along the first three boundbox edges. The parameters are similar to those of [axes TICKS](#), but unit cell scaling is not an option.

boundbox SCALE x.xx

With any of the above parameters, specifies to scale the boundbox by (a) multiplying by the scaling factor if positive or (b) adding to the size if negative. (Jmol 12.0)

boundbox \$isosurfaceID

Sets the current boundbox to match the extent of a given isosurface. (Jmol 12.0)

where

**[atom-expression]** is any [expression](#) that evaluates to a set of atoms

**[line-width-or-type]** is a line width or type for a drawing object -- ON, OFF, DOTTED, (integer, 1 to 19), (decimal, <2.0)

**[atom-expression-or-coordinate]** is any [expression](#) surrounded by parentheses or braces, or any {x y z} coordinate

**[xyz-coordinate]** is an xyz coordinate in the form {x y z}

See also:

[axes](#) [getProperty](#) [measure](#) [set \(visibility\)](#) [show unitcell](#)

[↑top](#) [🔍search](#) [📖index](#)

## break

[while](#) and [for](#) loops may be exited using **break** and truncated using **continue**. A number following **break** or **continue** indicates how many levels of for/while loops beyond the innermost one to break out of or continue.

```
n = ("").length
for (var i = 1; i <= n; i++) {
  for (var j = i + 1; j <= n; j++) {
    var dis = ("[]").distance(("[]"));
    if (dis < 1.23) {
      print "short i-j: " + i + "," + j + " " + dis%2
      measure {atomno=i} {atomno=j}
      continue;
    } else if (dis < 1.77) {
      print "medium i-j: " + i + "," + j + " " + dis%2
      line = "line"+i+"-"+j
      draw @line {atomno=i} {atomno=j}
      break 1;
    }
    print "long";
  }
}
```

See also:

[case](#) [catch](#) [continue](#) [default](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [return](#) [switch](#) [try](#) [var](#) [while](#)

[↑top](#) [🔍search](#) [📖index](#)

## calculate

(v. 12.0 -- adds calculations for standard hydrogen bonds, hydrogens, pointgroup, straightness, and struts)

Calculates specific quantities.

calculate AROMATIC

Calculates alternating single and double aromatic bonds for all bonds of type AROMATIC. If just one bond of a conjugated system is specified as AROMATICSINGLE or AROMATICDOUBLE, then all bonds of that system will be consistent with that bond. (Jmol 11.4). For example:

```
reset aromatic;
connect (atomno=3) (atomno=4) AROMATICDOUBLE;
calculate aromatic;
```

or

```
select carbon and within(1.6, [0 0 0]);
connect (selected) aromatic modify;
calculate aromatic;
```

calculate HBONDS [atom-expression] [atom-expression]

If no atom sets are indicated, performs the same as [hbonds calculate](#). Three types of hydrogen bond calculation are available.

RasMol-type pseudo-hbonds (PDB/mmCIF files only)	<b>select not hydrogen; set hbondsRasmol TRUE; calculate HBONDS</b>	Creates pseudo-hydrogen bonds only between protein amide and carbonyl groups or between nucleic acid base pairs using a RasMol-like (DSSP) calculation. Bonds can be differentiated using <a href="#">color hbonds TYPE</a> (see <a href="http://jmol.sourceforge.net/jscolors/#Hydrogen%20bonds">http://jmol.sourceforge.net/jscolors/#Hydrogen bonds</a> or <a href="#">color hbonds ENERGY</a> (red - high energy, less stable; blue - low energy, more stable). This energy is calculated according to the equation given at <a href="http://en.wikipedia.org/wiki/DSSP_(protein)">http://en.wikipedia.org/wiki/DSSP_(protein)</a>
Creates Jmol-type pseudo-hbonds	<b>select not hydrogen; set hbondsRasmol FALSE; calculate HBONDS</b>	pseudo-hydrogen bonds between O or N atoms, but not limited to backbone atoms. Uses an algorithm developed for Jmol involving parameters (see below). These hbonds are not colorable by energy.
Standard hydrogen bonds	all other cases	Creates bonds from hydrogen atoms on O or N to any type of atom other than hydrogen. When two atom sets are used, hydrogen atoms must be present in the first set; other atoms in that set will be ignored. Coloring of standard hydrogen bonds by "energy" is possible, but the value associated with that color should be taken as a true energy.

If two atom sets are provided, hydrogen bonds will be between atoms in the first atom set and atoms in the second atom set. These sets may overlap, and a common invocation of the command is simply **calculate HBONDS**, which uses the currently selected atoms for both sets. When the RasMol calculation is not used, the parameters **hbondsAngleMinimum** (default value 90) and **hbondsDistanceMaximum** (default 3.25 for pseudo-hydrogen bonds; 2.5 for standard hydrogen bonds) are used. The angle referred to is the bond angle between the candidate hydrogen (or pseudo-hydrogen) bond and the covalent bonds to the atoms involved. The idea is that all bond angles that include hydrogen bonds should be greater than some designated value. Note that the default value of 90 degrees may be too large for the general case. Values as low as 70 or 80 may be appropriate in some cases, and experimentation may be necessary. The distance maximum is used only for pseudo-hydrogen bonds when its value is set larger than 2.5.

calculate HYDROGENS [atom-expression] {default: \*}

Adds hydrogens at calculated positions based on bonding patterns at the designated atoms or, if no atoms are specified, at all atoms. Note that this command is not intended to be used for the assignment of hydrogen atoms in proteins. In order to work, all atoms must have any formal charges already designated, and all multiple bonds must be in place.

calculate POINTGROUP

Calculates the point group symmetry for a symmetrical or nearly symmetrical molecule. The calculation is carried out only on the currently selected atoms and is limited to at most 100 selected atoms. The symmetry-determining algorithm looks for proper and improper rotation axes using a variety of methods. In each case, a test is made as to whether all atoms subjected to a specific rotation or reflection map onto the positions of some other atom. The extent to which imperfections in symmetry will be tolerated depends upon two adjustable parameters. **pointGroupDistanceTolerance** (default 0.2 Angstroms) determines the maximum distance between a rotated atom and another atom in the molecule. **pointGroupLinearTolerance** (default 8.0 degrees) determines whether a potential axis matches one that has already been discovered. Setting these values to higher numbers allows more flexibility in terms of atom positions, but also may result in molecules being reported with higher symmetry than they really have.

calculate STRAIGHTNESS

Calculates a value for "straightness" (ranging from -1 to 1) within a biomolecular polymer (protein or nucleic) as defined by the following equation:

$$\text{straightness} = 1 - 2 * (\text{acos}(q_i / q_{i-1} * q_{i+1} / q_i) / \text{PI})$$

where  $q_i$  is the [quaternion](#) defining the frame of the  $i^{\text{th}}$  amino acid or nucleic acid. A value of 1 for straightness indicates that

the three amino acids (i-1, i, i+1) are perfect rotations around the same helical axis. The straightness can then be displayed as part of a label using the %T format code or as a color using [color straightness](#).

calculate STRUCTURE

Recalculates the polymer chains making up a protein or nucleic acid, and then recalculates the secondary structure of proteins and nucleic acids. The results are affected by any bonding that has changed via the **connect** command. A typical use is for PDB files that load with incomplete bonding (because the author specified only a fraction of the bonds). After loading, one can issue **connect** to use Jmol's autobonding feature, then **calculate structure** to recalculate the chains, helices, sheets, turns, and nucleic acid bases based on Jmol's implementation of [DSSP](#). The calculation is performed for all models containing any currently selected atoms. For these models all cartoons and other biomolecular shapes are turned off. The next **cartoons on** command will then show a complete set of cartoons.

calculate STRUTS

Generates [struts](#). Three parameters are used (defaults given): **set strutSpacing 6** sets the minimum spacing between struts, **set strutLengthMaximum 7.0** sets the maximum length that is allowed for a strut, and **strutsMultiple** when set TRUE allows multiple struts on a given atom. In addition, **set strutDefaultRadius 0.3** sets the default radius for struts.

calculate SURFACEDISTANCE FROM [atom-expression]

Calculates for each atom the property surfaceDistance, which is the distance of the atom to a van der Waals surface surrounding the specified subset of atoms of the model. [isosurface sasurface map property surfaceDistance](#) then creates an isosurface colored by distance from the specified subset; or either [color surfaceDistance](#) or [color property surfaceDistance](#) colors the atoms along the same lines.

calculate SURFACEDISTANCE WITHIN [atom-expression]

Calculates for each atom the property surfaceDistance, which is the distance of the atom to a "shrink-wrap" surface surrounding the set of preselected atoms (usually the entire model, often without solvent molecules). Values for atoms OUTSIDE this surface are not generally valid due to the nature of the calculation. Use **calculate surfacedistance FROM {atom expression}** instead for calculating distances outside of a van der Waals surface surrounding a subset of atoms in a model. [isosurface sasurface map property surfaceDistance](#) then creates an isosurface colored by distance from the specified subset; or either [color surfaceDistance](#) or [color property surfaceDistance](#) colors the atoms along the same lines.

where

[atom-expression] is any [expression](#) that evaluates to a set of atoms

See also:

[delete](#)

[top](#) [search](#) [index](#)

## cartoon or cartoons

Cartoons are the classic shapes they are used to depict alpha helices and beta-pleated sheets. A combination of cartoons and [rockets](#) can be displayed using cartoons along with [set cartoonRockets](#). Jmol 11.4 introduces the [set rocketBarrels](#) option, which removes the arrow heads from cartoon rockets.

cartoon ON/OFF {default: ON}

cartoon ONLY

Turns cartoon rendering on and all other rendering off.

cartoon [cartoon-radius]

Starting with Jmol 12.0, a negative number also implies **ONLY**.

where

[cartoon-radius] is the radius of the cartoon elements -- (decimal, <=4.0)

Examples:

See [structure.htm](#)



See also:

[backbone](#) [background dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

[top](#) [search](#) [index](#)

## CASE

(v. 12.0)

See [switch](#).

Note:

The CASE command does not require @{ ... } around Jmol math expressions.

See also:

[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [break](#) [catch](#) [continue](#) [default](#) [echo](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [message](#) [reset](#) [return](#) [set](#) [switch](#) [try](#) [var](#) [while](#)

## CATCH

(v. 12.0 - new)

See [try](#).

Note:

The CATCH command does not require @{ ... } around Jmol math expressions.

See also:

[break](#) [case](#) [continue](#) [default](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [return](#) [switch](#) [try](#) [var](#) [while](#)

[↑](#) [top](#) [🔍](#) [search](#) [📘](#) [index](#)

## cd

Changes the default directory or, with no parameters, displays the default directory. This command simply sets the Jmol parameter **defaultDirectory**. The directory can be local or it can have a URL prefix such as http:// or ftp://, and it may or may not be enclosed in quotes. Note that **x = file(" ")** sets x to the full path to the current default directory while **x = defaultDirectory** sets x to the current default directory as set by the user by the **cd** command (full path) or simply by setting the **defaultDirectory** parameter (which may or may not be the full path).

```
cd
    Displays the default directory.

cd ""
    Resets the default directory to the base directory of the application or applet.

cd "directoryName"
    Changes the default directory to that specified in the directory name. Quotes are optional. The standard notation of two periods indicates "up one level" as in cd ../temp. Note that forward slash, not back slash, should be used to separate directory names along a path.

cd ?
    (signed applet or Jmol application only) Displays a file dialog box allowing for creating and changing directories on the local system.

cd =
    Changes the default directory to that specified in the URL directory specified in the loadFormat variable.
```

See also:

[set \(files and scripts\)](#)

[↑](#) [top](#) [🔍](#) [search](#) [📘](#) [index](#)

## center or centre

Sets the center of rotation to be the center of the set of atoms defined by the [atom expression](#). This is calculated as the mean value of the coordinates of the selected atoms along each of the respective axes. If no atoms are selected then the center is set to the center of the bounding box (the default). The three values can be written using braces as a single coordinate, {x y z}, if desired.

center [\[atom-expression\]](#)

Centers the model on the specified atom set. Along with [show center](#), allows for the reading of the coordinated position of a specific atom using, for example, **center [CYS]4.O;show center;center**. See also [centerAt](#)

center [\[xyz-coordinate\]](#)

Centers the model on a specified model-frame coordinate

center [\[drawn-object\]](#)

Centers the model on a specified [draw](#) object

center

Recenters the model on the default center.

where

**[atom-expression]** is any [expression](#) that evaluates to a set of atoms

**[xyz-coordinate]** is an xyz coordinate in the form {x y z}

**[drawn-object]** is a drawn object -- \$name

Examples: [in new window using 1crn.pdb](#)

```
select [CYS]32
center selected
select *
```

See also:

[centerAt](#) [show](#)

[↑](#) [top](#) [🔍](#) [search](#) [📘](#) [index](#)

## centerAt

The **centerAt** command allows centering of the model in one of three different ways: based on an absolute coordinate position, based on an offset relative to the center of the bounding box (the overall application default), or based on an offset relative to the average position of the currently selected atoms. Centering on a specific atom or atom set without first selecting it is also available using the [center](#) command. If the three numerical values are omitted, they default to 0.0 0.0 0.0. The numbers can be in the form of a coordinate (with braces), {x y z}, if desired.

centerAt ABSOLUTE x y z {default: 0.0 0.0 0.0}

specifies an absolute coordinate for the center, in Angstroms

centerAt AVERAGE x y z {default: 0.0 0.0 0.0}

relative to the average atom position (also known as the "unweighted center of gravity")

centerAt BOUNDBOX x y z {default: 0.0 0.0 0.0}

relative to the center of the boundingbox, which is defined by the minimum and maximum atom center coordinates along each of the cartesian axes

Examples:

```
centerAt absolute 1.0 2.0 3.0
centerAt boundbox 1.0 2.0 -3.0
centerAt average 0.0 0.0 0.0
```

See also:

[center](#) [show](#)

## color or colour

(v. 11.4 -- adds fully customizable color schemes)

```
[object]
[translucent/opaque]
[color, property, or color scheme]
Color Inheritance
```

In general, the color command takes the following syntax:

**color [object] [translucent/opaque] [color, property, or color scheme]**

Colors can be designated as one of the [\[standard JavaScript colors\]](#), as a red-green-blue triplet in square brackets, [255, 0, 255], as a red-green-blue triplet expressed as a six-digit hexadecimal number in brackets, [xFF00FF], as a triplet expressed as a point in red-green-blue color space (Jmol 11.4) -- {255,0,255} or fractional {0.5, 0.5, 1} (Jmol 12.0). To specify a set of atoms to color, you can either select them first -- **select \*.N?**; **color green** -- or specify them using [atom expression](#) notation: **color {\*.N?} green**.

**[object]** [back](#)

The color command takes several forms, depending upon the type of object being colored: an atom object, a bond object, a chemical element, or a model object. This section of the guide discusses each of these in turn:

- [color \(atom object\)](#)
- [color \(bond object\)](#)
- [color \(element\)](#)
- [color \(model object\)](#)
- [color \(named object\)](#)
- [color \(scheme\)](#)

Additional information external to this documentation can be found in relation to [\[Jmol color schemes\]](#) and [\[standard JavaScript color names and codes\]](#). In addition, a page is available that lays out the [\[Jmol color command matrix\]](#). Color schemes may be customized using the [set userColorScheme](#) command.

**[translucent/opaque]** [back](#)

The color command allows an optional color modifier of TRANSLUCENT or OPAQUE, which can be used with any object, either alone or with a color. Starting with Jmol 11.2, TRANSLUCENT can take an integer in the range 0-6 (indicating eighths -- 0, 1/8, 2/8, etc.), 32-255 (indicating fraction of 256), or a decimal in the range 0.0 to 1.0. Larger numbers are more translucent -- dimmer. Currently implemented translucencies are -1 (Jmol 10 translucency), 0 (opaque), 0.125 (1, 32), 0.25 (2, 64), 0.375 (3, 96), 0.5 (4, 128), 0.625 (5, 160), and 0.75 (6, 192). Future versions of Jmol may include more options, so it is recommended that the decimal numbers be used. The default is TRANSLUCENT 0.5, which can be set using "defaultTranslucent = x.xx", where x.xx is a decimal number. For example:

```
color atoms TRANSLUCENT orange
color ribbons TRANSLUCENT 0.5 [255, 165, 0]
select oxygen; color opaque.
```

If neither TRANSLUCENT nor OPAQUE is specified, OPAQUE is assumed. Thus, **color atoms red** and **color atoms OPAQUE red** are synonymous.

**[color, property, or color scheme]** [back](#)

Besides the standard ways of representing a color, Jmol allows coloring based on properties or by one of the known [Jmol color schemes](#). If **color TEMPERATURE** is used, the color range will depend upon the setting of the [rangeSelected](#) flag. A new option for Jmol 11.0 is SURFACEDISTANCE.

Starting with Jmol 11.2, you can also color atom-based objects based on user-defined properties read from an external file or standard Jmol atom properties such as partialCharge that Jmol has read from the model file. In the case of Jmol atom properties, use the keyword PROPERTY followed by the name the property, as in **color atoms PROPERTY partialCharge**. In the case of user-defined properties (created using the [data](#) command), which always start with "PROPERTY\_", simply give the property name: **x = load("mydata.txt");data "property\_mydata @x";select model=2;color atoms property\_mydata**. When using either PROPERTY or PROPERTY\_xxx, you can set the absolute range of values that will span the full spectrum of colors of the propertyColorScheme you have chosen. Simply add the keyword ABSOLUTE followed by the minimum and maximum values you wish to use for the two ends of the spectrum. So, for example: **color atoms property temperature ABSOLUTE 0.0 30.0**.

**Color Inheritance** [back](#)

Many objects inherit both color and opacity from the underlying associated atom (which, themselves "inherit" their color by default from their associated chemical element). For example, by default a bond will inherit the two colors+translucencies of its endpoint atoms. If you simply 'color atoms translucent', then both the atoms and the bonds will be translucent. But if you 'color bonds opaque' or 'color bonds red' and also 'color atoms translucent' only the atoms will be translucent.

Starting with Jmol 11.2, the level of 'translucent' can be controlled. The algorithm allows for proper mixing of colors for two translucent objects and approximately correct mixing for more than two overlapping translucent objects.

color [\[color-scheme\]](#)

Sets the previously selected atom set to a color based on a color name or value or one of the [\[Jmol color schemes\]](#), namely one of "roygb" (default rainbow), "bwr" (blue-white-red), "rwb" (red-white-blue), "low" (red-green), or "high" (green-blue). Jmol 12.0 adds "bw" (black-white) and "wb" (white-black).

Examples:

```
define ~myset (*.N?)
select ~myset
color green
select *
color cartoons structure
color rockets chain
color backbone blue
```

color [\[color-scheme\]](#) TRANSLUCENT

The additional parameter TRANSLUCENT creates a gradient of translucency from transparent to opaque across the color scheme. (Jmol 12.0)

Examples:

```
define ~myset (*.N?)
select ~myset
color green
select *
color cartoons structure
color rockets chain
color backbone blue
```

where

**[color-scheme]** is to color based on a [\[Jmol color scheme\]](#). (CPK and NONE are synonymous here) -- (color name), [r, g, b], [xRRGGBB], ALTLOC, AMINO, CHAIN, CPK, FIXEDTEMPERATURE, FORMALCHARGE, GROUP, INSERTION, JMOL, MOLECULE, MONOMER, NONE, PARTIALCHARGE, RASMOL, RELATIVETEMPERATURE, SHAPELY, STRUCTURE, SURFACEDISTANCE

## color (atom object)

Sets the color of atom-related objects (atoms, backbone, cartoons, dots, halos, labels, meshribbon, polyhedra, rockets, stars, strands, struts (Jmol 12.0), trace, and vibration vectors).

color [\[atom-associated-object\]](#) [\[color-scheme\]](#)

Sets the color of atom-related objects based on a previously selected atom set to a specific color, a color scheme, or back to its default color (CPK), or to inherit the color of its associated atom (NONE). In the case of "color atoms", CPK and NONE both revert to the default color. In the case of "color labels", coloring a label to the background color automatically uses the background contrast color, not the actual background color (white or black, depending upon the background color). If it is desired for some reason to color a label the background color, then the label should be colored instead a color very close to the background color but not it exactly. For instance, to color labels black with a black background, use [0,0,1] instead of [0,0,0].

Examples: [in new window using caffeine.xyz](#)

```
select oxygen;color atoms green
select carbon;color atoms TRANSLUCENT white;color bonds green;
select carbon;color bonds OPAQUE [255,196,196]
```



See [atoms.htm](#) [bonds.htm](#)



where  
**[atom-associated-object]** is an object related to an atom -- ATOM, BACKBONE, CARTOON, DOTS, ELLIPSOID, HALOS, LABELS, MESHRIBBON, POLYHEDRA, RIBBONS, ROCKETS, SELECTIONHALOS, STARS, STRANDS, STRUTS, TRACE, VECTORS  
**[color-scheme]** is to color based on a [\[Jmol color scheme\]](#). (CPK and NONE are synonymous here) -- (color name), [r, g, b], [xRRGGBB], ALTLOC, AMINO, CHAIN, CPK, FIXEDTEMPERATURE, FORMALCHARGE, GROUP, INSERTION, JMOL, MOLECULE, MONOMER, NONE, PARTIALCHARGE, RASMOL, RELATIVETEMPERATURE, SHAPELY, STRUCTURE, SURFACEDISTANCE

See also:  
[background color \(bond object\) color \(element\) color \(model object\) color \(other\) color measures label set userColorScheme show](#)

[top](#) [search](#) [index](#)

## color (bond object)

Three types of bonds are distinguished by Jmol for coloring purposes: regular bonds, disulfide bonds, and hydrogen bonds. Each can be colored independently, and hydrogen bond colors in proteins can take on a special coloring scheme based on their connectivity.

color BONDS [\[color-none-CPK\]](#)  
 Colors selected bonds a specific color or resets them to inherit the color of their associated atoms.  
 color SSBONDS [\[color-none-CPK\]](#)  
 Colors disulfide bonds a specific color or resets them to inherit their color from their associated atoms.

color HBONDS [\[color-none-CPK\]](#)  
 Colors hydrogen bonds a specific color or resets them to inherit their color from their associated atoms. Additional HBOND color options include TYPE and ENERGY

color HBONDS ENERGY  
 Colors hydrogen bonds based on DSSP "energy" (see [calculate HBONDS](#) for details).  
 color HBONDS TYPE

Colors hydrogen bonds specifically in proteins based on how many residues one end is from the other. Note that to get this effect, one must first execute "hbonds ON" and then issue "color hbonds TYPE". The colors assigned are based on the number of residues between the interacting H atoms. This TENDS to indicate secondary structure, but is not perfect. The correlation between color and offset are as follows:

Color	Offset
green	-4
cyan	-3
white	+2
magenta	+3 (turns)
red	+4 (alpha-helix)
orange	+5
yellow	other (e.g. beta-pleated sheet)

where  
**[color-none-CPK]** is (color name), [r, g, b], [xRRGGBB], CPK, NONE

See also:  
[background color \(atom object\) color \(element\) color \(model object\) color \(other\) color measures hbonds set userColorScheme show](#)

[top](#) [search](#) [index](#)

## color (element)

You can use the 'color' command to specify customized default colors that are used for elements. [\[default Jmol element colors\]](#)

color carbon limegreen

color hydrogen [x32CD32];

These changes are not molecule-specific; they will continue in effect even if new molecules are loaded. However, in a page with multiple applets, each applet will have its own set of element colors.

If you choose to use this feature, you should consider encapsulating your favorite colors into a script and then executing that script as a subroutine. For example:

```
script LoadMyFavoriteColors.txt;
load foo.xyz;
load bar.xyz;
```

Note:

1. Custom element colors are independent of and are not affected by the currently selected set of atoms.
2. To reset custom element colors, use '[set defaultColors Jmol](#)' or 'set defaultColors Rasmol'.
3. 'translucent' or 'opaque' cannot be specified as part of the element color specification. (You cannot 'color carbon transparent green', for instance.)
4. At this time only elements can be custom colored. There is no support for customizing other color palettes such as those used for protein chains or groups.

color [\[element-name\] \[RGB-color\]](#)

where  
**[element-name]** is to color specific elements such as CARBON or HYDROGEN -- (element name)  
**[RGB-color]** is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]

See also:  
[background color \(atom object\) color \(bond object\) color \(model object\) color \(other\) color measures set userColorScheme show](#)

[top](#) [search](#) [index](#)

## color (model object)

Sets the color of various model objects.

color [\[model-object\] \[RGB-color\]](#)

Examples: [in new window using 1crn.pdb](#)



```
select [CYS]32 ;label %a: %x %y %z;color labels white;
set axes on;color axes green
set axes on;color axes [xFF00FF];
```

where  
**[model-object]** is AXIS1\*, AXIS2\*, AXIS3\*, AXES, BOUNDBOX, DRAW, ECHO, GEOSURFACE, HOVER, ISOSURFACE, LABEL, MEASUREMENTS, MO, PMESH, POLYHEDRA, SELECTIONHALOS, UNITCELL (\*starting with Jmol 11.1.20)  
**[RGB-color]** is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]

Examples: [in new window using 1crn.pdb](#)



```
select [CYS]32 ;label %a: %x %y %z;color labels white;
set axes on;color axes green
set axes on;color axes [xFF00FF];
```

See also:  
[background color \(atom object\) color \(bond object\) color \(element\) color \(other\) color measures set userColorScheme show](#)

[↑top](#) [🔍search](#) [📖index](#)

## color (named object)

Sets the color of an object created using [draw](#), [isosurface](#), [pmesh](#) or [polyhedra](#) using the name identifier preceded by a dollar sign (\$).

color [[drawn-object](#)] [[RGB-color](#)]

where

**[drawn-object]** is a drawn object -- \$name

**[RGB-color]** is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]

[↑top](#) [🔍search](#) [📖index](#)

## color (other)

(v. 11.0 -- new)

Sets the default color of the halos displayed by [selectionHalos](#). The default default selection halo color is GOLD. To assign colors based on the underlying atoms, use **color selectionHalos NONE**. This command setting persists for the life of the applet or application, like element colors. If the atom's halo color has been set using **select ...; color halos ...**, then **color selectionHalo** has no effect until those colors are returned to their default settings with **color halos none**.

color HIGHLIGHT [[RGB-color](#)]

Sets the color of the [highlight](#) ring.

color SELECTIONHALOS [[color-none-CPK](#)]

Sets the default color of the halos displayed by [selectionHalos](#). The default default selection halo color is GOLD. To assign colors based on the underlying atoms, use **color selectionHalos NONE**. This command setting persists for the life of the applet or application, like element colors. If the atom's halo color has been set using **select ...; color halos ...**, then **color selectionHalo** has no effect until those colors are returned to their default settings with **color halos none**.

where

**[color-none-CPK]** is (color name), [r, g, b], [xRRGGBB], NONE

**[RGB-color]** is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]

See also:

[background color \(atom object\)](#) [color \(bond object\)](#) [color \(element\)](#) [color \(model object\)](#) [color measures](#) [set userColorScheme show](#)

[↑top](#) [🔍search](#) [📖index](#)

## color (scheme)

Sets the working color scheme and range for determining correlations between property values and color values. The names "user" and its reverse, "resu", refer to a [user-defined color scheme](#). The RANGE values indicate what parameter values correspond to the first and last color values. The range values are required unless they have been set already in a previous command such as [isosurface](#). Once set, calculations such as **x = {atomno=3}.partialcharge.color** return a color value based on this color scheme and range. Starting with Jmol 11.4, you can implement any number of your own color schemes simply by including "=" after the name and adding a set of hexadecimal color values: **color "myScheme=[x00FF00] [xFFFF00] [x00FF00] [x00FFFF] [x0000FF]"**, for example. Then, **color atoms "myScheme"** will do just that. Starting with Jmol 11.4, you can redefine the Jmol coloring schemes as well simply by using "jmol=[x.....]..." or "rasmol=[x.....]..." as the name. To return these to their default values, include the equal sign but no values. Starting with Jmol 11.4, four additional built-in color schemes include "byElement\_Jmol", "byElement\_Rasmol", "byResidue\_Jmol" (corresponding to [color shapely](#)), and "byResidue\_Rasmol" (corresponding to [color amino](#)). Setting the range has no effect on these color schemes, as they are intended to correspond with specific elements (elemno) and residues (groupID). You can implement your own byElement or byResidue color schemes simply by using those keywords at the beginning of a color scheme definition. byElement color schemes start with "unknown", then run through the periodic table; byResidue color schemes start with "no group", then the amino acids and nucleic acids in order: ALA, ARG, ASN, ASP, CYS, GLN, GLU, GLY, HIS, ILE, LEU, LYS, MET, PHE, PRO, SER, THR, TRP, TYR, VAL, ASX, GLX, UNK, A, +A, G, +G, I,

+I, C, +C, T, +T, U, +U.

color "colorSchemeName" RANGE [min] [max]

[↑top](#) [🔍search](#) [📖index](#)

## color measures

Colors the measurement numbers and dotted lines. In Jmol 10.2, "color measures" change all measurement colors at once. In version 11.0, "color measures" acts on all future measures, allowing for selective coloring of measurements. Thus, "color measure" in 11.0 acts on (a) any measures currently with no color assigned and (b) on any future measures. If measurement colors have already been set, then "color measures NONE" needs to be invoked to turn off measurement colors prior to resetting them.

color measures [[RGB-color](#)]

where

**[RGB-color]** is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]

See also:

[background color \(atom object\)](#) [color \(bond object\)](#) [color \(element\)](#) [color \(model object\)](#) [color \(other\)](#) [set userColorScheme show](#)

[↑top](#) [🔍search](#) [📖index](#)

## compare

(v. 12.0 -- new)

Compares two models and optionally reorients the first model relative to the second based on a given atom-atom coordinate pairing or quaternion-based group-group orientation pairing. References to the atom-atom correlation algorithm can be found in the literature [1] and [2]. Quaternion-based orientation pairing is an unpublished technique specific to Jmol at this point. It minimizes the standard deviation of the correlated quaternion frames for groups in the two models using spherical averaging. (Results of this option depend upon the setting of set quaternionFrame.)

By default the command does not move any atoms and just reports RMSD. The independent options ROTATE and TRANSLATE allow the option to do just rotation, do just center-of-mass translation, or do both.

compare {model1} {model2} SUBSET {atomSet} ATOMS [paired atom list]

Compares the specified subset of atoms of two models, minimizing the RMSD of paired atom coordinates. If the SUBSET option is not specified, the models are matched atom-for-atom based on "SPINE" atoms (\*.CA, \*.N, \*.C for proteins; \*.P, \*.O3\*, \*.O5\*, \*.C3\*, \*.C4\*, \*.C5 for nucleic acids). The keyword ATOMS is optional and may be followed by any number of specific pairs of atom sets to be correlated. For example, **compare {2.1} {1.1} SUBSET (\*.CA) ATOMS {10-20} {50-60} {30-40} {20-30} ROTATE TRANSLATE** will correlate the positions of alpha carbon atoms in groups 10-20 and 30-40 of model 2.1 with corresponding atoms in groups 50-60 and 20-30 of model 1.1 and move all atoms in model 2.1 in the process. The result of this atom-atom pairing comparison is essentially the same as the PyMol pair\_fit command, though easier to implement and using an exact form of the structure-structure correlation rather than an iterative process.

compare {model1} {model2} ORIENTATIONS [paired atom list]

Compares the specified subset of atoms of two models, minimizing the RMSD of paired group quaternion orientations. If no other parameters are included, the models are matched group for group based on the current setting of **quaternionFrame**. The keyword ORIENTATIONS may be followed by any number of specific pairs of atom sets to be correlated. For example, **set quaternionFrame "C"; compare {2.1} {1.1} QUATERNIONS {10-20 or 30-40} {50-60 or 80-90} ROTATE TRANSLATE** will correlate the orientations of alpha carbon atoms in groups 10-20 and 30-40 of model 2.1 with corresponding orientations in groups 50-60 and 80-90 of model 1.1 and move all atoms in model 2.1 in the process. The result of this orientation pairing comparison gives the best fit of orientations and the best translation, but not necessarily the best rotation to fit coordinate positions. The ORIENTATION option has no corresponding PyMol command.

compare {model1} {model2} ORIENTATIONS [paired quaternion array list]

The ORIENTATION option allows for explicit comparison of quaternion arrays rather than atom lists. The result is independent of the setting of **quaternionFrame**. These arrays can be specified in terms of variables. For example: **qset1 = quaternion({10-20:A/1.1}); qset2 = quaternion({20-30:D/2.1}); compare {2.1} {1.1} ORIENTATIONS @qset2 @qset1**.

compare {model1} {model2} SMARTS or SMILES "smartsString"

(with the ROTATE and TRANSLATE options) Jmol 12.0 adds the capability to align two structures based on SMILES or SMARTS atom matching. The basic idea is to use a SMILES (whole molecule) or SMARTS (substructure) description to find the atoms in one structure that correlate one-for-one with atoms in the second structure, then find the rotation and translation that best aligns them. If no actual atom moving is desired, you can get the standard deviation alone using the compare() function with the STDDEV option. A return of "NaN" indicates that the desired SMILES/SMARTS match could not be made in one or the other structure.

[top](#) [search](#) [index](#)

## configuration or conformation or config

File types PDB, mmCIF, and CIF allow for the designation of certain atoms to be in "alternative locations" or in "disorder groups". This leads to two or more possible structures. While full treatment of this issue is not possible, Jmol can display the different possible "configurations" described in these files. for PDB and mmCIF files. This command selects the specified configuration (but does not [#.restrict] or [#.display] it, and also resets all biomolecular shapes (cartoons, traces, etc.) to use the values of that configuration. See also the atom property [configuration](#), which allows selection of configurations without updating shapes (Jmol 12.0).

configuration [configuration number]

[top](#) [search](#) [index](#)

## connect

[minimum and maximum distances]  
[source and target atom sets]  
[bond type]  
[radius option]  
[color option]  
[modify/create option]

The **connect** command allows real-time bond manipulation, allowing the user or application to connect and disconnect specific atom sets. The general syntax is as follows:

**connect** [minimum and maximum distances] [source and target atom sets] [bond type] [radius option] [color option]  
[modify/create option]

(**connect** by itself deletes all bonds and then creates bonds based on Jmol default bond-generating algorithms, all as single bonds, without respect to what bonding patterns might have been indicated in the model file.)

[minimum and maximum distances] [back](#)

Distances are given in Angstroms, either as decimals or integers. If only one distance parameter is given, it represents a maximum distance. If neither the minimum nor the maximum distance parameter is given, all connections between the two atom sets are made, regardless of distance. Starting in Jmol 12.0, the option to connect atoms based on ranges of percentage of bonding/ionic radii instead of fixed values can be specified using a % sign after a distance.

[source and target atom sets] [back](#)

The source and target atom sets are embedded [atom expressions](#) and therefore must be enclosed in parentheses. If the source atom set is not given, it is taken to be the currently selected atom set, "(selected)". If neither atom set is given, "(selected) (selected)" is assumed. Starting in Jmol 11.4, the atom expression "\_1" in the second selection set signifies "the atom selected in the first set". Thus, it is possible to select something like (**\_N**) (**\_O and not within(chain, \_1)**) -- meaning "all nitrogens and all oxygens not in the same chain as the selected nitrogen." Of course, this would be used with a distance qualifier.

[bond type] [back](#)

Unless otherwise specified, connections are automatically introduced as single bonds. Any one of the following bond types may be specified: SINGLE, DOUBLE, TRIPLE, QUADRUPE, AROMATIC, PARTIAL, PARTIALDOUBLE, HBOND, STRUT (Jmol 12.0) or UNSPECIFIED. In appearance, AROMATIC and PARTIALDOUBLE are identical except for which side of the bond is represented by a dashed line. PARTIAL and HBOND are both dashed, but they have different patterns, and newly created hydrogen bonds are only thin lines. Jmol 11.4 adds PARTIALTRIPLE, PARTIALTRIPLE2, AROMATICSINGLE, AROMATICDOUBLE, numeric bond order (including -1 for PARTIAL, 1.5 for AROMATIC, -1.5 for PARTIALDOUBLE, 2.5 for PARTIALTRIPLE, and -2.5 for PARTIALTRIPLE2), as well as a fully generalized set of partial bonds indicated with

connect ... partial N.M

where N is the number of lines (from 1 to 5) and M is a binary mask indicating which lines are dashed:

M	binary	meaning
1	00001	first line dashed
2	00010	second line dashed
3	00011	first and second lines dashed
4	00100	third line dashed
...	...	...
31	11111	all lines dashed

So, for example, we have:

partial 1.0	single
partial 1.1	same as "partial"
partial 2.0	double
partial 2.1	same appearance as "aromatic", though not "aromatic"
partial 2.2	partialDouble
partial 3.0	triple
partial 3.1	partialTriple
partial 3.4	parialTriple2

[radius option] [back](#)

Addition of the keyword "radius" followed by a distance in angstroms allows definition of the radius of a modified or newly created bond. If the modify/create option is absent, then "modify" is assumed; if the bond type is absent, then bonds of any type are set, but their bond type is not changed.

[color option] [back](#)

Addition of a color name or designation optionally along with the keyword "translucent" or "opaque" allows definition of the color and/or translucency of a modified or newly created bond. If the modify/create option is absent, then "modify" is assumed; if the bond type is absent, then bonds of any type are set.

[modify/create option] [back](#)

Four additional mutually exclusive options relate to what sort of connections are made. The default when a radius or color option is present is "Modify"; otherwise the default is "ModifyOrCreate". These include:

<b>Create</b>	Only new bonds will be created. If a bond of any type already exists between two matching atoms, it will not be affected.
<b>Modify</b>	Only pre-existing bonds will be modified. No new bonds will be created.
<b>ModifyOrCreate</b>	If the connection fits the parameters, it will be made. Bonds already present between these atoms will be replaced.
<b>Delete</b>	Delete the specified connections.

Examples: [in new window using caffeine.xyz](#)

```
# two ways to link atom #1 with atom#2
connect (atomno=1) (atomno=2) DOUBLE
select atomno=1,atomno=2; connect (selected)
```

```
# connect all carbons with hydrogens that are within the range 1.0 to 1.2 angstroms and are not already connected
connect 1.0 1.2 (carbon) (hydrogen) PARTIAL.CREATE
```

```
# change all bonds to single bonds
```



connect (all) (all) SINGLE MODIFY

# connect every atom with every other atom that is within 1.5 angstroms whether it is connected already or not  
connect 1.5 (all) (all) ModifyOrCreate

# delete all bonds WITHIN a selected atom set  
connect (selected) (selected) DELETE

# delete all bonds TO a selected atom set  
connect (selected) (not selected) DELETE

See also:

[bondorder](#) [hbonds set \(bond styles\)](#) [set \(files and scripts\)](#) [ssbonds](#) [wireframe](#)

[top](#) [search](#) [index](#)

## console

Throws up a console window from which a user can enter script commands and monitor the messages returned by Jmol as, for example, from the [show](#) or [getProperty](#) command.

[top](#) [search](#) [index](#)

## continue

See [break](#).

See also:

[break](#) [case](#) [catch](#) [default](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [return](#) [switch](#) [try](#) [var](#) [while](#)

[top](#) [search](#) [index](#)

## data

[Setting atom properties](#)

The **data** command allows data to be introduced in-line or via a variable. The command consists of two statements, **data "label"** and **end "label"**, between which the data are presented on as many lines as desired. "label" may be any string, though strings starting with "property\_" are special (see below). Quotes must be used in both the **data** line and the **end** line. The first word of the label defines the data type, but the label itself may be any number of words. If the data type is "model" as in the following example, then the data is interpreted as an in-line model (and loaded using the default lattice, if crystallographic). If the data type is "append", then the data is interpreted as a model, and the model is appended either into a new frame or into the last existing frame, based on the setting of **set appendNew**. Additional data types may be loaded and later [shown](#), but they will be ignored by Jmol.

```
background red;
data "model example"
2
testing
C 1 1 1
O 2 2 2
end "model example";show data
```

Note that the **data** statement itself should not include a semicolon at the end. In the specific case of a model file, if it is desired to use no new-line characters, you can start the data with | (vertical bar) and then use a vertical bar to separate all lines: **data "model example"|2[testing]C 1 1 1|O 2 2 2[end "model example";show data**. For this option you MUST start the data with a vertical bar immediately following the quotation mark closing the label or on the very next line. If the first character is a vertical bar or a new-line character, it is not part of the model. To include data representing more than one file, first define a data separator using **set dataSeparator**, for example, **set dataSeparator "----"**. Then use that separator between data sets. The **data** command thus provides an alternative to the JavaScript Jmol.js (applet-only) **loadInline()** function. It can be included in any [script](#), and commands can come before

and after it for further processing. Note that model data in the system clipboard can also be pasted into the applet console or entered into the application using Edit...Paste for direct introduction into Jmol. Starting with Jmol 12.0, using [load data](#) instead of just **data** you can load model data with all of the loading options of the standard LOAD command. See also [show data](#), [getProperty data](#), and [load "@x"](#).

**Setting atom properties** [back](#)

Atom property data may also be loaded into Jmol using the data command. In addition, some readers can create their own property\_XXX data. To assign property data, first select the atoms to which data are to be assigned. Then, to assign the data, use **DATA "property\_XXX"**, where "XXX" is any alphanumeric string. Data are assigned sequentially to the currently selected atom set, and will be assigned to atoms in the selection set one after another until either the data or the atom list is exhausted. If the data are exhausted before the selected atoms, then the value 0 is recorded for each of the remaining selected atoms. In this way, if only a few atoms need data, only a few can be selected and assigned values. Properties can be checked using atom labels (**label %(property\_XXX)**) The following special values for XXX read data into Jmol exactly as though read from a model file: coord, formalCharge, occupancy (0 - 100), partialCharge, temperature, valence, vanDerWaals, and vibrationVector. (Jmol 11.8 adds atomName, atomType, and element.) Any other label will be saved simply under its property name.

To read selected data that is in free-field-format (for example, from a spreadsheet), specify which field the data is in using **set propertyDataField = n**, where n > 0, prior to the data command. Lines having fewer than n tokens will be ignored. If the data is fixed-column format, then n is the starting column number, and set **propertyDataColumnCount** to be the number of columns to assign to this field. Lines shorter than the required number of characters will be ignored. Setting **propertyDataField** to 0 indicates that no data field is present, and data are to be read sequentially.

Atom selection need not be contiguous. If you want to associate specific data with specific atom numbers, a column containing these atom numbers (starting with 1 for each model loaded) can be specified using **propertyAtomNumberField = m**. If the data is fixed-column format, then m is the starting column number, and also set **propertyAtomColumnCount** to be the number of columns to assign to this field. Specifying 0 for m indicates that the set of currently selected atoms should be assigned values from the data.

Atom property data may also be loaded from variables. To do this, use add "@x", where x is a variable name within the quotation marks defining the first parameter of the data command: **DATA "property\_partialCharge @x"** or **DATA "property\_mydata 66 3 @x"**. No end line is required. The variable x should already contain a list of numbers, perhaps from using the x = [load](#)("myfile.dat"); perhaps just by creating a string of numbers: x = "2.3 3.4 5.6 7.8...".

The [data\(\)](#) function also allows direct conversion of data into arrays, which can be directly stored in an atom property using, for example,

```
{*.CA/2.1}.temperature = data(load("mydata.dat"),6,0,3)
```

meaning, "Read the file mydata.dat and store the sixth free-format field of each line starting from the third line as the temperature of the C-alpha carbons of model 2.1."

p>data "label"  
Defines a set of data in line, ending with a matching **end "label"**, where "label" is any string. Quotes are required. Certain labels have special meaning and are described more fully below.  
data "label @x"  
Defines a set of data with the given label from a variable (variable x in this case). Quotes are required, but no **end** command is required.  
data "data2d\_XXX"  
Defines a data set to be paired x,y data inline and ending with **end data2d\_XXX**, where XXX can be any alphanumeric string.  
data "property\_XXX propertyAtomField propertyDataField"  
Defines an atom property based on data provided next inline and ending with **end property\_XXX**, where XXX can be any alphanumeric string. **propertyAtomField** and **propertyDataField** are optional, and if provided override the set values for **propertyAtomField** and **propertyDataField**.  
data "property\_XXX propertyAtomField propertyAtomColumnCount propertyDataField propertyDataColumnCount"  
Defines an atom property based on data provided next inline and ending with **end property\_XXX**, where XXX can be any alphanumeric string. **propertyAtomField**, **propertyAtomColumnCount**, **propertyDataField**, and **propertyDataColumnCount** override set values for these parameters.  
data CLEAR  
Clears the data table.  
data element\_vdw 6 1.7; 7 1.8 END element\_vdw  
Defines the USER set of van der Waals radii on an element-by-element basis. Entries may be separated by semicolons or entered one per line. (Jmol 11.6). In the example given here, carbon (atomic number 6) is given a radius of 1.7, and nitrogen (atomic number 7) is given a radius of 1.8.

See also:

[set \(misc\)](#) [spacefill](#)

[top](#) [search](#) [index](#)

## default

(v. 12.0)

See [switch](#).

See also:

[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [break](#) [case](#) [catch](#) [continue](#) [echo](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [message](#) [reset](#) [return](#) [set](#) [switch](#) [try](#) [var](#) [while](#)

[top](#) [search](#) [index](#)

## define or @

Defines the specified variable to the the atoms selected by the [atom expression](#). In general, definitions are "static" in the sense that when they are generated with the **define** command, they represent the matching set of atoms indefinitely. However, if "DYNAMIC\_" is at the beginning of the variable name, as, for example, "DYNAMIC\_myatoms", then whenever that variable is used it is recalculated. (When used, the prefix "DYNAMIC\_" should be dropped.)

IMPORTANT NOTE: The **define** command should be used with some discretion. If you should define a term that later becomes a reserved keyword to any command in any future version of Jmol, your page may not be compatible with that future version. A simple way to avoid this situation is to put a tilde (~) in front of any definition you make. **Do not use a leading underscore (\_)**, as there are many "hidden" reserved definitions that start with that character.

define [\[variable-name\]](#) [\[atom-expression\]](#)

where

[\[variable-name\]](#) is a (string)

[\[atom-expression\]](#) is any [expression](#) that evaluates to a set of atoms

Examples: [in new window using 1blu.pdb](#)

```
define ~mygroup within(5.0,[FS4]102)
select ~mygroup
color atoms white
```

See [select.htm](#)

See also:

[initialize](#) [refresh](#) [reset](#) [restore](#) [save](#) [zap](#)

[top](#) [search](#) [index](#)

## delay

Causes the screen to refresh and the script to stop executing for the specified number of seconds.

delay [\[time-delay\]](#)

delay on

where

[\[time-delay\]](#) is in seconds -- (integer|decimal, >=0)

See also:

[exit](#) [goto](#) [loop](#) [pause](#) [quit](#) [resume](#) [step](#)

[top](#) [search](#) [index](#)

## delete

The DELETE command deletes atoms specified in the same as for the [select](#) command. For example: **DELETE hydrogen** or **DELETE atomno > 30**

delete

See also:

[calculate](#)

[top](#) [search](#) [index](#)

## depth

Slab and Depth together control the percentage of the molecule to be displayed based on clipping planes. See [{#.slab-}](#).

Examples: [in new window using 1crn.pdb](#)

```
slab 50; depth 0;slab on; # show the back half of the molecule
slab 100;depth 50; slab on;# show the front half of the molecule
slab 75; depth 25;slab on; # show middle 50% of the molecule
slab 50;depth 50;slab on; # show a plane that is 1 pixel deep
```

See also:

[slab](#)

[top](#) [search](#) [index](#)

## dipole or dipoles

ID [\[object id\]](#)  
[\[modifying parameters\]](#)  
[\[positions\]](#)

The **dipole** command allows for the drawing of a bond or molecular dipole arrow with or without a cross near the tail. Note that without the cross, since it can be drawn from any point in molecular space to any other, a "dipole" can be used by a web page developer for a simple arrow in order point to some particular aspect of the model having nothing to do whatsoever with dipole moments. The values of each dipole can be set by the user or will be estimated using partial charge data or molecular dipole information if available in the loaded model file. Only a very crude calculation is used to estimate at least the direction of all bond dipoles. The general syntax of the **dipole** command is as follows:

**dipole** [\[objectID\]](#) [\[modifying parameters\]](#) [\[positions\]](#)

**ID** [\[object id\]](#) [back](#)

The optional identifier such as "bond1" that can be referred to in later scripts as \$bond1. These words are arbitrary and, starting in Jmol 11.6, if preceded by the optional keyword ID may be any string. The special identifier **BONDS** (without the keyword ID) refers to the entire collection of bond dipoles -- those dipoled defined specifically as between two atoms. Similarly, the special identifier **MOLECULAR** (without the keyword ID) is primarily for files for which molecular dipole information is available. The value and placement of this dipole can also be set using the **dipole** command.

[\[modifying parameters\]](#) [back](#)

The dipole is defined using a small set of parameters. These include:

<b>CROSS NOCROSS</b>	include (default) or do not include a 3D cross near the tail of the arrow.
<b>DELETE</b>	Deletes the specified dipole if an identifier is given or all dipoles if no identifier is given; not used with any other parameters.
<b>WIDTH x.xx</b>	The width of the dipole in Angstroms. The default value is 0.005 Angstroms.
<b>ON/OFF</b>	Turns on or off the specified dipole or all drawn objects if no identifier is given; not used with any other parameters.
<b>OFFSET x.xx OFFSET n</b>	Dipoles are by default centered between the two endpoints. The OFFSET value sets the offset of the dipole from this position along the axis of its endpoints. In Angstroms if a decimal number is given; in percent of the distance between the two endpoints if an integer is used.
<b>OFFSETSIDE x.xx</b>	The offset of the dipole in Angstroms perpendicular to the axis of its endpoints. The default value is 0.4 Angstroms for atom-based dipoles and 0 for the molecular dipole.
<b>VALUE x.xxx</b>	A decimal number indicates the value of the dipole. Overall scaling is accomplished either by setting this number or using <a href="#">set dipoleScale</a> . The VALUE keyword is optional.

[positions] [back](#)

The positions of the endpoints of the dipole are set either using embedded atom expressions in parentheses, such as (atomno=1), or using a specific point in molecular space, {x y z}, either as a cartesian coordinate or a [fractional unit cell coordinate](#). If two atoms are designated, then the dipole becomes a member of the "bonds" dipole collection and can be colored with that group. If a set of atoms is indicated, the geometric center is used. Thus, (\*) indicates the geometric center of the molecule.

Examples:

See [examples-11/dipole.htm](#)



[↑top](#) [🔍search](#) [📘index](#)

## display

The opposite of [hide](#). Displays atoms and associated structures (bonds, halos, stars, cartoons, etc.) and hides all others. **Display** is similar to [restrict](#) in its action, but it is far more flexible. Atoms can be added to the displayed set using **display displayed or ...** or removed from the hidden set using **display displayed and not ...**. Unlike [restrict](#), the **display** command does not delete the hidden shapes. Thus, after **restrict none** all cartoons, traces, spacefill spheres, and bond sticks are effectively deleted. In contrast, **display none** is easily reversed using **display all**.

display [\[atom-expression\]](#)

where

**[atom-expression]** is any [expression](#) that evaluates to a set of atoms

Examples: [in new window using 1blu.pdb](#)



```
display protein
display not solvent
display within(3.0,[FS4]102)
```

See also:

[hide](#) [restrict](#) [select](#) [subset](#)

[↑top](#) [🔍search](#) [📘index](#)

## dots

Turns a dotted surface around the currently selected atoms. See also the [geoSurface](#) command and the settings [set dotsSelectedOnly](#) and [set dotSurface](#).

dots ON/OFF {default: ON}  
dots ONLY

Turns dot rendering on and all other rendering off.

dots VANDERWAALS

Draws dots at the van der Waals radius for the selected atoms. See [radli.xls](#).

dots IONIC

Draws dots at the (nominal) ionic radius for the selected atoms if the atom's formal charge has been read and is nonzero; uses the nominal covalent bonding radius otherwise ([radli.xls](#)).

dots n#%

Draws dots at the indicated percent of the van der Waals radius for each atom (maximum value 1000%).

dots (decimal)

Draws dots at the indicated radius in Angstroms for each atom (maximum value 10.0 Angstroms). Starting with Jmol 12.0, a negative number also implies **ONLY**.

dots +(decimal)

Draws dots at the indicated distance in Angstroms beyond the van der Waals radius for each atom (maximum value 10.0 Angstroms). The "+" sign is required.

dots ADPMIN n%

Draws dots at the radius corresponding to the minimum anisotropic displacement parameter for the selected atoms factored by the given percentage. See also [ellipsoid](#).

dots ADPMAX n%

Draws dots at the radius corresponding to the maximum anisotropic displacement parameter for the selected atoms factored by the given percentage. See also [ellipsoid](#).

See also:

[backbone](#) [background](#) [cartoon](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

[↑top](#) [🔍search](#) [📘index](#)

## draw

(v. 11.6 Jmol 11.6 adds **ARROW ARC, ARC, CIRCLE, and ID options, allows for mixed-type point sets, and special options Ramachandran, Quaternion, and PointGroup**)

ID [object id]  
[modifying parameters]  
[positions]  
[display options]

The **draw** command allows for the insertion of points, lines, and planes to a model. The general syntax of the **draw** command is as follows:

**draw [objectID] [modifying parameters] [positions]**

ID [object id] [back](#)

The optional identifier such as "line1" or "plane2" that can be referred to in later scripts as \$line1 or \$plane2. These words are arbitrary and, starting in Jmol 11.6, if preceded by the optional keyword ID may be any string. Starting with Jmol 11.6, specifically for the options ON, OFF, and DELETE, the id may include a wildcard character (asterisk). Thus, **draw v\* off** turns off all drawn objects having an ID starting with the letter "v".

[modifying parameters] [back](#)

Several options allow for a wide variety of simple structures to be drawn. These include:

"hoverText"	text that will appear after issuing <b>set drawHover TRUE</b> (Jmol 11.2)
ARROW	Draws a straight (two-point) or curve (more than two-point) arrow
ARC {pt1} {pt2} {ptRef} {nDegreesOffset theta fractionalOffset} SCALE scale DIAMETER diameter	Draws a theta-degree arc in a plane perpendicular to the line pt1--pt2 starting at nDegreesOffset degrees rotation from reference point {ptRef} at a point fractionalOffset from pt1 to pt2. The SCALE parameter is used to set the diameter of the overall circle containing the arc; the DIAMETER parameter sets the diameter of the curved arc line itself. <b>ARROW ARC</b> adds an arrow head.
ARC {pt1} {plane} {ptRef} {nDegreesOffset theta fractionalOffset}	As above, but uses the plane as a reference to define a perpendicular axis. (Jmol 11.8)

<b>CIRCLE {pt1} {pt2} SCALE scale DIAMETER diameter</b>	Draws a circle with center at pt1 (which may be an atom expression) in the plane perpendicular to the line between pt1 and pt2. If pt2 is not specified, the circle appears in 2D and remains in the plane of the screen when the model is manipulated. Together, the SCALE and DIAMETER parameters set the overall size of the circle. If no DIAMETER is given and an atom expression is given for pt1, the default diameter is one that includes those atoms; otherwise the default diameter is 1.0 Angstrom. The circle will be filled to form a solid disk unless the MESH NOFILL option is given.
<b>CIRCLE {pt1} {plane} SCALE scale DIAMETER diameter</b>	Draws a circle around pt1 in the indicated plane.
<b>COLOR (color)</b>	Sets the color of the drawn object at the time it is created. (The <a href="#">color</a> command can be used retroactively as well.)
<b>CROSSED</b>	Two lines (already drawn objects) specified next are crossed; switch the order of vertices for defining a plane.
<b>CURVE</b>	Draws a smooth curve through the given positions.
<b>CYLINDER</b>	Introduced in Jmol 11.6, draw CYLINDER creates a cylinder of the designated diameter. End caps can be set to closed/flat (FILL, the default) or open (MESH NOFILL).
<b>DELETE</b>	Deletes the object if an identifier is given or all drawn objects if none is given; not used with any other parameters.
<b>DIAMETER n</b>	Sets the number of pixels for the diameter of points, lines, arrows, and curves. Note that this means that zooming of the model does not change the width of these objects. Jmol 11.4 allows <b>n</b> to be a decimal value <b>x.x</b> , same as <b>WIDTH x.x</b> , which does scale.
<b>FIXED/MODELBASED</b>	Sets whether the surface generated is to be associated with the fixed window -- and thus appear with all frames/models -- or is to be associated with the currently displayed model (the default).
<b>ON/OFF</b>	Turns on or off the identified object or all drawn objects if no identifier is given; not used with any other parameters.
<b>PERP PERPENDICULAR</b>	Draw this object perpendicular to the next indicated object.
<b>PLANE</b>	Create a four-vertex quadrilateral even if only three points are given.
<b>REVERSE</b>	Reverse the order of vertices used if the next object listed is a line.
<b>ROTATE45</b>	Rotate a perpendicular plane to a line by 45 degrees.
<b>LENGTH (decimal)</b>	The length for a line/axis in Angstroms. The keyword LENGTH is optional.
<b>OFFSET {x y z}</b>	offsets the object by the given x, y, and z distances.
<b>SCALE (decimal) SCALE (integer)</b>	SCALE with a decimal value indicates a scaling factor for the drawn object. For example, <b>draw SCALE 1.5 (atomno=1) (atomno=2)</b> draws a line with length 1.5 times the distance from atom 1 to atom 2. The line is centered on the two atoms. The keyword SCALE is required in this case. Note that a draw command can consist of just an identifier and a scale. Thus, if \$line1 is already defined, <b>draw line1 SCALE 1.3</b> will rescale that line to 130% of the distance between its defining points. SCALE with an integer number indicates a percent scale. The keyword SCALE is optional in this case.
<b>VECTOR</b>	This option is similar to ARROW, and accepts two points. The first point is the origin; the second is of the form {dx,dy,dz}, indicating a vector to the next point rather than the point itself. [Jmol 11.6]
<b>VERTICES</b>	Generally the geometric center of an atom expression or drawn object is used for positioning. Added just before the atom set or object name reference, VERTICES indicates to use all vertices, not just the center point of the atoms in the expression or the points in the object.
<b>WIDTH x.x</b>	Sets the diameter of points, lines, arrows, and curves to a given width in Angstroms. Objects drawn will be scaled based on perspective and zoom setting (Jmol 11.4).
<b>"text"</b>	Starting with Jmol 11.2, a simple text label can accompany drawn objects. The text appears near the first point. Starting with Jmol 11.6, text starting with ">" will be associated with the last point instead of the first. The ">" character will be stripped before the text is displayed.

**[positions]** [back](#)

Positions define position of the point, the endpoints of the line/axis, or the corners of a plane. Positions can be indicated in any combination of any of the following four ways. Prior to Jmol 11.6, mixed types were grouped in the order shown here prior to processing; with Jmol 11.6, points are processed in a more intuitive way, in the order given on the command line.

{x, y, z}	a model-frame cartesian coordinate, in braces,
{x, y, z'}	for crystal structures, a unit-cell <a href="#">fractional coordinate</a> , in braces,
Subject	a previously defined drawing object such as \$line1 or \$plane2, preceded by "\$".
(atom expression)	an atom expression, in parentheses.

@{ {atomExpression} _split{}}	atom expressions split based on model index (Jmol 11.4).
-------------------------------	--

In addition, starting with Jmol 11.6, if two parameters are given, where the first evaluates to a point and the second is a four-vector (a b c d), a line is drawn from the point to the nearest point on the plane defined by **ax + by + cz + d = 0**. (Note that because quaternions and axisAngle are stored in quaternion format, which is internally the same as that used for planes, if a quaternion or axisAngle is used in this context, the line will be along the axis of rotation represented by the quaternion or axisAngle **q** to a point on a plane defined by **d = q.w = cos(theta/2)**, where **theta** is the rotation angle).

**[display options]** [back](#)

Display options for DRAW are indicated in the following table. These may be given at the end of the definition or in a later command having just these keywords and the identifier (or "ALL") of the desired draw object.

<b>FILL/NOFILL</b>	Display the drawn object as a smoothly filled surface.
<b>FRONTONLY/NOTFRONTONLY</b>	Display only the front half of the surface. This can be useful when the options <b>mesh nofill</b> are used.
<b>FRONTLIT /BACKLIT /FULLYLIT</b>	In some cases the object may appear flat. Using BACKLIT will fix this; FULLYLIT makes exterior and interior surfaces bright; FRONTLIT is the default setting.
<b>MESH/NOMESH</b>	Display a mesh of lines intersecting at the vertexes used to construct the object. For cylinders, the combination <b>MESH NOFILL</b> creates a cylinder with no end caps.
<b>ON/OFF</b>	Turn the object ON or OFF, but do not delete it.
<b>OPAQUE/TRANSLUCENT n</b>	Display the object as an opaque or translucent object. Several translucent options are available; see the <a href="#">color</a> command.

draw BOUNDBOX

Draws the currently defined [boundbox](#). Note that by default this is a solid. To show just edges, use options **MESH NOFILL**.

draw DELETE

Deletes all draw objects

draw FRAME {atom expression} {quaternion}

Draws a frame (an x,y,z axis set) at the given center with the given quaternion orientation. Quaternions are expressed using the [quaternion\(\)](#) function within a math [@{<sup>B</sup>}](#) wrapper. For example, **draw ID "q1" frame {0 0 0} @{quaternion(1,0,1,0)}** draws a frame at the origin that has been rotated 90 degrees relative to the Y axis. (Jmol automatically normalizes the quaternion to q0=0.70710677, q1=0, q2=0.70710677, q4=0, which represents a 90-degree rotation about the Y axis.)

draw HELIX AXIS

Draws a vector representing the local helical axis for the selected amino acid or nucleic acid residue, connecting it to the previous residue in its chain. The length of the arrow is the vertical height per residue. (Jmol 11.8)

draw INTERSECTION boundBox (plane expression)

Draws the portion of a plane that intersects the current [boundbox](#) based on a [plane expression](#).

draw INTERSECTION UnitCell (plane expression)

Draws the portion of a plane that intersects the current [unit cell](#) based on a [plane expression](#).

draw LIST

Lists all draw objects.

draw POINTGROUP [parameters]

Calculates and draws point group symmetry planes and axes for a symmetrical or nearly symmetrical molecule. As for [calculate pointgroup](#), the calculation is carried out only on the currently selected atoms and is limited to at most 100 selected atoms. Parameters include specification of a subset to draw (Cn, C2, C3, C4, C5, C6, C8, Sn, S3, S4, S5, S6, S8, S10, S12, Cs, Ci) optionally followed by an index (for example, **draw POINTGROUP C3 2** draws the second C3 axis only). A second option, SCALE x, allows adjusting the scale of the drawn planes and axes. The default scale of 1 puts the edge of planes directly through the outermost atoms. This command automatically sets **perspectiveMode OFF** so as to properly represent the planes and axes.

draw POLYGON [polygon definition]

The POLYGON option allows the ability to draw polygons based on a set of vertices and a set of faces. (Jmol 12.0) This capability allows drawing any number of flat triangular (not quadrilateral, but read on...) faces with or without edges around each face. The description is somewhat like that for PMESH files and involves (a) giving the number of vertices, (b) listing those vertices, (c) giving the number of faces, and (d) listing the faces with a special array syntax. Each face is described as an array indicating the three (0-based) vertex indices followed by a number from 0 to 7 indicating which edges to show a border on when the **mesh** option is given:

0	no edge
1	edge between first and second vertex
2	edge between second and third vertex

4	edge between third and first vertex
3, 5, 6, 7	combinations of the above.

For example: **draw POLYGON 4 {0 0 0} {1 1 1} {1 2 1} {0 5 0} 2 [0 1 2 6] [0 3 2 6] mesh nofill.**

draw QUATERNION [parameters]

Draws vectors for the previously selected residues representing the quaternion frame and rotational axis for each residue. The parameters for this command are the same as for [plot quaternion](#) (Jmol 12.0) or the [quaternion](#) command (Jmol 11.x). Vectors are named based on the axis (x, y, z, q), residue number, and chain. (Jmol 11.6)

draw RAMACHANDRAN

Draws curved arrows marked with PHI and PSI angles in planes perpendicular to the N-CA and CA-C bonds of the selected amino acids, respectively.

draw SYMOP {atom expression} {atom expression}

Draws the symmetry relations between any two atoms or groups or any two positions in space. For example, **draw SYMOP {molecule=1} {molecule=2}.**

draw SYMOP (integer) {atom expression} {atom expression}

Draws the symmetry relation associated with the specified symmetry operator between any two atoms or groups or any two positions in space. For example, **draw SYMOP {molecule=1} {molecule=2}.**

draw SYMOP [matrix]

Draws the symmetry operation associated with the given 4x4 matrix, which most likely comes from a variable, for example here the symmetry operation that is the product of two other symmetry operations: **draw SYMOP @{symop(11)\*symop(14)}.**

draw UNITCELL

Draws the currently defined [unitcell](#). Note that by default this is a solid. To show just edges, use options **MESH NOFILL**.

draw SYMOP [n or "x,y,z"] {atom expression}

Draws a graphic representation of a crystallographic space group symmetry operation. Operations include simple rotations, screw rotations, rotation-inversions, mirror planes, and glide planes. Either a number (for one of the model's symmetry operations) or a string indicating a Jones-Faithful operation may be used. The position may be an atom expression may be a point. If a point, it can be expressed either as a cartesian coordinate or a fractional coordinate (using a "/" in at least one position -- for example, { 1/2 0 0}). The ID of the draw command is prepended to the drawn object IDs. For example, **draw ID "s1" SYMOP "x,1/2-y,z" {1/2 1/2 1/2}.** See also the [Jmol Crystal Symmetry Explorer](#).

Examples:

See [examples-11/draw.htm](#)

See also:

[\[plane expressions\]](#) [\[isosurface lcaoCartoon mo pmesh polyhedra set \(misc\) show write \(model\) write \(object\) undefined\]](#)

[top](#) [search](#) [index](#)

## echo

In Jmol models can be annotated in one of three ways. Text can be associated with a specific atom using a [label](#), text can appear when the user hovers the mouse over an atom or other user-defined point in space for a designated period of time ([hover](#)), and text can be placed at a specified position on the window (2D **echo**) or at a point in space (3D **echo**). In addition, the text is echoed in the Java console, the Jmol [console](#), and the [set messageCallback](#) or [set echoCallback](#) function, if defined. Multi-line text can be generated using a vertical bar as a line separator.

See also the [message](#) command for variable-displaying capabilities that send information to the consoles and callback functions without displaying text. Starting with Jmol 11.6, one can also place JPEG, PNG, or GIF images at either a 2D screen position or a 3D molecular position. See [set echo](#) for details.

echo (string)

Examples: [in new window using caffeine.xyz](#)

```
set echo top left;font echo 20 serif bolditalic;color echo green
echo "I am green top left|20 serif bolditalic"
set echo myecho 350 150
echo this is|myecho;set echo myecho center
echo this|s|at|est
```

```
set echo myecho right
set echo myecho left
set echo top center
set echo top right
set echo top 70 320
```

See also:





[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [case default font for hover if label message reset set set \(highlights\) set \(labels\) set echo switch while](#)



[top](#) [search](#) [index](#)

## ellipsoid or ellipsoids

The **ellipsoid** command, introduced in Jmol 11.6, displays anisotropic displacement parameters (thermal ellipsoids) based on crystallographic Uij data in CIF or PDB files. The resulting ellipsoids have three mutually perpendicular axes that are not necessarily the same length. Overall ellipsoid size can be set to a percentage value (default 50%). Rendering styles are set globally using the **set** command and one or more of the settings described below. The diameter of the axis and arc lines can be set using **set ellipsoidAxisDiameter x.x** where x.x is a distance in Angstroms (default 0.02). Note that the Uij data when read from CIF files are also stored in the .temperature value for atoms as 100 times the geometric mean of the diagonal U-factor parameters, 100 \* (U11 \* U22 \* U33)<sup>0.3333</sup>.

User-defined ellipsoids can also be created by specifying the keyword ID followed by a user-named ID. These parameters may be indicated in any order, but the ellipsoid is not defined until the AXES parameter is specified.

<b>set ellipsoidAxes</b>	This option may be combined with any other option. When combined with balls, the color of the axes are set to white or black in order to contrast with the background.	
<b>set ellipsoidArcs</b>	When combined with balls, the color of the arcs are set to white or black in order to contrast with the background. This option is ignored when dots are displayed.	
<b>set ellipsoidBall</b>	This option is the default option and may be combined with ellipsoidAxes, ellipsoidArcs, or ellipsoidFill (which displays a cut-out octant).	
<b>set ellipsoidFill</b>		

	This option affects the rendering of arcs and balls. In the case of arcs, it fills in the arcs to form a set of three elliptical planes; in the case of balls, it provides a cut-out octant. When combined with <a href="#">spacefill ADPMIN n%</a> , where n is the current percentage size of the ellipsoid, the combination of fill and arcs provides a strikingly unique rendering of the anisotropic displacement parameters for an atom.	
<b>set ellipsoidDots</b>	This option renders a random set of dots defining the ellipsoid that sparkle as the model is manipulated. The number of dots can be set using <b>set ellipsoidDotCount n</b> where n is an integer (default 200). This option is ignored when ellipsoid balls are displayed.	

ellipsoid ON/OFF {default: ON}  
Turns ellipsoids on or off for the currently selected atoms.

ellipsoid nm%  
Sets the size of the ellipsoids for the currently selected atoms.

ellipsoid ID [object id] ON  
Turns this ellipsoid on.

ellipsoid ID [object id] OFF  
Turns this ellipsoid off.

ellipsoid ID [object id] AXES {ax ay az} {bx by bz} {cx cy cz}  
Sets the three perpendicular axes for the ellipsoid. These axes should be perpendicular. If they are not, the ellipsoid may not be rendered, or its shape will be unpredictable. The ellipsoid is not displayed until this parameter is set.

ellipsoid ID [object id] CENTER {x y z}  
Sets the center for this ellipsoid the specified point (which may be fractional).

ellipsoid ID [object id] CENTER { atom expression }  
Sets the center for this ellipsoid to the center of the specified atoms.

ellipsoid ID [object id] CENTER \$object  
Sets the center for this ellipsoid to the point specified by the object name (draw or ellipsoid)

ellipsoid ID [object id] COLOR [color parameters]  
Sets the color of the ellipsoid using parameters described in the [color](#) command.

ellipsoid ID [object id] DELETE  
Deletes the specified ellipsoid.

ellipsoid ID [object id] SCALE (decimal)  
Sets the scale of the ellipsoid relative to its axes lengths.

See also:  
[backbone](#) [background](#) [cartoon](#) [dots](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

## else

See [if](#).

See also:  
[break](#) [case](#) [catch](#) [continue](#) [default](#) [elseif](#) [for](#) [goto](#) [if](#) [return](#) [switch](#) [try](#) [var](#) [while](#)

[top](#) [search](#) [index](#)

[top](#) [search](#) [index](#)

## elseif

See [if](#).

See also:  
[break](#) [case](#) [catch](#) [continue](#) [default](#) [else](#) [for](#) [goto](#) [if](#) [return](#) [switch](#) [try](#) [var](#) [while](#)

[top](#) [search](#) [index](#)

## exit

When the [set scriptQueue](#) is turned on, each script waits for the previous to complete. Either [quit](#) or [exit](#) at the very beginning of a script command halts any previous still-running script. Processing then continues with the second command on the line. Anywhere else in the command, [quit](#) and [exit](#) abort that script only. In addition, [exit](#) clears the script queue of any remaining scripts, thus stopping all script processing.

See also:  
[delay](#) [goto](#) [loop](#) [pause](#) [quit](#) [resume](#) [set \(files and scripts\)](#) [step](#)

[top](#) [search](#) [index](#)

## fix

The [fix](#) command takes argument like display or select. But ensures no atoms of this set will be moved or dragged anywhere accidentally.

[fix](#) [\[atom-expression\]](#)

where  
[\[atom-expression\]](#) is any [expression](#) that evaluates to a set of atoms

[top](#) [search](#) [index](#)

## font

Sets font size, face (serif, sansSerif), and style (plain, italic, bold, bolditalic) in labels and other text-bearing elements. If only a font size is given and the object is a label, then only the size of the font is changed, not the font face or style. For all other objects, if only a font size is given, the font face and style are changed to sansSerif. In Jmol 11.6, for **font ECHO** you can apply a scaling factor that allows the font to scale with zoom. This scaling factor, in "pixels per micron", determines the absolute size of the font relative to a "standard" window size. (Scaled [echo](#) fonts can also be created using **set fontscaling TRUE**, then defining the echo text. The scaling factor also applies to images added with the [set echo](#) command.

[font](#) [\[object-with-text\]](#) [\[font-size\]](#) [\[font-face\]](#) {default: SansSerif} [\[font-style\]](#) {default: Plain} [\[scaling factor\]](#)

where  
[\[object-with-text\]](#) is AXES, ECHO, HOVER, LABEL, or MEASURE  
[\[font-size\]](#) is approximately the same as Rasmol -- (integer, 6 to 63)  
[\[font-face\]](#) is SERIF, SANSSERIF, or MONOSPACED  
[\[font-style\]](#) is PLAIN, BOLD, ITALIC, or BOLDITALIC

See also:  
[echo](#) [hover](#) [label](#) [set \(highlights\)](#) [set \(labels\)](#) [set echo](#)

[top](#) [search](#) [index](#)

## FOR

```
for (var i = 1; i < n i++) { ... }  
in-line FOR
```

Jmol supports both the standard **for** statement as well as a specialized in-line FOR construct that provides a powerful way to work with atom data.

**for (var i = 1; i < n i++) { ... } [back](#)**

**for** takes three arguments in parentheses and separated by a semicolon. The loop may be exited using [break](#) and truncated using [continue](#). For example, **for(var i=1; i <= 10; i++){...}** loops through a block of script ten times, incrementing the variable **i** by one each time. Any variables created with the **VAR** keyword are local to the **for** block.

**in-line FOR [back](#)**

In Jmol 11.8 you can create lists of values using a form of inline FOR function. The syntax is simply **xxx = for(dummyVariable;(atom expression);math expression)**. For example,

```
aniso = for(x;{*};x.adpmax - x.adpmin)
```

creates an array containing differences between the maximum and minimum anisotropic density parameters for a set of atoms. Arrays such as these can be used to assign properties to atoms or to color them. So, for example, we might want to set the radius of atoms based on temperature, but not the exact value of the temperature:

```
{*}.radius = for(x;{*};x.temperature/20)
```

Combined with the inline IF function, the inline FOR can be quite useful. For example:

```
{*}.color = for(x;{*};if(x.temperature > 10;"red";if(x.temperature < 2;"blue";"white")))
```

See also [functions](#) for how to define a function that operates on atoms one at a time with the syntax **{\*}.myFunction(a, b)**.

### Note:

The FOR command does not require **@{ ... }** around Jmol math expressions.

See also:

[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [break](#) [case](#) [catch](#) [continue](#) [default](#) [echo](#) [else](#) [elseif](#) [goto](#) [if](#) [message](#) [reset](#) [return](#) [set](#) [switch](#) [try](#) [var](#) [while](#)

[↑top](#) [🔍search](#) [📖index](#)

## frame or frames

Sets the current animation frame. Numbers refer to the physical position of the model in the file (1 being the first). Same as the [animation frame](#) command. See also [model](#). Note that you can show specific pairs or sets of frames or models by using **frame all** followed by **display (\*n,\*m,\*p)**, where **n**, **m**, and **p** are frame numbers. See also [set backgroundModel](#). For the applet, if [AnimFrameCallback](#) is enabled, a message indicating the frame change is sent to the associated JavaScript function. The simple command **frame** has no observable effect but forces an `animFrameCallback` message to be sent, which also serves to update the pop-up context menu. This is sometimes useful if scripting has changed something in the structure such as the addition of vibration vectors that would alter menu options (Jmol 11.8).

**frame (integer >= 1)**

Go to a specific model in the case of loading a single file with multiple models. In the case of the loading of a single PDB file containing MODEL records, the integer used here corresponds to the number in that record. In all other situations, the number used here is the sequential index of the model in the file, starting with 1. If more than one file is loaded, the number indicates the file, and all models in that file are overlaid.

**frame (decimal)**

Go to a specific model in a specific file when one or more files are loaded. The format for specifying which model to go to is the same as for [select](#) or [display](#): `file.model`. For example, **frame 2.3** goes to the third model in the second file listed in the most recent [load](#) command. Note that atoms in models chosen with the **frame** command must also be in the current [display](#)

set in order to be visible. So, for example, **display 2.1;frame 2.2** will display nothing; **display connected(hbond);frame 2.2** will display only the hydrogen-bonded atoms in model 2.2.

**frame (decimal) - (decimal)**

Sets the animation range and displays a range of models, possibly spanning multiple files. The hyphen is optional. If the hyphen is present but the second model number is missing, then all models from the designated model forward are assumed.

**frame 0**

Overlay all frames of the current frame range set. Note that this may not be all the models if the frame range has been set to a subset of the models or if multiple files are loaded and only models within one file have been specified with a previous **frame** command.

**frame 0.0**

Same as **frame ALL**.

**frame ALIGN { atom expression }**

(Jmol 11.8) Provides a way to align structures across a set of frames. This is important for certain animations. The atom expression is evaluated per frame, and the resultant point is aligned in each case.

**frame ALL**

Resets the frame range to all models and overlays them.

**frame LAST**

Go to the last frame in the frame range set.

**frame NEXT**

Go to next frame in the frame range set.

**frame PAUSE**

Pause animation at the current frame.

**frame PLAY (starting frame)**

Start playing at the specified frame number (current frame if number is omitted). Direction, speed of play, and mode of animation (ONCE, LOOP, or PALINDROME) are set using [animation mode](#).

**frame PLAYREV (starting frame)**

Start playing at the specified frame number (current frame if number is omitted), reversing the direction.

**frame PREVIOUS**

Go to previous frame in the current frame set.

**frame RANGE (starting frame) (ending frame)**

Sets the range of frames to play as an animation and sets the current frame to the first number given. If the starting frame number is larger than the ending frame number, then play is in reverse. If only one number is given, then the range is set from that frame through the last frame in the file. If both numbers are omitted, then the range is set to include all frames in the file.

**frame RESUME**

Resume playing from the current frame. (Same as **PLAY**.)

**frame REWIND**

Return to the first frame in the frame range set.

**frame TITLE "title"**

Sets a title for the frame (or, in Jmol 12.0, all frames currently visible), which appears in the bottom left corner of the applet or application. Starting with Jmol 12.0, if the title includes an expression such as `"@({_modelName})"`, then that expression is evaluated whenever the model is rendered (for example, when the frame is changed). To set all titles at once, first make all frames visible. with **frame all**, then issue the **frame title** command.

Examples: [in new window using cyclohexane\\_movie.xyz](#)

```
model 1  
model NEXT  
model PREVIOUS  
model 0;select *;wireframe 0.1;spacefill 0.2  
anim on  
model 0;select *;wireframe off;spacefill off;  
select 1.1 # in Jmol10 use */1  
wireframe 0.1;spacefill 0.2;color atoms red;  
select 1.35;wireframe 0.1;spacefill 0.2;color atoms blue
```

See [animation.htm](#)

See also:

[animation](#) [invertSelected](#) [model](#) [move](#) [moveto](#) [rotateSelected](#) [set \(misc\)](#) [spin](#) [translate](#) [translateSelected](#) [zoom](#) [zoomto](#)



## frank

Determines whether or not "Jmol" is indicated in the bottom right corner of the window.

frank ON/OFF  
Turning the frank off is disabled for the signed applet running on a web server.

[↑top](#) [🔍search](#) [📖index](#)

## geoSurface

Turns a crude geodesic molecular surface on or off around the currently selected atoms. If a decimal with an explicit "+" sign is given, or [set solvent ON](#)) is in effect, the resultant surface is a crude solvent-accessible surface. This command has the same syntax as the [dots](#) command. To color the surface, use [color geosurface](#). For a smoother surface, use [isosurface SASURFACE 1.2](#).

geoSurface ON/OFF {default: ON}  
geoSurface ONLY  
Turns geosurface rendering on and all other rendering off.  
geoSurface VANDERWAALS  
Draws a geodesic surface at the van der Waals radius for the selected atoms. See [radii.xls](#).  
geoSurface IONIC  
Draws a geodesic surface at the (nominal) ionic radius for the selected atoms if the atom's charge has been read and is nonzero; uses the nominal covalent bonding radius otherwise ([radii.xls](#)).  
geoSurface (integer)  
Draws a geodesic surface at the indicated percent of the van der Waals radius for each atom (maximum value 1000%).  
geoSurface (decimal)  
Draws a geodesic surface at the indicated radius in Angstroms for each atom (maximum value 10.0 Angstroms).  
geoSurface +(decimal)  
Draws a geodesic surface at the indicated distance in Angstroms beyond the van der Waals radius for each atom (maximum value 10.0 Angstroms). The "+" sign is required. This surface approximates the solvent-accessible surface with the indicated solvent probe radius. Typically this number is +1.2 or +1.4. This command overrides the [set solvent/set radius](#) method of defining the solvent-accessible surface.

See also:  
[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

[↑top](#) [🔍search](#) [📖index](#)

## getProperty

The **getProperty** command sends information to the [Jmol console](#) or message callback function defined for a Jmol applet using the `jmolSetCallback("messageCallback", funcName)` function in Jmol.js or via the [set](#) command. Either a simple text string in the case of a file property or a valid JSON (JavaScript Object Notation) string in the case of a molecular property is returned. Used with `jmolScriptWait()`, the **getProperty** script command provides a powerful way to interact with the Jmol applet. Even simpler, though, is to use one of the Jmol.js built-in JavaScript commands `jmolGetPropertyAsString()`, `jmolGetPropertyAsJSON()`, `jmolGetPropertyAsJavaObject()`, or (most useful, probably) `jmolGetPropertyAsArray()`. For example,

```
var modelInfo = jmolGetPropertyAsArray("modelInfo")
alert(modelInfo.modelCount)
for (int i = 0; i < modelInfo.modelCount; i++)
  alert(modelInfo.models[i].name).
```

In addition, when using the Jmol application, see the note at [show](#) regarding setting the output to go directly into a file on your system. Starting with Jmol 11.3.45, this information is also available using the Jmol math [getProperty\(\)](#) function.

getProperty animationInfo

JSON structure describing the current state of animation. See [animationInfo.txt](#).  
getProperty appletInfo  
JSON structure describing the applet, including, for example, the applet version, compile date, Java version, and name of the applet object. See [appletInfo.txt](#).  
getProperty atomInfo (atom expression)  
JSON structure describing the atoms in the model. A second (optional) parameter specifies a subset of the atoms. The default is (visible). Parentheses are required. See [atomInfo.txt](#).  
getProperty atomList (atom expression)  
getProperty auxiliaryInfo  
JSON structure describing auxiliary information that is in the loaded file. This information is file-dependent and might include, for example, symmetry information, molecular orbital coefficients, dipole moments, partial charges, and/or vibrational modes. See [auxiliaryInfo.txt](#).  
getProperty bondInfo (atom expression)  
JSON structure describing the bonds in the model. A second (optional) parameter specifies a subset of the atoms that are involved in the bonds. The default is (visible). Parentheses are required. See [bondInfo.txt](#).  
getProperty boundBoxInfo  
A simple JSON array containing the coordinates of the center and corner of the volume containing the molecule. See [bondBoxInfo.txt](#).  
getProperty centerInfo  
A single JSON array giving the current center coordinate. See [centerInfo.txt](#).  
getProperty chainInfo (atom expression)  
JSON structure describing the chains in a biomodel (PDB or mmCIF, for example). Information for each residue of the chain is provided. A second (optional) parameter specifies a subset of the atoms. The default is (visible). Parentheses are required. See [chainInfo.txt](#).  
getProperty dataInfo type  
Returns an array having two elements. Generally the first element is the data label, and the second element is the **data** itself. **getProperty data TYPES** returns a first element that is the string "types" and a second element that is a comma-separated list of the available data types.  
getProperty extractModel (atom expression)  
The `extractModel` keyword delivers text in the form of a MOL file, allowing up to 999 atoms and 999 bonds to be "extracted" from the model as an independent structure.  
getProperty fileContents  
The contents of the currently loaded file.  
getProperty fileContents filepath  
The contents of ANY file on the web or, if operating locally, any file on the hard drive in the directory containing the JAR file or any directory below that.  
getProperty fileHeader  
The file header for the file. This will depend upon the file format; some file formats may not have file headers.  
getProperty fileName  
The file name of the currently loaded file.  
getProperty image  
A base-64 encoded JPEG suitable for writing by the applet to an image tag using `` (does not work in Microsoft Explorer).  
getProperty jmolStatus statusNameList  
JSON structure describing the current state of one or more StatusManager properties.  
getProperty jmolViewer  
getProperty measurementInfo  
JSON structure describing the currently-defined measurements for the model, including the atoms involved, the measurement type, and the value of the measurement in decimal and in string formats. See [measurementInfo.txt](#).  
getProperty messageQueue  
getProperty menu  
A tab-separated string defining the current Jmol menu, including what options are currently enabled. See, for example, [misc/menu.tab](#). See also [show MENU](#).  
getProperty minimizationInfo  
A summary of the minimization that was carried out. See [minimizationInfo.txt](#).  
getProperty modelInfo  
JSON structure describing each model in the loaded model collection. See [modelInfo.txt](#).  
getProperty moleculeInfo (atom expression)

JSON structure describing each molecule (covalent set of atoms) in the model, including the number of atoms, the number of elements, and the molecular formula. A second (optional) parameter specifies a subset of the atoms. The default is (visible). Parentheses are required. See [moleculeInfo.txt](#).

getProperty orientationInfo

JSON structure describing the moveTo command required to return to the currently displayed orientation. See [orientationInfo.txt](#).

getProperty polymerInfo (atom expression)

JSON structure similar to chainInfo describing the residues in a biomodel. A second (optional) parameter specifies a subset of the atoms. The default is (visible). Parentheses are required. See [polymerInfo.txt](#).

getProperty shapeInfo

JSON structure listing a small amount of information relating to shapes displayed with the model (molecular orbitals, isosurfaces, cartoons, rockents, dipoles, etc.) See [shapeInfo.txt](#).

getProperty stateInfo (atom expression)

getProperty transformInfo

JSON structure representing the current transformation matrix describing the current orientation of the model. See [transformInfo.txt](#).

See also:

[boundbox](#) [script](#) [set \(callback\)](#) [show](#)

[top](#) [search](#) [index](#)

## goto

Transfers script execution to the [message](#) command having the matching text. If an underscore is at the beginning of the text, then the text is not displayed, otherwise the message text is displayed in the console as usual for that command.

message \_test

...

...

goto \_test

The **goto** command is largely superseded in Jmol 11.4 by [if](#), [while](#), and [for](#). These alternative commands are recommended.

See also:

[break](#) [case](#) [catch](#) [continue](#) [default](#) [delay](#) [else](#) [elseif](#) [exit](#) [for](#) [if](#) [loop](#) [pause](#) [quit](#) [resume](#) [return](#) [step](#) [switch](#) [try](#) [var](#) [while](#)

[top](#) [search](#) [index](#)

## halos

Displays a translucent two-dimensional ring around an atom. Halos are similar to [stars](#), except halos may also be used for automatically displaying which atoms are currently selected. (This option is enabled using [selectionHalos](#).) The radius can be set using the same options as for [spacefill](#), but the actual radius of the halo will always be from 4 to 10 pixels larger than the nominal radius.

halos ONLY

Turns halo rendering on and all other rendering off.

halos ON/OFF{default: ON}

Turn halos on or off. The radius will change with the current spacefill setting.

halos reset

Resets the radius of the halo to the default [spacefill](#) radius.

[top](#) [search](#) [index](#)

## hbonds

(v. 12.0 -- full support for hydrogen bonds)

Hydrogen bonds can be turned on or off, given custom widths in Angstroms, or colored (see [color hbonds](#) and [set HBONDS](#) ). In addition, Jmol 12.0 allows calculation of actual hydrogen bonds (when hydrogen atoms are present in the selected set) or "pseudo" hydrogen bonds if hydrogen atoms are not present.

hbonds ON/OFF{default: ON}

hbonds [[width-in-angstroms](#)]

hbonds CALCULATE

Calculates hydrogen bonds involving atoms currently selected and displays them. See [calculate HBONDS](#) for details.

where

[[width-in-angstroms](#)] is a (decimal, <2.0)

See also:

[bondorder](#) [color \(bond object\)](#) [connect](#) [set \(bond styles\)](#) [set \(files and scripts\)](#) [ssbonds](#) [wireframe](#)

[top](#) [search](#) [index](#)

## help

Opens a new browser window to the interactive help page. See also [set helpPath](#). Currently for the applet only; the query will be searched as an exact, complete phrase (applet only).

help query

[top](#) [search](#) [index](#)

## hide

The opposite of [display](#). Hides atoms and associated structures (bonds, halos, stars, cartoons, etc.). **Hide** is similar to [select](#) or [restrict](#) in its syntax. Unlike [restrict](#), though, **hide** is completely reversible using [hide none](#). (**Restrict** acts by setting the "size" of the object to 0; **hide** leaves the size the same, but just hides the object until unhidden. Group-based structures such as [cartoons](#) and traces are hidden whenever their lead atom (the one that determines their position in space) is hidden. Hidden atoms can be selected with [select hidden](#) or deselected with [select not hidden](#). Atoms can be added to the hidden set using [hide hidden or ...](#) or removed from the hidden set using [hide hidden and not .....](#)

hide [[atom-expression](#)]

where

[[atom-expression](#)] is any [expression](#) that evaluates to a set of atoms

See also:

[display](#) [restrict](#) [select](#) [subset](#)

[top](#) [search](#) [index](#)

## history

The **history** command turns command history recording on and off. Each time it is invoked, the command history list is cleared. See also [set history](#) and [show history](#).

history ON/OFF{default: ON}

[top](#) [search](#) [index](#)

## hover

Turns on and off pop-up labels that appear when the user "hovers" the mouse over the atom or a point associated with a [draw](#) object. **hover off** disables the hover message; **hover on** enables it again. Any other parameter or string is used as the label, which can contain atom property fields such as %a or %U. To change the delay time prior to the label appearing, use **set hoverDelay x.x** where **x.x** is the delay time in seconds. Setting this time to 0.001 seconds effectively makes the hover label appear immediately. See also [label](#). Multiple lines can be indicated using | (vertical bar). In the Jmol applet version 11.0, even with **hover OFF**, the hover message can be sent to a JavaScript function on the applet's page using [set hoverCallback](#).

See also:  
[echo](#) [font](#) [label](#) [set \(highlights\)](#) [set \(labels\)](#) [set echo](#)

## IF

if/else/elseif  
in-line IF

Jmol scripting allows for both the standard **if/else/elseif** statement set as well as the standard Java/JavaScript-like inline (**ifThis ? thenThis : elseThis**) construct.

**if/else/elseif** [back](#)

The **if** statement takes a logical expression of variables and evaluates it either as TRUE or FALSE. Tests include a wide variety of flags that can be set using the [set](#) command as well as any [user-defined variables](#). The standard commands **elseif** (or **else if**) and **else** are supported. **If** blocks are defined by matched pairs of braces:

```
if (doSpacefill) {
  spacefill only
} elseif (doWireframe) {
  wireframe only
} else {
  wireframe 0.15;spacefill 20%
}
```

These constructs are compatible with [goto](#). However, using **goto** to jump into the middle of an **if**, **while**, or **for** block is considered bad form and is not recommended.

Note that with the Jmol application multiline scripts are not allowed on the script window command line. Either separate commands by semicolon and introduce them all on the same line, or store the script in a file and run it using the [script](#) command. This is not an issue with the applet.

**if/else/endif** syntax is also supported. Starting with Jmol 11.8, if/else/endif syntax can be entered on a single line without unnecessary semicolons:

```
if (this) print "this" else print "that"
if (this) { print "this" } else { print "that" }
if (this); print "this"; else; print "that"; endif;
```

are all identical.

**in-line IF** [back](#)

In Jmol 11.8 you can use a similar construct to Java's/JavaScript's (**ifTrue ? thisValue : orThisTrue ? thatValue : otherValue**). Jmol accepts this format and also allows for a syntax that uses semicolons:

```
xxx = if(ifTrue; thisValue; thatValue)
```

For example,

```
set echo top left;echo @({_modelnumber > 1.1 ? "previous" : "next"})
```

or, equivalently,

```
set echo top left;echo @({if(_modelnumber > 1.1;"previous" ; "next"})
```

[top](#) [search](#) [index](#)

. Used in conjunction with inline FOR, the inline IF provides a powerful and concise way to work with molecular data.

Note:

The IF command does not require @{ ... } around Jmol math expressions.

See also:

[Jmol command syntax](#) [Jmol math](#) [Jmol parameters](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [break](#) [case](#) [catch](#) [continue](#) [default](#) [echo](#) [else](#) [elseif](#) [for](#) [goto](#) [message](#) [reset](#) [return](#) [set](#) [switch](#) [try](#) [var](#) [while](#)

## initialize

The **initialize** command is a specialized script command only intended for use within a state script created using the [write state](#) or [save state](#) commands. It initializes variables and settings to original Jmol values but does not do a full reset to Jmol's initial state. If that capability is desired, one should save the initial state and then [restore](#) it.

See also:

[define](#) [load](#) [refresh](#) [reset](#) [restore](#) [save](#) [set \(files and scripts\)](#) [zap](#)

## invertSelected

(v. 11.2 -- new)

This command allows inversion of the selected atoms through a point or across a plane. With no parameters, the model is inverted through its geometric center.

invertSelected POINT point\_definition

Following the POINT keyword may be any expression that can be evaluated to give a point, including {x y z} (cartesian or fractional), a draw object identifier, \$xxx, or an atom expression in parentheses, which will be evaluated as an average position.

invertSelected PLANE plane\_express

inverts the selected atoms across a plane defined using any valid [plane expression](#).

invertSelected HKL {h k l}

Specifies that the coordinates are the Miller indices of the plane to be used for the inversion.

invertSelected STEREO {center} {atomsToInvert}

Carries out a rotation of 180 degrees around a center about a line connecting the center and the geometric center of any connected atoms not in the list to invert. The results is the standard organic chemist's "stereochemical inversion" at that point -- which does not invert stereochemistry along the branches. See also [set picking invertStereo](#)

See also:

[animation](#) [frame](#) [model](#) [move](#) [moveto](#) [rotateSelected](#) [set \(misc\)](#) [spin](#) [translate](#) [translateSelected](#) [zoom](#) [zoomto](#)

## isosurface

(v. 12.0 -- adds "=XXXX" for retrieving data directly from the Uppsala electron density server, LATTICE {{a b c}} for rendering multiple copies of an isosurface, and MEP functionType to support mapping of molecular lipophilic potential)

ID [object id]  
[construction/mapping parameters] -- molecular/solvent surfaces  
[construction/mapping parameters] -- molecular orbitals  
[construction/mapping parameters] -- general shapes  
[construction/mapping parameters] -- general file loading  
[surface object] -- molecular/solvent surfaces

[top](#) [search](#) [index](#)

[top](#) [search](#) [index](#)

[top](#) [search](#) [index](#)

```
[surface object] -- atomic and molecular orbitals
[surface object] -- general shapes
[surface object] -- file/data-derived isosurfaces
[additional mapping-only parameters]
MAP [color mapping dataset]
[color and contour options]
[display options]
```

Jmol can generate a large variety of objects using the method of isosurfaces. Many of these surfaces can be color-mapped. Two Jmol commands (**isosurface** and **mo**) render isosurfaces. The **isosurface** command itself provides ultimate control over these surface renderings. Before using the **isosurface** command, if you are interested in rendering molecular orbitals, you should first take a look at the **mo** command.

The general syntax of the **isosurface** command is as follows:



**isosurface** ID [object id] [construction/mapping parameters] [surface object] [additional mapping-only parameters] MAP [color mapping dataset] [display options] . While the id is optional, it is highly recommended. Starting with Jmol 11.6, specifically for the options ON, OFF, and DELETE, the id may include a wildcard character (asterisk). Thus, **isosurface pl\* on** turns on all isosurface objects having an ID starting with "pl".

Data for these surfaces can be in the form of Gaussian CUBE files, APBS OpenDX multigrid files (for molecular electrostatic potentials; starting in Jmol 11.2), Xplor electron density text file data (Jmol 11.4), PIt formatted electron density files (Jmol 11.4), PMESH files (see the **pmesh** command) files, or **JVXL (Jmol Voxel)** format files, molecular orbitals and their associated Gaussian or Slater basis functions from computational software packages, or a number of internally calculated surface types: Shroedinger hydrogen-like atomic orbitals, LCAO-"cartoon" orbitals, spheres, ellipsoids, tear-drop shaped lobes, user-defined functions f(x,y), and Jmol-calculated solvent surfaces (accessible or excluded). In addition, starting with Jmol 11.2, atom-based data from a variety of sources can be mapped onto an isosurface. If a CUBE or APBS or JVXL file is used as the source, it need not be the file that was loaded using the **load** command, which reads only the atom position, not the surface. A separate isosurface command is used to read the scalar field data and construct the isosurface. This **isosurface** represents the points in space where scalar values cross a specified "cutoff" value. Inside the surface, values are greater or equal to a specified positive cutoff or less than or equal to a specified negative cutoff. The default cutoff depends upon the type of object displayed, but for CUBE, APBS, or JVXL files it is 0.02. Note that positive and negative surfaces may be created separately, or, using the SIGN keyword, they can be generated together.

You can construct different isosurfaces with different shapes and sizes by reading the same CUBE or APBS file more than once with different parameters, or by reading different files or selecting different volumes in a given file or different surfaces contained in a given JVXL file. By naming these individual surfaces with unique identifiers you can control display settings and color for each of the surfaces independently.

Color mapping of one object onto another is as simple as listing both an object and a dataset within the same isosurface command. Several keywords affecting the mapping are allowed. The default color scheme uses a red-->orange-->yellow-->green-->blue rainbow, where red represents minimum values and blue represents maximum values, but several other schemes are available (see below). Starting with Jmol 11.8, mapped data can be expressed as a series of contour lines.

CUBE, APBS, and JVXL files may be gzip-compressed.

**isosurface** [object id] [construction/mapping parameters] [surface object] [additional mapping-only parameters] MAP [color mapping dataset] [display options]

ID [object id] [back](#)

The optional identifier allows you to refer back to this isosurface later for turning the surface on or off, deleting the surface, or changing display options. It must be either the first parameter or the second, just after DELETE. If the identifier is missing, behavior depends upon version. Starting with Jmol 11.6, the ID keyword is optional but recommended, because then one can use any name for the ID; otherwise the name must not be one of the many Jmol command or keyword names. Prior to Jmol 11.2, leaving off the ID when creating an isosurface created a new isosurface; starting with Jmol 11.2, leaving off the ID when creating an isosurface replaces the current isosurface with the new one.

[construction/mapping parameters] -- molecular/solvent surfaces [back](#)

<b>ADDHYDROGENS</b>	For a solvent or sasurface object, accounts for missing hydrogens on carbon atoms only just prior to generating the surface. A simple sp <sup>3</sup> model is used.
<b>IGNORE {atom expression}</b>	Specifies a set of atoms to completely ignore when generating a solvent-related surface ( <b>SOLVENT</b> or <b>SASURFACE</b> ). Typically these might be solvent molecules or atoms: <b>IGNORE {solvent}</b> .
<b>IONIC radius</b>	Atom radius relative to the ionic radius. Options include:

	<table> <tr> <td>x.x</td><td>specific absolute number of Angstroms for every atom</td></tr> <tr> <td>+/-x.x</td><td>offset greater(+) or less(-) than the ionic radius</td></tr> <tr> <td>nn%</td><td>percent of the ionic radius</td></tr> <tr> <td>+/-nn%</td><td>percent offset from the ionic radius</td></tr> </table>	x.x	specific absolute number of Angstroms for every atom	+/-x.x	offset greater(+) or less(-) than the ionic radius	nn%	percent of the ionic radius	+/-nn%	percent offset from the ionic radius
x.x	specific absolute number of Angstroms for every atom								
+/-x.x	offset greater(+) or less(-) than the ionic radius								
nn%	percent of the ionic radius								
+/-nn%	percent offset from the ionic radius								
<b>SELECT {atom_exp}</b>	Selects a specific subset of the atoms for this surface. Same as <b>select atom_exp; isosurface....</b> except the atoms are selected only within the isosurface command, not the overall script.								
<b>SET n</b>	If an isosurface is fragmented, this option allows visualization of only the set of triangles composing one specific surface fragment. (Jmol 11.8.RC3)								
<b>VDW radius</b>	Atom radius relative to the van der Waals radius. Options include: <table> <tr> <td>x.x</td><td>specific absolute number of Angstroms for every atom</td></tr> <tr> <td>+/-x.x</td><td>offset greater(+) or less(-) than the van der Waals radius</td></tr> <tr> <td>nn%</td><td>percent of the van der Waals radius</td></tr> <tr> <td>+/-nn%</td><td>percent offset from the van der Waals radius</td></tr> </table>	x.x	specific absolute number of Angstroms for every atom	+/-x.x	offset greater(+) or less(-) than the van der Waals radius	nn%	percent of the van der Waals radius	+/-nn%	percent offset from the van der Waals radius
x.x	specific absolute number of Angstroms for every atom								
+/-x.x	offset greater(+) or less(-) than the van der Waals radius								
nn%	percent of the van der Waals radius								
+/-nn%	percent offset from the van der Waals radius								
<b>WITHIN x.xx {atomExpression or point}</b>	only creates a surface within the specified distance in Angstroms from the specified atoms or point.								

[construction/mapping parameters] -- molecular orbitals [back](#)

<b>IGNORE {atom expression}</b>	Specifies a set of atoms to completely ignore when generating a molecular orbital, thus showing only selected atomic contributions.
<b>SCALE x.xxx</b>	In the case of molecular orbitals, scales the volume around the orbital. It may be useful in cases where Jmol misjudges the full extent of an orbital, thus truncating it.
<b>SELECT {atom expression}</b>	Selects a specific subset of the atoms for this molecular orbital. (Essentially the opposite of IGNORE.)
<b>SIGN</b>	For molecular orbitals derived from cube files, indicates that the data have both positive and negative values and that they should both be displayed.
<b>SQUARED</b>	Data are to be squared prior to surface generation. (Jmol 11.4)

[construction/mapping parameters] -- general shapes [back](#)

<b>ANISOTROPY {a b c}</b>	Sets the anisotropic distortion of an object in the x, y, and z directions.
<b>CAP</b>	See <b>SLAB</b> . The distinction between SLAB and CAP is that SLAB leaves the isosurface open at the boundaries, whereas CAP closes it off.
<b>CENTER {x y z}</b>	Centers an atomic orbital, sphere, ellipsoid, or lobe at a specific point in molecular space. In the case of crystal structures, these may be <a href="#">fractional coordinates</a> .
<b>CENTER {atom_exp}</b>	Centers an atomic orbital, sphere, ellipsoid, or lobe on a certain atom or at the geometric center of this set of atoms.
<b>CENTER \$object</b>	Centers an atomic orbital, sphere, ellipsoid, or lobe on a certain drawn object such as a point, line, or plane.
<b>ECCENTRICITY {ex cy cz f_ab}</b>	Sets the eccentricity of a sphere, ellipsoid, atomic orbital, or lobe. The first three numbers define the "principal" axis; the fourth parameter defines the ratio of the other two perpendicular axes to this main axis.
<b>PHASE "type"</b>	Indicates that an orbital or other object is to be colored based on the position of the voxel in space. With an atomic orbital and no parameters, indicates regions of (+) and (-) orbital value with different colors. Valid types include "x", "y", "z", "xy", "xz", "yz", "z2", and "x2-y2".
<b>SCALE x.xxx</b>	In the case of objects for which eccentricity can apply (spheres, ellipsoids, atomic orbitals, and lobes), scales the object (default 1.0).
<b>SLAB [plane definition]</b> <b>SLAB BOUNDBOX</b> <b>SLAB UNITCELL</b>	[Jmol 12.0] clean slabbing of an isosurface based on a plane definition such as x=3 or {1 0 0 -3} or the currently defined unitcell or bbox. A negative sign prior to a plane definition indicates "the opposite-facing plane". The distinction between SLAB and CAP is that SLAB leaves the isosurface open at the boundaries, whereas CAP closes it off.

[construction/mapping parameters] -- general file loading [back](#)

<b>=XXXX</b>	(Jmol 12.0) adds the capability to automatically load electron density maps from the <a href="#">Uppsala Electron Density Server</a> into the application and signed applet. Supporting this are the global variables <a href="#">set edsUrlCutoff</a> and <a href="#">edsUrlFormat</a> , which set the method of getting cutoff and file from electron density server. The BOUNDBOX isosurface option is automatically applied.
<b>ANGSTROMS</b>	For a cube file or user-defined function f(x,y), indicates that the volumetric definitions are in Angstroms instead of Bohr (default).
<b>BLOCKDATA</b>	Indicates that data for surfaces in multiple-surface CUBE files are in sequential blocks rather than the Gaussian standard of being interspersed, where all data values for a coordinate point are listed together.
<b>BOUNDBOX</b>	Specifies that the surface to be generated must be within the currently defined <a href="#">boundbox</a> . (Jmol 12.0)
<b>COLOR DENSITY</b>	Allowing rendering of the actual grid of numbers (volume rendering) of the data rather than an actual isosurface. With <b>CUTOFF 0.0</b> , this setting delivers the entire set of data points. It is recommended that the BOUNDBOX parameter be used with a relatively small boundbox setting in order to not have an out-of-memory condition resulting from this option. For example: <b>boundbox scale 1.2 {tyr};isosurface boundbox cutoff 1.6 "3hyd_map.ccp4.gz" mesh nofill</b>
<b>CUTOFF x.xxx</b>	Sets the cutoff value defining an isosurface. Typically, smaller values correspond to a larger object. Cutoffs can be either positive or negative. In the case of a molecular orbital, a positive number indicates to use both positive and negative cutoffs. Adding an explicit "+" sign before the number indicates that only the positive part of the surface is desired. (See also SIGMA, below.)
<b>DEBUG</b>	Produces voluminous detail for program debugging.
<b>FIXED/MODELBASED</b>	Sets whether the surface generated is to be associated with the fixed window -- and thus appear with all frames/models -- or is to be associated with the currently displayed model (the default).
<b>FULLPLANE</b>	for PLANE objects, indicates that color mapping should be extended to complete the plane. (Jmol 12.0)
<b>GRIDPOINTS</b>	Adds the specific gridpoints used by the "marching cubes" algorithm for the calculation of the isosurface. Primarily for discussion and debugging of the isosurface algorithm.
<b>INSIDEOUT</b>	For certain datasets it sometimes happens that the surface rendered appears inside-out (dark on the outside and bright on the inside). If this is the case, add <b>INSIDEOUT</b> to the isosurface command prior to specification of the file to load. Jmol will reverse the sense of what is inside and what is outside the surface. This flag only affects rendering in Jmol, not export to PovRay. (Jmol 11.4)
<b>MODEL n</b>	Sets the identity of the model with which this isosurface is to be associated. (Defaults to the currently displayed model.)
<b>RESOLUTION x.x</b>	Sets the resolution of a constructed isosurface three-dimensional grid of "voxels", in points per Angstrom. Does not apply to CUBE or Jvxl files, which have a resolution defined by internal file parameters.
<b>SIGN c1 c2</b>	Indicates that cube data have both positive and negative values, and that they should both be displayed using the value given for CUTOFF and its negative. The two colors to use may be given optionally.
<b>SIGMA x.xx</b>	Sets the cutoff based on the root mean square deviation of the data. (Currently supported only for the MRC/CCP4 file format; Jmol 12.0.)
<b>SQUARED</b>	Data are to be squared prior to surface generation. (Jmol 11.4)
<b>WITHIN x.xx {atomExpression or point}</b>	only creates the portion of the surface within the specified distance in Angstroms of the specified atoms or point. (Jmol 12.0)

[surface object] -- molecular/solvent surfaces [back](#)

Several isosurface options relate specifically to molecular or solvent-related surfaces.

<b>CAVITY cr er</b>	(Jmol 11.2) Renders a depiction of a molecular cavity. The optional parameters cr and er determine the overall properties of the cavity. cr (cavity radius, default 1.2) sets the radius in Angstroms of the "probe" molecule that would fit into the cavity. er (envelope radius, default 10) sets the radius of the probe used to define the outer limits of the molecule. Smaller numbers for the cavity radius lead to more detailed cavities; smaller numbers for the envelope radius lead to cavities that are more internal and extend less toward the outer edge of the molecule. Qualifier <b>INTERIOR CAVITY</b> creates isosurfaces only for cavities that do not extend to the outer surface of the molecule. Qualifier <b>POCKET CAVITY</b> creates isosurfaces only for pockets that extend to the outer surface of the model, showing them more as troughs or open "pockets".
<b>MEP</b>	Depicts the molecular electrostatic potential, as calculated by SUM(q <sub>i</sub> /r <sub>i</sub> ), where q <sub>i</sub> is the partial charge on atom i (as found in the loaded model file) and r <sub>i</sub> is the distance from atom i to a particular point in space. The molecular electrostatic potential is not typically displayed itself. Rather, it is usually mapped onto a molecular surface. For example: <b>isosurface resolution 6 SOLVENT map MEP</b> produces a smooth surface at the van der Waals distance around the molecule colored by the molecular electrostatic potential.
<b>MOLECULAR</b>	Same as <b>SOLVENT 1.4</b>

<b>SASURFACE radius</b>	Depicts the "solvent-accessible" surface based on the currently selected atom set. This surface is described by the center of the solvent probe as it rolls along the surface. It is larger than the "molecular" surface. The radius is optional. Starting with Jmol 12.0, if either the <b>VDW</b> or the <b>IONIC</b> keywords are present, <b>sasurface 0</b> is assumed.
<b>SOLVENT radius</b>	Depicts the "solvent-excluded" or "molecular" surface around the currently selected atoms (or the entire model if no atoms are selected). If only a subset of the atoms is selected, then only the corresponding subset of the molecular surface is depicted. This surface is defined as the surface formed by rolling a spherical solvent "probe" around the molecule at the distance of the van der Waals radii of the atoms. The specification of the radius of this probe is optional; its default setting is determined by the <b>set radius</b> command (Jmol default 1.2).

[surface object] -- atomic and molecular orbitals [back](#)

Both atomic orbitals and molecular orbitals can be displayed in Jmol. The "LCAO cartoon" option creates the sort of dumbbell-shaped cartoonish orbitals seen in textbooks in discussion of pi bonding and hybridization.

<b>ATOMICORBITAL n l m Zeff</b>	The Schrodinger solution to the hydrogen atom wavefunction. The three quantum numbers n, l, and m, must be such that abs(m) <= l < n. For solutions with imaginary roots, the two m values simply designate the two real linear combinations of these imaginary solutions. The optional effective nuclear charge, Zeff, determines the overall size of the orbital (default is 6, carbon). Add the keyword PHASE for a two-color orbital, which can be colored using the SIGN keyword followed by two color names or values.
<b>LCAOCARTOON "type" (atom_exp)</b>	Draws a lobe or p orbital (two lobes) centered at the FIRST atom of the specified expression. (Typically this would be an expression for a single, specific atom, such as <b>atommo=3</b> ). See the <a href="#">LcaoCartoon</a> command for a discussion of the possible types of LCAO cartoons (as well as a simpler way to create them).
<b>LOBE {cx cy cz f_ab}</b>	Draws a single tear-drop shaped object (an xy-compressed center lobe of a dz2 orbital) anywhere at any size in any direction with any amount of distortion. The first three numbers define the axis of the lobe; the fourth parameter defines its eccentricity -- the ratio of the other two perpendicular axes to this main axis.
<b>MO n</b>	Denotes the n-th molecular orbital described in the file that is currently loaded. Adjusting the CUTOFF to suit the situation is recommended. Molecular orbitals are automatically bicolor; color them one specific color using COLOR and just one color name or value. RESOLUTION can be used to good effect to increase or decrease the precision of the rendering. <b>Note that only the atom-based orbitals for the currently selected atoms are utilized.</b> (Although, if no atoms are selected, all atomic orbitals are used in the calculation.) Thus, one can selectively see how atomic orbitals from each atom in the molecule are contributing to any given molecular orbital.
<b>MO HOMO/LUMO +/-n</b>	Selects for display a molecular orbital based on energy-proximity to the highest-occupied or lowest-unoccupied molecular orbital. For example, <b>isosurface MO HOMO</b> or <b>isosurface MO LUMO +3</b> .

[surface object] -- general shapes [back](#)

There are several general shapes that can be created as "isosurfaces". These include:

<b>ELLIPSOID {cx cy cz f_ab}</b>	Draws an ellipsoid having a single unique axis. The first three numbers define the "principal" axis; the fourth parameter defines the eccentricity of the ellipsoid -- the ratio of the other two perpendicular axes to this main axis.
<b>HKL {h k l}</b>	Creates a plane through a crystal based on the Miller indices hkl. Adding <b>map molecular</b> creates a slice through the crystal highlighting atomic positions.
<b>PLANE</b>	Indicates that what is desired is not really an isosurface but rather a planar slice through the data set. Using COLOR RANGE, the range of mapped values can be changed. The range -0.008 0.008 is recommended for molecular orbitals. Planes, like other surface objects, can be mapped or left unmapped and just colored. Planes are designated using one of the methods for <a href="#">plane expressions</a> .
<b>SPHERE radius</b>	Draws a sphere of the given radius in Angstroms.

[surface object] -- file/data-derived isosurfaces [back](#)

Isosurfaces can be created in Jmol using external file-based "volume" or "polygon" data.

<b>FILE "filename" n</b>	Depict the n-th isosurface from the specified file. Current file formats supported include:		
	<table> <tr> <td>APBS</td><td><a href="http://apbs.sourceforge.net">http://apbs.sourceforge.net</a> volume data files</td></tr> </table>	APBS	<a href="http://apbs.sourceforge.net">http://apbs.sourceforge.net</a> volume data files
APBS	<a href="http://apbs.sourceforge.net">http://apbs.sourceforge.net</a> volume data files		

	CUBE	<a href="#">Gaussian cube</a> format volume data files
	Chem3D	<a href="#">chem3d</a> 3dxml structure files may contain volume data
	DSN6/OMAP	binary files created by the <a href="#">Uppsala Electron Density Server</a> (Jmol 12.0)
	EFVET	<a href="#">eF-site EFVET</a> format surface files
	Jaguar PLT	<a href="#">Jaguar</a> plt orbital files
	JVXL	<a href="#">Jmol Voxel</a> format files are highly compressed surface files created by Jmol using the <a href="#">write</a> command.
	MRC/CCP4	binary <a href="#">AMI MAP</a> files have the advantage that they contain information about the root mean square deviation for the data set, allowing use of the SIGMA keyword (Jmol 12.0).
	OBJ	<a href="#">AutoDesk Wavefront Advanced Visualizer OBJ</a> surface data files
	PMESH	Jmol can read both ASCII and binary pmesh surface formats (see below)
	XPLOR/CNS	<a href="#">XPLOR MAP</a> files
	Empty quotes indicate that the loaded structure file already has surface data present (CUBE files, for example, contain coordinates), and that that data should be used for construction of the surface. The optional number "n" specifies which volume or surface should be used in the case of a CUBE file with multiple orbitals or a JVXL file with multiple surfaces. Thus, for example, <b>load test.cube.gz;isosurface FILE ""</b> will load the coordinates from the GZIPped cube file, then display its first isosurface. A default directory can be set using <a href="#">set defaultDirectory</a> , and for applets a proxy server can be set for non-host file loading using <a href="#">set appletProxy</a> . For the CUBE file format, units of Bohr are assumed unless the keyword ANGSTROMS precedes this parameter. The keyword FILE is not required.	
FUNCTIONXY "functionName" {originXYZ} {ni xi yi zi} {nj xj yj zj} {nk xk yk zk}	The FUNCTIONXY keyword specifies that the Z-value at a given X,Y coordinate will be provided via a JavaScript function in the case of the applet or via a JmolStatusListener call in an application or, if the function name begins with <b>file</b> :, the numbers will be read from that file (relative to the current directory). The parameters mirror the parameters in the header of a CUBE file. Units of ANGSTROMS are assumed. Thus, we require an origin of the voxel space and for each nominal direction x, y, and z, the number of grid points and the direction of the unit vector along that edge. These four quantities must be in braces. In the case of the Jmol applet, there are three reading options.	
	ni>0 and nj>0	functionName(app, ix, iy) one point is returned for each call to the function.
	ni<0 and nj>0	functionName(app, ni, nj) an entire set of z values separated by white space will be returned as a string. Data are read from low to high coordinates, with X changing in the outer loop (slowly) and Y changing in the inner loop: a11, a12, a13,...a21, a22, a23,..., etc.
	ni<0 and nj<0	functionName(app, ni, nj, Fxy) the Fxy[-ni][-nj] array will be filled with z values in the case of the applet or functionXY(functionName, ni, nj, Fxy) will be called in the case of the application.
	If the functionName starts with the string "file:", then Z value data will be loaded from a file instead of a JavaScript function in the order described above for reading data from a returned string.	
FUNCTIONXYZ "functionName" {originXYZ} {ni xi yi zi} {nj xj yj zj} {nk xk yk zk}	(Jmol 11.8) Similar to FUNCTIONXY, except that in this case the X,Y,Z cube data will be provided in the order (for x = 1 to ni)(for y = 1 to nj)(for z = 1 to nk). The first two options listed for FUNCTIONXY are not available, and the signs of ni and nj are ignored. The data structure Fxyz[ni][nj][nk] must be filled with data by the function call functionName(app, ni, nj, nk, Fxyz) in the case of the applet or functionXYZ(functionName, ni, nj, nk, Fxyz) in the case of the application.	
INLINE @varName	The isosurface data may be in a variable already. For example: <b>x = load("mydata.dat"); isosurface INLINE @x</b> first loads the data into the variable x, then displays the isosurface from that data, possibly giving the opportunity to peek at the data first. It is advisable to reset the variable after use to improve performance, however note that the state will only be preserved if the value of the variable is left unchanged. (Jmol 12.0)	
OBJ "filename"	(Jmol 11.8) The OBJ keyword indicates that the file data to be read comes in the <a href="#">Wavefront Object</a> file format. Jmol uses the vertex and face records of this file in order to create a set of polygons that are colored using the name of the group record, which is assumed to have the hexadecimal format <b>g kRRGGGBB</b> . Note that Jmol does not read material (.mtl) files and so instead relies on this simpler method of assigning colors. Polygons are limited to triangles and quadrilaterals.	
OBJ "filename" n	Optionally, a specific group from the OBJ file can be read. <b>isosurface OBJ "sample.obj" 3</b> , for instance, reads only the third group of faces.	
PMESH "filename"	A "pmesh" is a surface data set consisting of a set of vertices and a set of polygons defining the "facets" of the surface. Polygons are limited to triangles and quadrilaterals. File formats readable by Jmol include:	
	numeric pmesh	This relatively simple format can be found in <a href="#">10x10pmesh.txt</a> . Jmol reads this file in free format -- values simply need to be separated by some sort of white space.
		100 3.0000 3.0000 1.0000

		<pre> 2.3333 3.0000 1.0000 ... (98 more like this) 81 5 0 10 11 1 0 ... (80 more sets like this) </pre> <ul style="list-style-type: none"> <li>• The first line defines the number of grid points defining the surface (integer, n)</li> <li>• The next n lines define the Cartesian coordinates of each of the grid points (n lines of x, y, z floating point data points)</li> <li>• The next line specifies the number of polygons, m, to be drawn (81 in this case).</li> <li>• The next m sets of numbers, one number per line, define the polygons. In each set, the first number, p, specifies the number of points in each set. Currently this number must be either 4 (for triangles) or 5 (for quadrilaterals). The next p numbers specify indexes into the list of data points (starting with 0). The first and last of these numbers must be identical in order to "close" the polygon.</li> </ul>
<b>binary pmesh</b>		<p>A more compact binary pmesh format is described in <a href="#">pmesh.bin.txt</a>. In this case the PMESH keyword is optional, as Jmol can discern the data type from the first four bytes of the file contents. The format has the following specification:</p> <pre> 4 bytes: P M \1 \0 4 bytes: (int) 1 (test for bigEndian) 4 bytes: (int) vertexCount 4 bytes: (int) polygonCount 64 bytes: reserved ---then for each vertex: 12 bytes: (float) x, (float) y, (float) z ---then for each polyhedron, the number of vertices (from 1 to 4) followed by the index of the vertex in the vertex list, starting with 0: [(int)nVertices, (int)v1, (int)v2, ..., (int)vn] </pre>
<b>PMESH INLINE</b> <b>"pmesh-data"</b>		<p>Numeric pmesh data may be specified "in-line" without reference to a separate file. The keyword PMESH is optional but recommended. This is particularly useful for pmesh objects with few vertices. Note that the <a href="#">draw</a> command also can be used for this purpose.</p>

**[additional mapping-only parameters]** [back](#)

If any parameters specifically relate to the mapping of the surface, not the generation of it, then they can come after the specification of the surface object. Keywords such as COLOR RANGE, CONTOUR, DEBUG, FIXED, FULLPLANE, MODELBASED, MAP, REVERSECOLOR, SCALE3D, and SELECT (when it relates to the color mapping) fall into this category.

**MAP [color mapping dataset]** [back](#)

Except for SPHERE, ELLIPSOID, LOBE, and LCAOCARTOON, which have no CUBE-file equivalent, all the other surface types (including FUNCTIONXY starting with Jmol 11.6) can be used as CUBE-type data sets in order to color map another surface, because they all involve the intermediate generation of voxel data within Jmol. Used with PLANE as a surface object, a slice through a set of data can be color-contoured. The keyword MAP is optional, recommended for readability. The keyword set MAP SQUARED indicates that the values should be squared prior to mapping (Jmol 11.4). The Jmol parameter [isosurfacePropertySmoothing](#) (Jmol 11.4; default TRUE) determines whether the property is smoothed out over the isosurface or assigned specifically to the nearest atom.

Atom based-properties can be mapped onto a surface using one of the following options. Starting with Jmol 12.0, if **MEP** or **MEP functionType** is followed by **PROPERTY xxx** or **property\_xxx** or **variable...**, that data will be used instead of partialCharge data.

MEP	molecular electrostatic potential, using partial charge data within the file or assigned to atoms using the [...]. <b>partialCharge</b> = ... or <b>data</b> command syntax. A standard Coulomb function is used (1/d).														
MEP functionType	<div>(Jmol 12.0) allows setting the function used for the mapping as for <b>Chime</b>, where functionType is the number 0, 1, 2, or 3:</div> <table><tr><td>0</td><td>1/d</td><td colspan="2">Coulomb's law distance function (same as rasmol potential distance function)</td></tr><tr><td>1</td><td>e<sup>-</sup>(-d/2)</td><td colspan="2">Gaillard, P., Carrupt, P.A., Testa, B. and Boudon, A., J.Comput.Aided Mol.Des. 8, 83-96 (1994)</td></tr><tr><td>2</td><td>1/(1+d)</td><td colspan="2">Audry, E.; Dubost, J. P.; Colleter, J. C.; Dallet, P. A new approach to structure-activity relations: the "molecular lipophilicity potential". Eur. J. Med. Chem. 1986, 21, 71-72</td></tr></table>			0	1/d	Coulomb's law distance function (same as rasmol potential distance function)		1	e <sup>-</sup> (-d/2)	Gaillard, P., Carrupt, P.A., Testa, B. and Boudon, A., J.Comput.Aided Mol.Des. 8, 83-96 (1994)		2	1/(1+d)	Audry, E.; Dubost, J. P.; Colleter, J. C.; Dallet, P. A new approach to structure-activity relations: the "molecular lipophilicity potential". Eur. J. Med. Chem. 1986, 21, 71-72	
0	1/d	Coulomb's law distance function (same as rasmol potential distance function)													
1	e <sup>-</sup> (-d/2)	Gaillard, P., Carrupt, P.A., Testa, B. and Boudon, A., J.Comput.Aided Mol.Des. 8, 83-96 (1994)													
2	1/(1+d)	Audry, E.; Dubost, J. P.; Colleter, J. C.; Dallet, P. A new approach to structure-activity relations: the "molecular lipophilicity potential". Eur. J. Med. Chem. 1986, 21, 71-72													

	3 e <sup>-</sup> (-d)	Fauchere, J. L.; Quarendon, P.; Kaetterer, L. Estimating and representing hydrophobicity potential. J. Mol. Graphics 1988, 6, 203-206.
	These additional functions thus allow Jmol to use <b>isosurface...MAP MEP</b> to visualize molecular lipophilic potential (MLP) as well.	
<b>PROPERTY xxx</b>	where xxx is an atom-based property value such as partialCharge or temperature or vandervaals	
<b>property_xxxx</b>	The linking underscore signifies that the property was provided separately using the DATA command and is not model-file based. A previous SELECT ...; DATA "property_xxxx"....end "property_xxxx" or, if the data are from a variable, SELECT...; DATA "property_xxxx @x", must have already been issued. Note that when data is created in this way, only the selected atoms are assigned data values. Atoms thus selected need not be contiguous, but the data will be read from the variable or DATA command line contiguously, disregarding spaces, tabs, line ends, and any string that would not evaluate to a number. This allows for targeting just a specific set of atoms for an isosurface and associated data.	
<b>VARIABLE x</b>	The property is in a variable named "x", possibly from a command such as <b>x = load("mydata.txt")</b> . In this case, the variable must contain a value for every atom in the model, even if only a subset of the atoms is being used for the surface.	

[color and contour options] [back](#)

<b>COLOR &lt;color&gt;</b>	Colors an isosurface the specified color name or value, such as [x <b>FF0000</b> ].
<b>COLOR RANGE x.xxx y.yyy</b>	Indicates to color the specified range of value from one end to the other end of the color scheme. If numbers are not included or COLOR RANGE ALL is specified, then the minimum and maximum data values in the file are used. [Jmol 11.4].
<b>COLOR "c1 c2 c3..."</b>	Where c1, c2, etc. are color names or values, such as <b>red</b> or [x <b>FF0000</b> ] within double quotes, this option allows specification of discrete colors to be used for contour mapping.
<b>COLORSCHEME "xxx"</b>	Sets the color scheme to one of "roygb" (default rainbow), "bw" (black/white, Jmol 12.0), "bwr" (blue-white-red), "rwb" (red-white-blue), "wb" (white/black, Jmol 12.0), "low" (red-green), or "high" (green-blue). An optional parameter TRANSLUCENT creates a gradient of translucency from transparent to opaque across the color scheme [Jmol 12.0]. An additional scheme "sets" colors the isosurface different colors for different surface fragments (for example, internal cavities).
<b>CONTOUR n</b>	Specifies to display the object as a set of contour lines. Then number of lines is optional; 9 contour lines are shown by default. Using the CONTOUR keyword sets the default display to be CONTOURLINES NOFILL.
<b>CONTOUR -n</b>	With a negative number, specifies for a plane or f(x,y) object the specific single contour to depict.
<b>CONTOUR DISCRETE [a,b,c,d,e,...]</b>	Sets the contour levels to discrete values. In addition, see the COLOR option, above, for the discrete coloring option. [Jmol 12.0]
<b>CONTOUR INCREMENT {from,to,step}</b>	Sets the contour values starting with the <b>from</b> value to the <b>to</b> value in increments of <b>step</b> . In addition, see the <b>COLOR option, above, for the discrete coloring option.</b> [Jmol 12.0]
<b>REVERSECOLOR</b>	For colorschemes in particular, the REVERSECOLOR option switches the direction of the color scheme. This parameter should be given just prior to the COLORSCHEME parameter.
<b>SCALE3D x.x</b>	generates a 3D plot of the desired scale from a mapped plane. It can be introduced either with the original definition of the isosurface or later, after the plane has been created and displayed. Negative numbers invert the graph (forming valleys instead of mountains); 0 removes the 3D scaling. Note that this rendering can be combined with CONTOUR to form a 3D "topo map". [Jmol 12.0]

[display options] [back](#)

Display options are indicated in the following table. These may be given at the end of the definition of the surface or in a later command having just these keywords and the identifier (or "ALL") of the desired isosurface.

<b>CONTOURLINES/NOCONTOURLINES</b>	[Jmol 11.8] turns on and off contour lines generated using the CONTOUR option (see above).
<b>DOTS/NODOTS</b>	Display dots at each mesh vertex point.
<b>FILL/NOFILL</b>	Display the isosurface as a smoothly filled surface.
<b>FRONTONLY/NOTFRONTONLY</b>	Display only the front half of the surface. This can be useful when the options <b>mesh nofill</b> are used.
<b>FRONTLIT /BACKLIT /FULLYLIT</b>	In some cases the isosurface may appear flat. Using BACKLIT will fix this; FULLYLIT makes exterior and interior surfaces bright; FRONTLIT is the default setting.
<b>MESH/NOMESH</b>	Display a mesh of lines intersecting at the vertexes used to construct the isosurface.
<b>ON/OFF</b>	Turn the isosurface ON or OFF, but do not delete it.

<b>OPAQUE/TRANSLUCENT n</b>	Display the isosurface as an opaque or translucent object. Several translucent options are available; see the <b>color</b> command.
<b>TRIANGLES/NOTRIANGLES</b>	Display separated triangles (primarily for debugging purposes).

#### isosurface AREA

Calculates the area of the current isosurface and stores that value in the **isosurfaceArea** variable. The AREA parameter may also accompany any other parameters in the construction of an isosurface. If the surface is fragmented, the result is an array; otherwise it is a decimal number.

#### isosurface VOLUME

Calculates the volume of the current isosurface and stores that value in the **isosurfaceVolume** variable. The VOLUME parameter may also accompany any other parameters in the construction of an isosurface. If the surface is fragmented, the result is an array; otherwise it is a decimal number.

#### isosurface DELETE

Deletes all isosurfaces.

#### isosurface LATTICE {a b c}

Jmol 12.0 adds the capability to duplicate isosurface areas based on the unit cell lattice. This is a rendering option, so it can be applied any time after an isosurface is created. It is best done with packed unit cells and slabbed isosurfaces. For example: **load quartz.cif ; isosurface slab unitcell vdw; isosurface lattice 3 3 3**

#### isosurface LIST

Lists all isosurfaces

Examples: [in new window using ethene-HOMO.cub.gz](#)

```
isosurface pos05 0.05 "ethene-HOMO.cub.gz";isosurface neg05 -0.05 "ethene-HOMO.cub.gz";
# now load some other surface further out;
isosurface pos01 0.01 "ethene-HOMO.cub.gz";isosurface neg01 -0.01 "ethene-HOMO.cub.gz";color isosurface translucent; # make neg01 translucent
isosurface pos01 nofill mesh; # make pos01 a mesh;
color isosurface translucent; # make the pos01 mesh translucent, too
isosurface neg01 dots; # make neg01 show dots, too
isosurface neg01 nofill; # only dots
isosurface nodots nomesh fill; #everybody is back to a solid ...;color isosurface opaque; # ... and opaque
isosurface neg01; #select neg01;
color isosurface green;
isosurface pos01;color isosurface violet;
slab on; slab 50; # slab in order to see the inside
slab off; # all done!
```

See [examples-11/isosurface.htm](#)

See also:

[\[plane expressions\]](#) [leaoCartoon](#) [mo](#) [pmesh](#) [polyhedra](#)

## label or labels

Atom label parameters  
Column formatting  
Label applications

Turns on and off atom labels based on a previous selection. If a string is given, it is used as the label. See also **hover** and **echo**. Additional options include setting the **font**, **color**, **background**, x- and y-**offsets**, and the text **alignment** (left, center, or right), and using | (vertical bar) as a line separator for multiline labels. Default settings for these characteristics can be set by first issuing **select none**, so that no real label is set.

Atom label parameters [back](#)

Starting with Jmol 11.8, you can specify atom parameters two different ways. You can use the older notation consisting of a percent sign followed by a single character, or you can use a new notation that is more flexible and easier to remember consisting of a percent sign followed by a keyword in brackets. For instance, you can use **select ".ca;label "%n%R"** or **select ".ca;label "%[group] %[resno]"**. See [atom properties](#) for a detailed list of atom properties that can be included in labels.

### Column formatting [back](#)

Standard C++ "pformat" formatting is also available. So, for example, "%0.1x" is the x-coordinate rounded to one decimal place; "%-8.3[partialcharge]" is the partial charge left aligned in an 8-character field with 3 digits to the right of the decimal point; "%05.0[temperature]" is the integer-rounded B-factor zero-filled on the left in a 5-character field.

### Label applications [back](#)

You can use labels for more than just labeling atoms. Using the .label() function you can create variables and save them to files or deliver them to a webpage using JavaScript, thus producing customized output of your choice. For example, the commands

```
x = {*.ca}.label("%5[group] %7[resno] %10.2phi %10.2psi");
write VAR x "phipsi.xls"
```

creates a file that lists phi and psi Ramachandran angles for a protein. (A similar output can be obtained using the [write Ramachandran](#) command.) From a web page, the following JavaScript retrieves bond information for the first silicon atom from an applet:

```
var x = jmolEvaluate('print [_S1][1].bonds.label("%6a1 %6a2 %ORDER %4.2LENGTH")')
```

Note that the [write](#) command can be used with the signed applet from the [console](#) to save data on a local drive even if the applet is from another source, such as <http://www.rcsb.org>.

label ON/OFF {default: ON}

**Labels on** delivers a default label to the currently selected atoms. The default for this label (%[identify]) depends upon the file type and whether more than one file is loaded:

single model, non-PDB data	%[atomName] # %[atomNo]
multiple model, non-PDB data	%[atomName]/%[model] # %[atomNo]. Note that the model will be displayed in "file.model" notation. For example, <b>C12/3.2 #10</b> would be an atom in the third model of the second file loaded.
single model, PDB data	[%[group]][%[sequence]]:[%[chain]].[%[atomName]]%[%[altloc]] # %[atomNo]. (Not all residues will show the chain or altloc, in which case the preceding colon or period, respectively, is not displayed. For example: <b>[MET]1:C.N #608</b> )
multiple model, PDB data	[%[group]][%[sequence]]:[%[chain]].[%[atomName]]%[%[altloc]]/%[model] # %[atomNo]

Note that **labels off** deletes the current formatting, and **labels on** always restores the default label.

label TOGGLE (atom expression)

Toggles the labels on or off for the specified set of atoms.

Examples: [in new window using 1crn.pdb](#)

```
select nitrogen
label %a: %x %y %z
```

See [labels.htm](#)

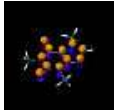
See also:

[color \(atom object\)](#) [echo](#) [font hover set \(highlights\)](#) [set \(labels\)](#) [set echo](#)

## IcaoCartoon

The **IcaoCartoon** command displays cartoonish atomic p and hybrid sp, sp<sup>2</sup>, sp<sup>3</sup> orbitals like those commonly seen in textbooks in discussions of the method of linear combination of atomic orbitals.

Any number of the following options may be strung together in the same command. The [isosurface](#) command can also be used for the creation of LCAO cartoons. Selections for scale, color, and translucency are "persistent" -- carrying over from one command to the next -- and thus need only to be given once per model loading if they are not to be changed.



Note that these options must be given prior to or along with the actual command creating the cartoon; they do not act on Icaocartoons that have already been made. The CREATE keyword and its associated orbital type, optionally with the added keyword MOLECULAR, must be the very last keyword when multiple keywords are involved.

IcaoCartoon ON/OFF {default: ON}

Turn on/off the selected LCAO cartoon.

IcaoCartoon CREATE "[type]"

Creates a new LCAO cartoon of the given type at the currently selected atoms. Of the selected atoms, only atoms with compatible sigma hybridization are used. The CREATE keyword is optional. Valid types include are shown below. In addition to those listed, adding a "-" sign prior to the designation -- "-sp2" for example -- denotes the position 180 degrees rotated from the described position.

"cpk"	[Jmol 12.0] Creates a sphere at the current spacefill radius. The reason this could be useful is that such spheres, though associated with an atom, can be slabbed and capped like an isosurface. This allows for a useful "unit cell only" rendering of spacefill models, for example, using <b>spacefill ionic;IcaoCartoon scale 1.0 CAP unitcell "cpk";spacefill off</b> . (We turn the spacefill off -- it was just to provide the reference sizes.)
"lp"	the "lone pair" position at an AX3E center such as the N atom in NH <sub>3</sub> , with three bonds in sp <sup>3</sup> sigma hybridization, or the sp <sup>2</sup> -hybridized lone pair in an AX2E center such as the unprotonated N atom in histidine
"lpa", "lpb"	the two sp <sup>3</sup> sigma-hybridized lone pair positions at an AX2E bent sigma-hybridized center, such as that in H <sub>2</sub> O.
"px", "py", "pz"	Standard p orbitals at an sp or sp <sup>2</sup> sigma-hybridized center.
"s"	standard spherical s orbital at any center.
"sp2a", "sp2b", "sp2c"	the three sigma bonding/nonbonding positions around an sp <sup>2</sup> sigma-hybridized center.
"sp3a", "sp3b", "sp3c", "sp3d"	the four sp <sup>3</sup> sigma-hybridized positions at any sp <sup>3</sup> sigma-hybridized center.

IcaoCartoon CREATE "[type]" MOLECULAR

Creates a new LCAO cartoon of the given type using the molecular axes system as reference. Applicable only for "px", "py", "pz", "-px", "-py", and "-pz".

IcaoCartoon COLOR [\[RGB-color\]](#)

Colors the orbital one specific color for the Icaocartoon being created with that command or future Icaocartoons.

IcaoCartoon COLOR [\[RGB-color\]](#) [\[RGB-color\]](#)

If this is a p orbital, colors the two lobes different colors for the Icaocartoon being created with that command or future Icaocartoons.

IcaoCartoon DELETE

Delete the LCAO cartoons on the currently selected set of atoms.

IcaoCartoon LIST

Lists all LCAO cartoons

IcaoCartoon SCALE (decimal)

Sets the scale of the LCAO cartoon for the Icaocartoon being created with that command or future Icaocartoons.

IcaoCartoon SELECT (atom expression)

Selects a set of atoms. In the absence of this keyword, the previously selected atom set is used.

IcaoCartoon SELECT "[type]"

For the already selected atom set, selects what type of orbital for an on/off/delete operation.

IcaoCartoon TRANSLUCENT or OPAQUE

Allows for translucent or opaque lobes for the Icaocartoon being created with that command or future Icaocartoons. For translucency options, see [color](#).

where

**[RGB-color]** is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]

Examples:

See [examples-11/lcao.htm](#)

See also:

[\[plane expressions\]](#) [\[isosurface mo pmesh polyhedra\]](#)

## load

(v. 11.8 adds the {ijk i'j'k' -l} and PACKED and VIBRATION options. Also load "@x")

[File types]  
[ZIP/JAR files and JmolManifest]  
[General Options]  
[Crystallographic Options]

Loads the specified file or URL. A wide variety of file types are supported. In general, resolution of file type is based on internal file cues, not the filename or file extension. However, this resolution process can be overridden by specifying a prefix to the file name consisting of the file type followed by two colon characters: **load "molpro(xml)::myfile"**. Multiple files and a selected model from a multi-model file can be read as well. Files may be Gzipped, and multiple files can be read from compressed ZIP and JAR collections. A default directory can be set using [set defaultDirectory](#), and for applets a proxy server can be set for non-host file loading using [set appletProxy](#). After the filename, the options listed below are available. These options must appear in the order given in the table below. In addition, the load command can take parameters that specify the number of unit cells to generate, the space group or Jones-Faithful operators to use, and the unit cell dimensions. Using these parameters, the space group and unit cell information within a file can be overridden, and even simple XYZ data files can be turned into crystallographic data sets. Starting with Jmol 11.6, the FILTER keyword allows selective loading of atoms as well as construction of the biologically relevant molecule (PDB "BIOMOLECULE" records. See also the headings [load APPEND](#), [load FILES](#), [load MENU](#), [load MODELS](#), and [load TRAJECTORY](#). Note that with the Jmol application (not the applet) you can also use Edit...Paste to load molecular coordinate from the system clipboard. The same capability for the applet can be had using [data "model"](#).

[File types] [back](#)

Jmol reads a large number of data formats. Example files can be found at <http://jmol.svn.sourceforge.net/viewvc/jmol/trunk/Jmol-datafiles>. Supported file types include:

ADF	<a href="#">Amsterdam Density Functional</a> output file
AIMS	(Jmol 11.8)
AMPAC	<a href="#">AMPAC</a> file (Jmol 12.0)
Argus(XML)	<a href="#">ArgusLab</a> AGL file
Chem3D(XML)	CambridgeSoft <a href="#">Chem3D</a> C3XML file
CASTEP	<a href="#">CASTEP</a> file format (Jmol 11.8)
CIF	International Union of Crystallography <a href="#">Crystallographic Information File</a> , including Macromolecular Crystallographic Information file (mmCIF)
CML(XML)	<a href="#">Chemical Markup Language</a> file
CRYSTAL	<a href="#">Crystal09</a> solid state computation output, including support for 1D (polymer) and 2D (slab) periodicity. This file format creates the atom properties <b>property_spin</b> and <b>property_magneticMoment</b> [Jmol 12.0].
CSF	Fujitsu <a href="#">Sygress Explorer</a> (formerly CAChe) chemical structure file, including the reading of ab initio, semiempirical, gaussian, and density functional molecular orbitals
CUBE	Gaussian <a href="#">cubegen</a> output file
DGRID	Jmol 12.0 adds a <a href="#">DGRID</a> file reader. These files are generalized representations of output from a variety of quantum mechanical calculation packages, including especially <a href="#">ADF</a> .
FoldingXYZ	XYZ file created by the <a href="#">Folding@home project</a>
GAMESS	<a href="#">General Atomic and Molecular Electronic Structure System</a> output file
Gaussian	<a href="#">Gaussian</a> output file
GhemicalMM	<a href="#">Ghemical</a> molecular mechanics file (MM1GP)
GRO	<a href="#">GROMACS</a> .gro file format (Jmol 11.8)
HIN	<a href="#">HyperChem</a> native file

Jaguar	National Center for Supercomputing Applications Jaguar output file																
JME	<a href="#">Java Molecular Editor</a> file format (a 2D, not a 3D, format)																
MDTOP, MDCRD	<a href="#">AMBER</a> Molecular dynamics topology files and associated coordinate files. (Jmol 11.8)																
MOL, MOL2	Symyx (formerly Molecular Design) <a href="#">structure data files</a> , including SDF and CTAB V2000 files																
MOLDEN	<a href="#">Molden</a> data file																
MOLPRO(XML)	<a href="#">Molpro</a> structure file																
Mopac	<a href="#">OpenMopac</a> output file (MOPOUT)																
MopacGraphF	<a href="#">OpenMopac</a> GRAPHF output file (for molecular orbitals)																
NWCHEM	Pacific Northwest National Laboratory <a href="#">NWChem</a> output file																
Odyssey	WaveFunction <a href="#">Odyssey</a> data file (ODYDATA)																
Odyssey(XML)	WaveFunction <a href="#">Odyssey</a> XODYDATA file																
PDB	<a href="#">Protein Data Bank</a> file																
PQR	Position/Charge/Radius data file produced by the <a href="#">Adaptive Poisson-Boltzmann Solver</a> project																
PSI	<a href="#">PSI3</a> output reader (coordinates only)																
QCHEM	<a href="#">Q-Chem</a> output file																
SHELX	<a href="#">SHELX</a> output file																
Spartan	WaveFunction <a href="#">Spartan</a> data file																
SpartanSmol	WaveFunction binary <a href="#">Spartan</a> SMOL data file, including full MacSpartan Spartan directories in ZIP format																
V3000	Symyx (formerly Molecular Design) <a href="#">V3000 Connection Table</a> (CTAB or SDF) data file																
WebMO	<a href="#">WebMO</a> molecular orbital file reader																
VASP	Vienna Ab Initio Simulation Package <a href="#">VASP</a> vasprun.xml files.																
Wien2k	<a href="#">Wien2k</a> data files. WIEN2k performs electronic structure calculations for solids using density functional theory. Using the option <b>spacegroup "none"</b> disregards symmetry information given in the file and simply reads the atom coordinates given in the file, including MULT atom records. For example, <b>load t.struct {1 1 1} spacegroup "none"</b> (Jmol 11.8)																
XYZ	Minnesota Supercomputer Institute XMol file format. Various extensions to this file format allow reading of the following information separated by whitespace: <table><tr><td>element</td><td>x</td><td>y</td><td>z</td><td>vibX</td><td>vibY</td><td>vibZ</td></tr><tr><td>element</td><td>x</td><td>y</td><td>z</td><td>charge</td><td>vibX</td><td>vibY</td><td>vibZ</td><td>atomNumber.</td></tr></table> In this last format, introduced in Jmol 12.0, if the charge is an integer, it is read as <b>formalCharge</b> ; if it is decimal, then as <b>partialCharge</b> . Any information past x y z is optional, and if missing or uninterpretable as a number (for example, "X" or "--") will be ignored.	element	x	y	z	vibX	vibY	vibZ	element	x	y	z	charge	vibX	vibY	vibZ	atomNumber.
element	x	y	z	vibX	vibY	vibZ											
element	x	y	z	charge	vibX	vibY	vibZ	atomNumber.									

[ZIP/JAR files and JmolManifest] [back](#)

Jmol can read specific files within compressed [ZIP and JAR collections](#). In addition, for the **load** command specifically, Jmol will look for a file within the collection with the name **JmolManifest** and follow the directives within it. These directives may be as simple as a list of files to be loaded, one filename per line (which will be read in the order listed). Lines starting with # are comment lines, which may contain any text, but also may contain one or more of the following keywords:

#EXCEPT_FILES	The list of files specifies files to ignore, not files to load; all other files will be loaded.
#IGNORE_ERRORS	Try to read files, but ignore errors when a file is not recognized as a valid model format that Jmol can read. This option allows easy "mining" of ZIP collections for files that Jmol can recognize, ignoring all others.
#IGNORE_MANIFEST	Ignore this manifest in its entirety -- simply read all files in the order retrieved by the ZIP file iterator.

[General Options] [back](#)


The following options may be indicated after specifying the filename. Each is optional, but if more than one option is indicated, options must be given in the order listed in this table.

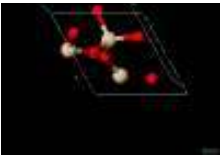
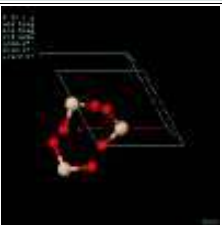
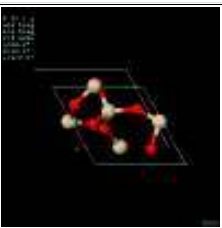
MANIFEST	(Jmol 11.4) If the file being loaded is a ZIP or JAR file, Jmol will search for a file in that compressed file collection with the
----------	--

"manifestOptions"	name "JmolManifest" and process it accordingly (see above). However, in the <b>load</b> command, if the keyword <b>MANIFEST</b> and a quoted string follows the filename, then Jmol will use this string as the manifest instead, with lines of the manifest separated by vertical bar " " characters. In this way, standard ZIP collections can be read, and the order of file loading can be specified. For example: <b>load "test.zip" manifest "CH3CL.MOL CH4.MOL"</b> reads only these two files from the ZIP collection, in this order. If the file contains a manifest and that manifest is simply to be ignored, the quoted string should read "IGNORE_MANIFEST".		
(integer)	Loads only the specified model number (if positive) or vibration number (if negative, starting with Jmol 12.0), skipping the others. For PDB files, the number indicated is the number specified in the MODEL record. (Not supported by all file types.) Starting in Jmol 11.8, see also <a href="#">load MODELS</a> .		
{unitCell(s)}	see Crystallographic Options, below		
<b>FILTER</b>	(Jmol 11.6) The FILTER parameter specifies file-type specific load options. Options should be separated by commas. For atom selection, more than one specification may be made using a comma between specifications: *:1,*:2, but * and ! may not be mixed within any one type (atom name, group name, or chain ID). These include		
	file type	option	description
	PDB	BIOMOLECULE n	load only the specified biomolecule, as specified in the REMARK 350 APPLY THE FOLLOWING TO CHAINS record, applying the symmetry transformations specified in the REMARK 350 BIOMT records
	PDB	NOSYMMETRY	Along with BIOMOLECULE, indicates that the symmetry transformations in the REMARK 350 BIOMT records should be ignored.
	PDB	#i or !#i	select only BIOMT transformation i (#i) or NOT transformation i (!#i). Note that BIOMT #1 (the identity) cannot be deselected -- it is always part of the loaded set of atoms.
	PDB and CIF	*.XX or !.XX	select only atoms of the designated type (*.XX) or only atoms NOT of the specified type (!.XX).
	PDB and CIF	[XXX] or ![XXX]	select only atoms of the designated group name ([XXX]) or only atoms NOT of the specified group name (![XXX]).
	PDB and CIF	*:X or !:X	select only atoms in chain X (*:X) or only atoms NOT of chain X (!:X).
	GAMESS, Gaussian, QChem	NBO	(Jmol 11.8) Specifies to read only natural bonding orbitals created with the NBO 5.0 option <b>AONBO=P</b> .
	GAMESS, Gaussian, QChem	EIGEN or !NBO	(Jmol 11.8) Read only standard eigenfunctions.
	GAMESS, Gaussian, QChem, NWChem	nboCharges	(Jmol 11.8) Use atomic partial charges from the NBO calculation. Note that this flag is independent of the <b>NBO</b> filter option. To use both, separate them with a comma: <b>load "myfile.out" FILTER "NBO,nboCharges"</b> .

[Crystallographic Options] [back](#)

The following crystallographic options may be indicated, starting with the specification of the unit cell. If more than one option is indicated, options must be given in the order listed in this table.

{unitCell(s)}	(Jmol 11.0) Jmol can read unit cell and symmetry information from selected file types (for example, CIF, PDB, or SHELX). The specific set of unit cells to load can be specified one of two ways -- either using the notation {i j k} or the notation {ijk i'j'k' n}.		
	{i j k}	Loads a block of unit cells between the origin, {0 0 0} and the specified unit cell system coordinate. Used alone, {i j k} is only for working with files containing both unit cell and space group information (CIF, SHELX, CML, for example). The particular choice {3 3 3} is significant, in that it loads 27 unit cells, forming a solid block around a central cell. The unit cell display can then be moved to the origin of this central cell using <b>unitcell {1 1 1}</b> , and the display of atoms can be restricted to that center cell using <b>restrict cell=666</b> or <b>restrict cell={2 2 2}</b> . Multiple unit cell loading can be combined with the single-model loading by indicating the model number first, then the number of unit cells: <b>load "myfile.cif" 15 {3 3 3}</b> . Quotes are not required. Starting in Jmol 11.1, there is no restriction other than memory on the size of i, j, and k (except that all must be positive).	
	{ijk i'j'k' -1}	Loads a block of unit cells within the range ijk and i'j'k' (which should include 555) and packs all atoms into the designated set of cells. The keyword <b>PACKED</b> may be used in place of {555 555 -1} or	

	after a designated set of cells: <b>load t.struct {2 2 2} PACKED (Jmol 11.8)</b>	
{ijk i'j'k' 0}	Loads a block of unit cells within the range ijk and i'j'k' (which should include 555) WITHOUT normalizing the operators. All symmetry-generated atoms are placed based on the exact definition of the symmetry operations found in the file or designated using the <b>spacegroup</b> keyword (see option below). Note, however, that if explicit operations are not provided and therefor must be generated from a spacegroup name, they will be normalized. The list of operations used can be obtained using <a href="#">show symmetry</a> .	
{ijk i'j'k' 1}	Loads a block of unit cells within the range ijk and i'j'k' (which should include 555), normalizing the operators to move the geometric center of the generated set of atoms into cell 555, then applying the lattice translation. Thus, <b>load "filename" {555 555 1}</b> is equivalent to <b>load "filename" {1 1 1}</b> . For example, <b>load "myfile.cif" {444 666 1}</b> loads a block of 27 unit cells, with the geometric center of all units with the bounds of the fractional coordinate range {-1 -1 -1/} to {2 2 2/}.	
<b>RANGE x.xx</b>	(Jmol 11.4) Restricts the atoms loaded to those within a given range in angstroms. If x.xx is positive, then this range is relative to the entire set of atoms that would be generated using <b>load "filename" {1 1 1}</b> ; if x.xx is negative, then the range is relative to the atoms that would be generated using just <b>load</b> itself (the base x,y,z symmetry set).	
<b>SPACEGROUP "name"</b>	(Jmol 11.0) Loads a block of unit cells between the origin, {0 0 0} and the specified unit cell system coordinate. In addition, the symmetry inherent in the file is ignored, and the specified space group is applied. Quotes are required around the space group name. If the space group name itself includes double quotes, use two single quotes or an "escaped double quote" (") instead. For example: <b>P 32 2'</b> (single quotes here) or <b>P 32 2'</b> , not <b>P 32 2"</b> . Generally Jmol reads the Jones-Faithful operators from a file, however if the spacegroup name is specified as "ignoreOperators", Jmol will ignore any explicit file-based Jones-Faithful operators and instead create the symmetry based on parsing of the space group symbol in the file (Hermann-Mauguin, Hall, or international table number). If the name is a semicolon-separated list of Jones-Faithful operators, such as "x,y,z;x+1/2,y,z", Jmol will ignore any explicit file-based operators and instead create the symmetry based on the list provided.	
<b>UNITCELL {a b c alpha beta gamma}</b>	(Jmol 11.0) Specifies the unit cell to use for models in this file. If a unit cell is specified in a file, then this option allows overriding that specification. When both SPACEGROUP and UNITCELL are provided, Jmol can display molecules found in standard Cartesian coordinate files (XYZ, MOL, PDB) as packed unit cells.	

load

The load command by itself reloads the current file.

load "filename" (integer)

Jmol automatically determines file type based upon the contents of the file. Quotes are recommended. Files containing fractional coordinates are displayed with their unit cell visible. **load ""** reloads the current file. For files containing multiple models, an optional integer after the file name will load only the specified model. If this number is negative, it refers to a specific vibrational mode in files that contain that information (Jmol 12.0). If this number is 0, it refers to the last model in the set (Jmol 12.0).

load "filetype::filename"

Starting with Jmol 11.2, file format can be forced by prefixing a filename with "xxxx:" where "xxxx" is a Jmol file type. This should not be necessary in most cases, since Jmol determines file type by scanning the first lines of a file for file-type-specific content. In certain cases, however, where there are extensive comments at the beginning of a file, or a file type

(such as MDCRD) does not have any distinguishing characteristics, it is possible for Jmol to fail to discover the file type or to misassign it. In that case, adding "xxxx:" is necessary.

load @variableName

Loads the file with the name specified by variable **variableName**. Jmol 12.0 adds the capability to load a set of files that are defined in an array variable.

load "@variableName"

Starting with Jmol 11.8, you can load a model from data contained in a variable. This allows modification of the data prior to display, for example. Quotes are necessary, as without them -- load @x -- it is the file name that is expected in variable x, not the file contents. For example: **x = load("quartz.cif");load "@x" {2 2 2};reset x**. Note that to save memory, it is a good idea to clear the variable using **reset x** just after loading. An interesting aspect of this option is that it allows data from a remote file to be saved in the state. This means that if you use **write state "somefile.spt"**, then that single state script will contain the full file data in a DATA statement. It will be "transportable" and no additional model file will be required.

load =XXXX

Starting with Jmol 11.2, you can load PDB files directly from <http://www.rcsb.org> or another server of your choice. (This depends upon the setting of [set loadFormat](#).) Simply preface the four-letter PDB id code with "=" and, and Jmol will load the file. Starting with Jmol 12.0, adding "AS ." ("as" with a period) you can save that file automatically in the default directory (as xxxx.pdb.gz), and using, for example, **load =1crn AS "myfile.pdb"**, you can save it to some other local file name. Starting with Jmol 12.0, the default is to transfer the file in g-zipped format; If you add ".pdb" to the expression -- **load =1crn.pdb**, for example -- then Jmol will transfer and save the uncompressed PDB file.

load \$XXXX

Starting with Jmol 12.0, you can load SMILES strings, and Jmol will turn them into 3D models using the [smi23d server](#) at Indiana University. As for reading files from any source outside your domain, you will have to use the signed applet or Jmol application to do this. The service uses PCMODEL v9.1 for its conversion and should deliver stereochemistry appropriately. These files can be saved as MOL files using [write xxx.mol](#), and if the conformation is not to your liking, switching to [set modelkitMode](#) or using [set picking dragMinimize](#) you can quickly adjust the model to the desired conformation.

load SMILES "smilesString"

An alternative to the \$ syntax for loading SMILES strings that allows for the space character.

load keyword "filename"

An optional keyword (APPEND, FILES, MODELS, or TRAJECTORY) may be supplied prior to the quoted filename. Other keywords are ignored. (Jmol does not use the Chime-style keyword to specify "file format". Rather, starting with Jmol 11.2, file format can be forced by prefixing a filename with "xxxx:" where "xxxx" is a Jmol file type. However, this should not be necessary in most cases, since Jmol determines file type by scanning the first lines of a file for file-type-specific content. (In certain cases, where there are extensive comments at the beginning of a file, it is possible for Jmol to fail to discover the file type or to misassign it. In that case, xxxx: should be used.)

load "filename" FILTER "filter specification"

For individual file types, it is possible to filter the data in the file as it is loaded. The FILTER keyword followed by a quoted string allows for this. Specific filters include:

file type	filter
CIF, GROMACS, PDB	[XXX] or ![XXX], .XXX; or !.XXX; :X, !:X to specify inclusion or exclusion of specific residue types, atom types, or chains. "!" indicates NOT, multiple selections are treated as "OR" without "!" and "AND" when "!" is present, for example, <b>load "1sva.pdb" FILTER "!=CA" loads only alpha carbons; ...FILTER "![HOH]" filters out water molecules.</b>
CRYSTAL	<b>FILTER "input"</b> load input coordinates only; <b>FILTER "NOVIB"</b> do not load vibrations; <b>FILTER "CONV"</b> load conventional, not primitive cells.
Cygress	<b>FILTER "noOrient"</b> prevents application of the rotation matrix found in the file as the default rotation.
GAMESS (US)	<b>CHARGE=LOW</b> indicates to load Lowden charges rather than Mulliken charges.
GAMESS, GAUSSIAN, GenBO, Jaguar, PSI, and QCHEM	<b>FILTER "xxx" or "yyy" or FILTER "!"xxx !yyy ...</b> where "xxx" and "yyy" are words on the line Jmol uses to identify a molecular orbital. This allows selective loading of specific types of molecular orbitals -- such as "alpha", "beta", or "NBO" -- for any of these file types, "POPULATION" or "EDMISTON" or "PIPEK" for GAMESS,
JME	<b>FILTER "noMin"</b> loads the 2D file, adjusts atoms in preparation for minimization, but does no minimization.
MOL	<b>FILTER "2D"</b> indicates to consider the file a 2D file and to apply a automatic hydrogen addition and 2D-to-3D conversion immediately upon loading. "2D-noMin" does the hydrogen addition but no minimization.
PDB	<b>FILTER "BIOMOLECULE n"</b> , where n is a number > 0 indicating which biomolecule to load. In addition, <b>#n</b> or any number of <b># n</b> or <b>!#n</b> can be indicated in order to load just a specific subset of biomolecular transformations related to the specified biomolecule.
SPARTAN	<b>FILTER "noOrient"</b> prevents application of the rotation matrix found in the file as the default rotation.

load "remoteFilename" AS "localFileName"

Loads a remote file and then also saves it locally. For use with the Jmol application and signed applet only. (Jmol 12.0)

Examples:

See [examples-11/sym.htm](#)

See also:

[initialize set \(files and scripts\) zip](#)



[top](#) [search](#) [index](#)

## load APPEND

Adding APPEND to the load command appends a file or a set of files to the current model set without replacing the current model. (Jmol 11.2) Load parameters FILTER (Jmol 11.6), MANIFEST (Jmol 11.4), MODELS (Jmol 11.8), TRAJECTORY (Jmol 11.8), or unit cell/symmetry specification may be included with the APPEND keyword. By default, a new frame is created for each model added. For example:

load "myfile.pdb";load APPEND "myligand.xyz";frame \*:display 1.1,2.1

If **set appendNew false** has been issued and only one model is in the appended file, then the file model is added to the currently displayed model, creating no additional frames. Starting in Jmol 11.4, multiple files may be appended if the specified file is a ZIP file. In that case, specifying the manifest in the command after the filename is an option.

[top](#) [search](#) [index](#)

## load DATA

(v. 12.0)

Starting with Jmol 12.0, the **DATA "model..."** and **DATA "append..."** commands are deprecated in favor of **load DATA...** All of the parameter options for the **load** command are thus also available for the **data** command.

[top](#) [search](#) [index](#)

## load FILES

Adding FILES to the load command indicates that a set of files are to be loaded (Jmol 11.2). Filenames must be quoted. Each model encountered is put into a new "frame." No additional parameters other than COORD are allowed (see below).

load FILES "filename1" "filename2" <#>

For multiple file loading, all parameters after the first must appear in quotes. Each model is loaded into a new frame, and frames are addressed using a decimal notation -- 1.1 for the first model in the first file, 1.2 for the second model in the first file, 2.1 for the first model in the second file, etc. For example, **select (oxygen) and (1.3, 1.4)** selects all oxygens in the third or fourth model of the first file loaded. See also [set backgroundModel](#). (For backward compatibility with Jmol 11.0, frames may also be addressed by 1001, 1002, 2001, 2002, etc. where 1002 corresponds with "1.2", above.)

[top](#) [search](#) [index](#)

## load MENU

The special command **load MENU** allows loading of a custom Jmol popup menu.

load MENU "menufile"

File format is that written by Jmol using the [show MENU](#) or [write MENU](#) command. See examples at [misc/Jmol.mnu](#) and [misc/test.mnu](#).

[top](#) [search](#) [index](#)

## load MODELS

The MODELS keyword allows loading of a specific subset of models from a file that contains a model collection. Introduced in Jmol 11.8, two syntaxes are available. Note that model numbers refer to the sequence of models encountered in the file, starting with 0, and typically do NOT correspond to the model numbers indicated in PDB MODEL records. Once the models are loaded, they can be accessed using the file.model decimal notation as 1.1, 1.2, 1.3, etc.

load MODELS {first last stride} "filename"

In the first syntax, first and last models along with a "stride" (step) are specified. Models are then read using an equivalent of **for (i = first; i <= last; i = i + stride)**. Numbers start with 0 for the first model; -1 for **last** indicates "to the end of the file." For example, **load MODELS {0 10 2}** "..." would load six models, models 0, 2, 4, 6, 8, and 10.

load MODELS ((i j k:l m...)) "filename"

In the second syntax, a list of the specified files to load is given in the Jmol "bitset" syntax. This syntax uses braces within parentheses. Specific models are listed, starting with 0 for the first model. Ranges of models are indicated using colons. For example, **load MODELS {{0 2 4:6}}** "..." loads five models -- models 0, 2, 4, 5, and 6. **load MODELS {{1}}** "..." loads the second model only.

[top](#) [search](#) [index](#)

## load TRAJECTORY

Loads a file as a "trajectory", meaning a single model with a series of atom coordinates. (Similar to the sort of multiple-frame animations that the Chime plug-in was able to load.) Loading a file as a trajectory saves substantially on memory requirements, since there is only one set of atoms and bonds. The defining restriction for trajectories is that only one frame can be viewed at a time. Each trajectory is loaded into its own frame as though it were a distinct model, and frames are accessed as usual using the [frame](#) or [model](#) command. In addition, any reference to a specific trajectory, such as **select 1.3**, switches to that set of coordinates, and [Measurements](#) and the position of [cartoons](#) are automatically recalculated when a new trajectory is displayed. Changes to colors in one trajectory effect the same change for all trajectories, since there is really only one model, just different atom positions. Starting with Jmol 11.8, multiple files may be loaded as independent trajectories using APPEND TRAJECTORY in place of simply TRAJECTORY. Jmol 12.0 allows reading of PDB files that contain no MODEL line and are instead simply concatenated versions of the same atoms as trajectories using the TRAJECTORY keyword. Each model must start with atom number 1 for this to work.

load TRAJECTORY "filename"

Loads the specified file as a trajectory.

load TRAJECTORY {first last stride} or ((i j k:l m...)) "filename"

Loads the specified subset of models from the file as a trajectory. See [load MODELS](#) for details.

load TRAJECTORY "filename" FILTER "filter specification" COORD {first last stride} or ((i j k:l m...)) mldcd::crdfile1

The presence of a COORD keyword indicates that the trajectory is to be built from a set of files that includes an AMBER molecular dynamics topology file and associated coordinate files. FILTER is optional, but recommended. For example, **FILTER "![WAT]"** prevents loading of water molecules. Any number of coordinate files may be specified. Coordinates are loaded as a set of trajectory steps. At least one COORD keyword must be given, and each specification of first, last, and stride must be preceded by a COORD keyword. {first last stride} or ((i j k:l m...)) is optional, and if not provided defaults to {0 -1 1}, which is interpreted as "all trajectory steps from each file." See [load MODELS](#) for documentation on {first last stride} and ((i j k:l m...)).

[top](#) [search](#) [index](#)

## load [property]

Adding the keyword OCCUPANCY, PARTIALCHARGE, TEMPERATURE, VIBRATION, or XYZ to the load command instructs Jmol to load only that sort of information from a file.

<b>load OCCUPANCY</b>	Occupancy data are in the form of integers between 0 and 255. Numbers less than 0 are saved as 0; numbers greater than 255 are saved as 255.
<b>load PARTIALCHARGE</b>	Partial charge data are in the form of decimals roughly in the range +/-3.4 x 10 <sup>-38</sup> .
<b>load TEMPERATURE</b>	Temperature (B-Factor) data are saved as decimals with 0.01 precision within the range -327.68 to 327.67.
<b>load VIBRATION</b>	Vibrational data are in the form of vectors.

<b>load XYZ</b>	Loads only the coordinates, replacing the current coordinates of an already-loaded model with those in the file to be read. (Jmol 11.8)
-----------------	---

The data are applied to the currently selected set of atoms based solely on atom position. All standard load parameters are accepted, although many will be ignored, however whereas the default for a normal LOAD operation is to load all files, the default is to read data only from the first model in a multi-model file (or the specific model indicated with an integer after the file name). For each "atom" position and vector that is read, Jmol applies the data to all selected atoms having a unit cell normalized position within [loadAtomDataTolerance](#) (default 0.01) Angstroms of the position read from the file. If the file being loaded contains embedded Jmol script commands, those commands will be processed after the application of the data. For example, **load "myfile.struct" {5 5 1} PACKED; select \_O; set loadAtomDataTolerance -0.2; load VIBRATION "vibs.xyz"** 3 first loads a set of unit cells from myfile.struct, then applies only to the oxygen atoms the third vibration set found in vibs.xyz. Oxygen atoms in all unit cells will be given data even though the data in vibs.xyz might only be for one unit cell. (Jmol 11.8)

See also:

[vibration](#)

[top](#) [search](#) [index](#)

## LOG

(v. 12.0 -- new)

Logs data to a file. Jmol 12.0 adds a new command specifically for the signed applet and the application. The LOG command works the same as [print](#) but records the information in a log file. If the printed data starts with the characters NOW, then those are replaced by the date and time. For example: log "NOW " + getProperty("modelInfo"). The file to log to must first be designated using **set logFile "someName"**. This name will be prepended with "JmolLog\_" and must not contain any directory path. The file will always be created in the Jar file directory. Note that logging is not ever possible with the web-based version, even with the signed applet, but signed applet or application running locally can log to a file. In addition to explicit use of the LOG command, two settings, **set logCommands** and **set logGestures** allow automatic tracking of commands and gestures (swipe, pinch, zoom, spin) to the designated log file.

Note:

The LOG command does not require @ { ... } around Jmol math expressions.

[top](#) [search](#) [index](#)

## loop

Causes the script to restart at the beginning, with an optional time delay. In Jmol 11.0 when the default [set scriptQueue ON](#), a looping script can only be stopped using the script command [quit](#) or [exit](#) either alone or at the beginning of another script. In Jmol 11.2, see also [goto](#).

loop [\[time-delay\]](#)

loop on

where

[\[time-delay\]](#) is in seconds -- (integer|decimal, >=0)

Examples: [in new window using 1blu.pdb](#)

color bonds red  
delay 3  
color bonds green  
loop 1

See also:

[delay](#) [exit](#) [goto](#) [pause](#) [quit](#) [resume](#) [step](#)



[top](#) [search](#) [index](#)

## mapProperty

(v. 12.0 -- new)

The mapProperty command allow copying of an atomic property from one set of atoms to another. The operation involves identifying two sets of atoms and associated properties and also a common "key" property such as atomno or resno. If no key is given, atomno is assumed. A shortcut allows quick transfer of atom selection.

```
mapProperty {atomExpression1}.property1 {atomExpression2}.property2 propertyKey
```

For each atom Y in atomExpression2 that matches an atom X in atomExpression1 based on propertyKey, Y.property2 is made to equal X.property1. Property2 must be setttable. For example, **mapProperty {1.1}.temperature {2.1}.property\_t atomno; color {2.1} property\_t "rwb"** , which would color atoms in model 2.1 (perhaps a [plot](#)) based on temperature values for model 1.1.

```
mapProperty SELECTED {atomExpression} propertyKey
```

This form of the mapProperty command is a shortcut for **mapProperty {selected}.selected {atomExpression}.selected propertyKey**. For example, **mapProperty SELECTED {2.1}** selects atoms in model 2.1 that match atomno with atoms that are already selected.

[top](#) [search](#) [index](#)

## measure or measures or monitor or monitors

(v. 11.0 -- adds several new capabilities)

Renders a measurement between the specified atoms. See also [set \(measure\)](#). Two general syntaxes are available. In the older syntax, a series of two to four atom numbers are given, and the appropriate measure (distance, angle, or dihedral angle) is then displayed. The newer, more general syntax is as follows:

```
measure RANGE <minValue> <maxValue> ALL|ALLCONNECTED|DELETE (<atom expression>) (<atom expression>) ...
```

Using this syntax one can specify a set of measurements to define all at once. Note that these sets are embedded [atom expressions](#) that must be enclosed in parentheses. If neither ALL nor ALLCONNECTED is present, only the first matching atom in the entire model set (all frames, so probably the first frame) is matched in each atom expression. When ALL or ALLCONNECTED is specified, all matching criteria in all frames are generated, thus allowing for "animated" measures. In general, this syntax restricts measurements to within the same model. However, measures can also be between two atoms in different frames (different models) as long as each atom expression evaluates to a single, specific atom. (To specify a particular atom in a particular model, use "AND \*n", where n is the model number, "ATOMNO=3" by itself, for example, will indicate the third atom in each model/frame, but "ATOMNO=3 and \*/6" specifies only atom 3 in model 6). If a measurement is made between atoms in different models, both models must be displayed in order for the measurement to appear. A simple way to display two specific models is to use [display \\*/i or \\*/j](#), where i and j are two model numbers.

Thus, for example, **measure (\*) (\*)** will measure nothing, because both expressions will simply match the first atom in the first frame. **measure allconnected (\*3) (\*3) (\*3)** will measure every angle associated with bonds for model 3; **measure allconnected (\*) (\*)** will measure every bonded distance in every loaded model; **measure all (\*) (\*)** measures all possible interatomic distances in all models (not recommended!).

For the applet, using [getProperty measurementInfo](#) will then deliver full information relating to all measurements.

```
measure ON/OFF {default: ON}
```

Turns on and off the distance, angle, dihedral measurement labels and measurement lines. (To turn off just the labels, use [set measurement OFF](#)

```
measure "n:labelFormat"
```

Changes all previously defined measurement labels of a given type (n = 2, 3, or 4) to the indicated format. The default label is "%VALUE %UNITS" for all types. Also available is "%#(percent number-sign)", which gives the 1-based number of the measurement. Atom information can be included as for [labels](#), adding 1 or 2 to the format code to indicate which atom. So, for example, **set defaultDistanceLabel "%a1 -- %a2 distance = %0.0VALUE"** delivers the two atom names along with the value of the measurement rounded to the nearest integer with no units indicated.

```
measure (two to four atom expressions, each in parentheses) "labelFormat"
```

Show the distance, angle, or dihedral angle formed by the FIRST atom in each atom expression. The format is optional.

```
measure (integer) (integer) "labelFormat"
```

Two atoms specify a distance measurement with an optionally given format. Prior to Jmol 11.8, this older syntax selected the first atom in the overall set of models with the given atom number. Starting with Jmol 11.8, this syntax selects the first atom in the currently visible frame set.

```
measure (integer) (integer) (integer) "labelFormat"
```

Three atoms specify an angle measurement. The format is optional. Prior to Jmol 11.8, this older syntax selected the first atom in the overall set of models with the given atom number. Starting with Jmol 11.8, this syntax selects the first atom in the currently visible frame set.

```
measure (integer) (integer) (integer) (integer) "labelFormat"
```

Four atoms specify a dihedral angle measurement. The format is optional. Prior to Jmol 11.8, this older syntax selected the first atom in the overall set of models with the given atom number. Starting with Jmol 11.8, this syntax selects the first atom in the currently visible frame set.

```
measure TICKS X|Y|Z {major,minor,subminor} FORMAT [%0.2f, ...] SCALE {scaleX, scaleY, scaleZ} |x.xx FIRST x.xx {point1} {point2}
```

Creates a measure line with ticks along it. There are three levels of ticks - major, minor, and "subminor." Only the major ticks have labels. Which of these tick levels are displayed and the distance between ticks depends upon the parameter that takes the form of a point. The optional keyword FORMAT allows formatting of the labels for the major ticks. These are based on an array of strings given after the FORMAT keyword. If the array is shorter than the number of ticks, the formats in the array are repeated. Following that, the optional keyword SCALE allows setting the scale either for each axis direction independently (scaleX, scaleY, scaleZ) or overall (as a decimal number). An optional keyword FIRST allows setting of the initial value of the measure. Finally, two points must be indicated.

```
measure ALL (two to four atom expressions) "labelFormat"
```

Show the distance, angle, or dihedral angle formed by ALL atoms in the first expression with ALL atoms of each additional atom expression. The format is optional.

```
measure ALLCONNECTED (two to four atom expressions) "labelFormat"
```

Show the distance, angle, or dihedral angle formed by ALL atoms in the first expression with ALL atoms of each additional atom expression, provided they form a connected set. The format is optional.

```
measure DELETE
```

Deletes all measurements.

```
measure DELETE (integer)
```

Deletes a specific measurement, in order of their creation, starting with 1.

```
measure DELETE (two to four atom expressions)
```

Deletes all matching distance, angle, or dihedral angle measurements that are currently defined based on the atom expressions.

```
measure RANGE (decimal) (decimal) ALL|ALLCONNECTED|DELETE (two to four atom expressions, each in parentheses)
```

Adding RANGE and two decimal numbers modifies the above commands to limit the measurements created or deleted to only those within this specific range of values in Angstroms (distance) or degrees (angles). The word "RANGE" itself is optional but recommended.

Examples:

See [examples: 11/measure.htm](#)

See also:

[axes](#) [boundbox](#) [unitcell](#)

[top](#) [search](#) [index](#)

## meshribbon or meshribbons

A mesh ribbon is similar to a strand, but is more the quality of a loosely woven fabric.

```
meshribbon ON/OFF {default: ON}
```

```
meshribbon ONLY
```

Turns meshribbon rendering on and all other rendering off.

```
meshribbon [mesh-ribbon-radius]
```

Starting with Jmol 12.0, a negative number also implies **ONLY**.

where

**[mesh-ribbon-radius]** is the overall radius of the mesh ribbon -- (decimal, <=4.0)

Examples:

See [structure.htm](#)



See also:

[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [geoSurface](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

[↑top](#) [🔍search](#) [📖index](#)

## message

Sends a string of text to the messageCallback function (for the applet). The [set MessageCallback](#) command can be used to set this JavaScript function. The message is also entered into the scriptStatus queue. Variable values and math expressions can be included in messages and echos. These use the following syntax: **message my variable = @{variablename}**, provided a variable with that name exists. Note that the braces are required -- message my variable = @x will simply deliver "my variable = @x". The @{...} syntax is also supported in this context: **message the fraction 3 / 2 is @{ 3 / 2 }**. Basically, any expression that can appear in a [print](#) command can appear in @{...} in a message or echo. For example: **message @{ {\_N and connected(2,\_H)}.size } NH2 groups are present**. If in an echo that is displayed, these variables are updated dynamically. Since the Jmol application can be run "headless" -- with no display -- using the -ions set of flags, you can use a designed message command to deliver model information to other programs, however the [print](#) command has a simpler syntax and is more flexible.

message (string)

See also:

[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [case default echo for if reset set switch while](#)

[↑top](#) [🔍search](#) [📖index](#)

## minimize or minimization

(v. 11.6 -- NEW)

Minimizes the structure using an adapted UFF force field (Rappe, A. K., et. al.; J. Am. Chem. Soc. (1992) 114(25) p. 10024-10035). Generally the minimization runs in its own thread. If it is desired to wait until the process has completed prior to continuing, use **set useMinimizationThread false**.

minimize

Carries out a minimization using the default convergence criterion specified by [set minimizationCriterion](#) or until the maximum number of steps specified by [set minimizationSteps](#) is reached. If [set minimizationRefresh](#) is TRUE, then the model refreshes after each minimization step, producing a sort of animated minimization.

minimize ADDHYDROGENS

First adds hydrogen atoms, then minimizes the structure.

minimize CANCEL

Stops a running minimization.

minimize CLEAR

Stops a running minimization and clears all constraints

minimize CONSTRAINT CLEAR

Clears any current constraints.

minimize CONSTRAINT (two to four atom expressions) (decimal)

Constrains the distance, angle, or dihedral angle involving two to four atoms. If a given atom expression involves more than one atom, only the first atom in the set is made part of the constraint.

minimize CRITERION

Overrides the setting of [minimizationCriterion](#) for this minimization.

minimize ENERGY

Do no minimization -- just calculate the energy

minimize SELECT [\[atom-expression\]](#)

Minimizes only the specified atom set. (Any previous or additional setting of FIX is respected.)

minimize STEPS (integer)

Overrides the setting of [minimizationSteps](#) for this calculation.

minimize STOP

Same as CANCEL.

minimize FIX [\[atom-expression\]](#)

Specifies an atom set to keep in fixed position during this minimization or future minimizations.

where

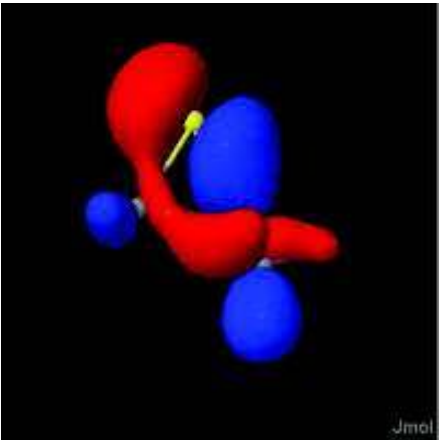
[\[atom-expression\]](#) is any [expression](#) that evaluates to a set of atoms

[↑top](#) [🔍search](#) [📖index](#)

## mo

(v. 11.0 -- NEW, 11.2 adds HOMO and LUMO)

The **MO** (molecular orbital) command displays molecular orbitals contained in a variety of file formats. One simply loads the file, then selects which orbital to display with **MO n** where "n" is the number of the molecular orbital to display. (With some file formats, you may need to use the [frame or model](#) command to call up the specific model having the MOs.) **MO NEXT** and **MO PREVIOUS** allow for quick browsing. Several adjustments can be made, including MO COLOR (allows for different colors for the positive and negative lobes), MO CUTOFF (smaller cutoff value gives larger orbitals) and MO RESOLUTION (higher resolution gives cleaner curves but slows surface generation). In addition, MO PLANE creates a planar slice through the orbital. Option changes take effect immediately with the currently displayed orbital and stay in effect for later MO commands until a new file is loaded.



Note that ONE molecular orbital is allowed per model (that is, per frame). The [isosurface](#) command can be used for more advanced molecular orbital display options or for displaying several planes or orbitals simultaneously. Default rendering prior to Jmol 11.1.28 is FILL; starting with Jmol 11.1.28 MESH NOFILL FRONTONLY

mo ON/OFF {default: ON}

Turn on/off the molecular orbital.

mo (integer)

Selects the specific molecular orbital to display, starting with the number 1.

mo COLOR [\[RGB-color\]](#)

Colors the orbital one specific color

mo COLOR [\[RGB-color\]](#) [\[RGB-color\]](#)

Colors regions where the wave function is less than zero the first color; regions where the wavefunction is greater than zero the second color.

mo CUTOFF (decimal)

Sets the cutoff value for the isosurface that defines the orbital. This number may be dependent upon the computational package used to generate the orbitals. Values in the range 0.005 - 0.05 may need to be experimented with in order to get the best display. Values closer to zero lead to surfaces further from the atoms (larger orbitals). Both positive and negative cutoffs are allowed. A positive number indicates to use both positive and negative cutoffs. Adding an explicit "+" sign before the number indicates that only the positive part of the surface is desired.

mo DELETE

Delete the molecular orbital.

mo HOMO [+/-n]

Selects the highest doubly- or singly-occupied molecular orbital, optionally an orbital +/-n from this orbital. (Only applicable to data sets that have orbital occupancies indicated in the file.)

mo LUMO [+/-n]

Selects the lowest unoccupied molecular orbital, optionally an orbital +/-n from this orbital. (Only applicable to data sets that have orbital occupancies indicated in the file.)

mo MODEL n or x.y

Specifies the model for this command; must be followed by an orbital specification such as **HOMO** or **12**.

mo NEXT  
Displays the next MO in the file using the same characteristics as the currently displayed one.

mo NOPLANE  
Goes back to full orbital display rather than a planar slice.

mo PLANE plane\_expression  
Indicates that what is desired is a color-mapped planar slice through the orbital using one of the ways of expressing a [plane](#).

mo PREVIOUS [\[RGB-color\]](#)  
Displays the previous MO in the file using the same characteristics as the currently displayed one.

mo RESOLUTION (decimal)  
Sets the resolution of the isosurface rendering in "points per Angstrom". Higher resolution leads to smoother surfaces and more detail but carries the penalty of slower surface generation. Typical values for molecular orbitals are 4-10 points per Angstrom.

mo TITLEFORMAT "format"  
Sets the format of the orbital title appearing in the upper left corner of the applet. Special format characters include:

%E	energy
%F	filename
%I	molecular orbital number
%M	model number
%N	total number of molecular orbitals
%O	occupancy
%S	symmetry
%U	energy units
	(vertical bar) new line
?	(at the beginning of a line) indicates to disregard line if no data for that line are present in the file

If a formatted item is not indicated in the file, then it is left blank. The default title is "%F | Model %M MO %I/%N | Energy = %E %U | ?Symmetry = %S | ?Occupancy = %O". The command **MO titleFormat ""** may be used to show no title.

where  
**[RGB-color]** is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]

Examples:

See [examples-11/mo.htm](#)

See also:  
[\[plane expressions\]](#) [\[isosurface\]](#) [\[caoCartoon\]](#) [\[pmesh\]](#) [\[polyhedra\]](#)

## model or models

Same as the [frame](#) command. See also [set backgroundModel](#).

Examples: [in new window using cyclohexane\\_movie.xyz](#)

```
model 1
model NEXT
model PREVIOUS
model 0;select *;wireframe 0.1;spacefill 0.2
anim on
model 0;select *;wireframe off;spacefill off;
select 1.1 # in Jmol10 use */1
```

```
wireframe 0.1;spacefill 0.2;color atoms red;
select 1.35;wireframe 0.1;spacefill 0.2;color atoms blue
```

See [animation.htm](#)

See also:  
[animation frame invertSelected move moveto rotateSelected set \(misc\) spin translate translateSelected zoom zoomto](#)

## move

The move command provides powerful animation capabilities. It allows you to specify rotations, zooming, and translations to be performed in a specified period of time. xRot, yRot, and zRot are rotations about the cartesian axes in degrees. Zoom specifies a zoom factor, xTrans, yTrans, and zTrans are translations in the range -100 to 100. If you do not know what slab is, just put in a zero. see the slab command for more information. This command has been superceded by the **moveTo** command.

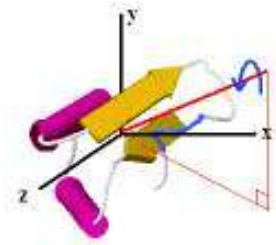
move [\[x-rotation\]](#) [\[y-rotation\]](#) [\[z-rotation\]](#) [\[zoom-factor\]](#) [\[x-translation\]](#) [\[y-translation\]](#) [\[z-translation\]](#) [\[slab-cutoff\]](#) [\[seconds-total\]](#) [\[move-frames-per-second\]](#) {default: 30} [\[maximum-acceleration\]](#) {default: 5}

where  
**[x-rotation]** is the degrees of rotation about x -- (integer)  
**[y-rotation]** is the degrees of rotation about y -- (integer)  
**[z-rotation]** is the degrees of rotation about z -- (integer)  
**[zoom-factor]** is a scaling factor -- (integer)  
**[x-translation]** is the distance offset along x -- (integer)  
**[y-translation]** is the distance offset along y -- (integer)  
**[z-translation]** is the distance offset along z -- (integer)  
**[slab-cutoff]** is the cutoff for the slab display -- (integer)  
**[seconds-total]** is the amount of time to wait -- (decimal)  
**[move-frames-per-second]** is the frames per second to move -- (integer)  
**[maximum-acceleration]** is the maximum acceleration -- (integer)

See also:  
[animation frame invertSelected model moveto rotateSelected set \(misc\) spin translate translateSelected zoom zoomto](#)

## moveto

The moveto command rotates the molecule to a predefined orientation. Two formats can be used. In each, the first (optional) parameter specifies the number of seconds during which the molecule should rotate smoothly from the current orientation to the new orientation. A 0 for this first parameter specifies an instantaneous reorientation. Starting with Jmol 12.0, if the axis is {0 0 0} and the degrees are 0, then the molecule is not reoriented during the operation. In conjunction with "show orientation" this command allows reading and restoring specific user-specified orientations.



moveto timeSeconds FRONT|BACK|LEFT|RIGHT|TOP|BOTTOM

A simple use of moveTo just has six optional directions.

moveto timeSeconds {x y z} degrees zoomPercent transX transY {x y z} rotationRadius navigationCenter navTransX navTransY navDepth

In the second option, the second parameter is a coordinate {x, y, z} defining the axis relative to the default orientation about which the molecule should be rotated. The third parameter is the counterclockwise (right-hand) rotation in degrees about this axis. "moveto 0 {0 0 1} 0" rotates the model to the default orientation (equivalent to "reset"). If the angle parameter is 0 but any one of x, y, or z is nonzero, then no reorientation occurs (because the axis has been specified, but the rotation is 0 degrees). Following these parameters is the zoom setting in percent, the X- and Y-positions of the rotation center on the screen, as percent of width and height, respectively. The actual molecular coordinate of the rotation center along with the rotation radius (which determines the magnification associated with **zoom 100**) are next. The final parameters define the navigation center molecular coordinate, its X- and Y- position on the screen in percent, and the depth of the navigation point in percent of model depth (100 = front, 0 = rear). In conjunction with "show/save/restore orientation" this command allows reading and restoring specific user-specified orientations.

moveto timeSeconds {x y z} degrees 0 transX transY (atom expression) 0 zoomAdjustment navigationCenter navTransX navTransY navDepth

If the zoom setting prior to translation positions is 0, and an atom expression is used for the point, then the moveTo can be designed to automatically zoom to the scale that would fill the screen with that set of atoms. The optional zoom adjustment is in the form +n, -n, \*n, or /n, as for [zoomTo](#).

moveto timeSeconds {x y z} degrees (atom expression) 0 zoomAdjustment navigationCenter navTransX navTransY navDepth

If no translation is involved, then there is also no need for the zoom setting of 0 prior to the atom expression.

moveto STOP

Stops an asynchronous moveto operation that has been started after setting **set waitForMoveTo FALSE**. [Jmol 12.0]

Examples: [in new window using 1crn.pdb](#)

```
moveto 0 1 0 0 -90; #view from top
moveto 0 0 1 0 90; #view from right
moveto 0 0 1 0 -90; #view from left
moveto 2 1 0 0 90; #view from bottom with smooth 2-second transition
moveto 0 0 0 0 200; #default orientation with instantaneous transition and a zoom of 200%
```

See [moveto.htm](#)

See also:

[animation frame](#) [invertSelected model](#) [move](#) [navigate](#) [rotateSelected](#) [set \(misc\)](#) [set \(navigation\)](#) [spin](#) [translate](#) [translateSelected](#) [zoom](#) [zoomto](#)

[↑top](#) [🔍search](#) [📖index](#)

## navigate or navigation

The **navigate** command allows exploring of the model from the perspective of an airplane capable of flying through the model. In each case, an optional time in seconds determines the time to reach the objective. If no time is given, the change occurs over two seconds. (In the case of PATH, this is the time to each point along the path, not the total path.) Subcommands can be grouped in the same command by separating them with "/", for example: **navigate 2 DEPTH 30 / 5 ROTATE 180 / DEPTH 20 / TRANSLATE X 10**.

navigate timeSeconds CENTER {x y z}

Bring the observer to the specified molecular coordinate, which may be fractional in the case of crystal structures.

navigate timeSeconds CENTER { atom expression }

Bring the observer to the geometric center of the set of atoms specified in parentheses.

navigate timeSeconds CENTER Subject

Bring the observer to the geometric center of the points corresponding to the indicated [object](#).

navigate timeSeconds DEPTH percent

Bring the observer forward or backward to the depth indicated as a percent from back (0) to front (100) of the model. Values can be negative, in which case until a rotate command is given, only screen background will be seen. Values greater than 100 put the observer outside the model.

navigate timeSeconds PATH Subject

Follows the Hermite path given by the specified [draw](#) object (such as a line, curve, or arrow). This allows easy dynamic development of paths using a predefined draw object followed by **set picking draw** and **show draw**.

navigate timeSeconds PATH (any combination of coordinates, atom expressions, and objects)

Follows a Hermite path defined by a set of nodes expressed in terms of any combination of atom sets, draw objects, or coordinates.

navigate timeSeconds QUATERNION { quaternion }

Reorients the model to the orientation specified by the quaternion given in {x y z w} format.

navigate timeSeconds QUATERNION MOLECULAR { quaternion }

Reorients the model to the orientation specified by a molecular frame quaternion, as, for example, retrieved by **q = quaternion({resno=30})** and then used in **moveto quaternion MOLECULAR @q**. Note that because this quaternion refers to the rotation required to transform the reference frame to the specified molecular frame, in order to "move to" this orientation, one must use its inverse (bring the molecular frame to the reference frame, **q\_orientation = !q\_molecular**). The keyword MOLECULAR simply applies this inversion.

navigate timeSeconds QUATERNION { atom expression }

Reorients the model to the orientation specified by the quaternion specified by the atom expression and the current setting of **quaternionFrame**.

navigate timeSeconds ROTATE X degrees

Rotates around the X axis at the navigation center.

navigate timeSeconds ROTATE Y degrees

Rotates around the Y axis at the navigation center.

navigate timeSeconds ROTATE Z degrees

Rotates around the Z axis at the navigation center.

navigate timeSeconds TRACE (atom expression)

Navigates along the trace of a protein or nucleic acid. The "ride" can be changed depending upon the settings of [set sheetsmoothing](#) and [set tracealpha](#).

navigate timeSeconds TRANSLATE x.xx y.yy

Translates the navigation screen offset to the specified positions expressed as percent of width (X) and height (Y) of the applet/application window.

navigate timeSeconds TRANSLATE X x.xx

Translates the navigation screen offset horizontally by the specified percent of applet/application window width.

navigate timeSeconds TRANSLATE Y y.yy

Translates the navigation screen offset vertically by the specified percent of applet/application window height.

navigate timeSeconds TRANSLATE {x y z}

Translates the navigation screen offset to the screen position corresponding to the given molecular coordinate.

navigate timeSeconds TRANSLATE (atom expression)

Translates the navigation screen offset to the screen position corresponding to the geometric center of the specified atoms.

navigate timeSeconds TRANSLATE Subject

Translates the navigation screen offset to the screen position corresponding to the center of the vertices of the specified draw object.

See also:

[moveto](#) [set \(navigation\)](#) [zoomto](#)

[↑top](#) [🔍search](#) [📖index](#)

## parallel/process (v. 12.0)

Jmol 12.0 introduces parallel processing for Jmol. Jmol 12 will be able to use multiple processors on a multiple-CPU machine. Basically what you can do is to tell Jmol which statements in a script you want to run in parallel, and it will do that. The way this is done is to create a function using the keyword **PARALLEL** in place of the keyword **function**. Within that block of code, any group of commands surrounded by **PROCESS{ }** will be collected and run in parallel just before Jmol returns from the function. Any commands NOT within these sets will be run BEFORE any **PROCESS** commands. For example:

```
parallel twosurfaces(model1, model2) {
  var x = 1
  process {
    isosurface s1 model @model1 molecular; color isosurface red
  }
  process {
    isosurface s2 model @model2 molecular; color isosurface green }
  x = 2
}

load files "1crn.pdb" "1blu.pdb"
twosurfaces("1.1", "2.1")
frame *
```

In this case, the variable x will be 2 BEFORE the isosurfaces are created. See also [multi-mo.txt](#) and [multi-surface.txt](#), and [multiProcessTest.txt](#). You can selectively turn on and off the use of multiprocessors using [set multiProcessor](#). If this setting cannot be set

true, then it means you do not have a multiprocessor machine. Not all processes will work; currently the only implemented parallel processes are for isosurfaces and molecular orbitals. The parallel capability of Jmol should be considered experimental at this time.

[top](#) [search](#) [index](#)

## pause or wait

Pauses script execution until [resume](#), [step](#), [quit](#), or [exit](#) is issued. Any text on the command line after **pause** is sent to the user as a message when the script pauses. During the paused condition, commands can be entered from the Jmol application console, and they will be executed. The next command to be executed can also be checked by issuing ? from the console.

pause message

See also:

[delay](#) [exit](#) [goto](#) [loop](#) [quit](#) [resume](#) [step](#)

[top](#) [search](#) [index](#)

## plot

(v. 12.0 replaces [quaternion](#) and [ramachandran](#) commands, and adds new property options)

Jmol can create a variety of Ramachandran plots and quaternion maps. In addition, Jmol 12.0 introduces the capability to quickly generate simple plots relating two or three atom properties. A new frame is created. In the case of property and Ramachandran plots, this frame has its own independent orientation; in the case of quaternion maps, rotation of the map is synchronized with rotation of the model. Only one data frame may be visible at a time. Also related to this command for quaternions and Ramachandran plots is the [draw](#) command, which allows depicting of these measures on the model itself.

plot PROPERTIES property1 property2

Creates a 2D plot relating two atom properties for the currently selected atom set. For example: **select \*;plot properties atomno temperature**. Additional optional parameters MIN {x y z} and MAX {x y z} may follow. Setting these values sets the scale of the graph within Jmol and truncates data values outside of this range. (The z value in this case is ignored.)

plot PROPERTIES property1 property2 property3

Creates a 3D plot relating two atom properties for the currently selected atom set. For example: **select \*.CA;plot properties phi psi resno**. Additional optional parameters MIN {x y z} and MAX {x y z} may follow. Setting these values sets the scale of the graph within Jmol and truncates data values outside of this range.

plot QUATERNION w, x, y, or z

Creates the quaternion representation of the protein or nucleic acid in the three dimensions not given by the specified axis.

plot QUATERNION a,r DIFFERENCE

Creates the quaternion difference representation of the protein or nucleic acid in "w" projection. "a", for absolute, produces a visualization of  $q[i] / q[i-1]$ , which defines the helical axis of a helix or beta-pleated sheet; "r", for relative, produces a visualization of  $q[i-1] \setminus q[i]$ , which defines the relative rotation of the (i)th residue from the perspective of the (i-1)th residue, defining the pitch of the helix. The default visualization for "r difference" (utilizing **set quaternionFrame "c"**) for a typical protein is an ellipse having a major axis tilted at 36 degrees from the quaternion X axis and minor axis along the quaternion Z axis. This ellipse represents a composite rotation from one amino acid to the next involving a rotation about the  $q[i]$  frame Z axis (perpendicular to the N-CA-C plane) of approximately 180 - 110 degrees ((180-110)/2 = 36 degrees), followed by a rotation about the  $q[i]$  frame X axis (CA-C bond) of 180 +  $\psi[i-1] + \phi[i]$  degrees.

plot QUATERNION a,r DIFFERENCE2

Creates a visualization of  $dq[i+1] / dq[i]$ , where **dq** is defined as  $q[i+1] / q[i]$  for "a" or  $q[i] \setminus q[i+1]$  for "r". In the case of "r difference2", for most amino acids this is a point relatively close to the quaternion x axis.

plot RAMACHANDRAN

Creates a Ramachandran plot for a the currently displayed protein model.

plot RAMACHANDRAN r

Creates a "relative" Ramachandran plot for a the currently displayed protein model with the third axis being theta, a measure of "straightness" as calculated using a Ramachandran angle approximation:  $\text{straightness}[i] = 1 - \text{acos}(\text{abs}(\cos(\text{theta}[i] / 2))) / 90$ , where i refers to a residue and theta is the angle associated with the relative quaternion second derivative  $dq[i]dq[i-1]$ . Theta can be approximated using Ramachandran angles for "C" straightness simply as  $\text{theta} = \text{approx} - \psi[i] - \psi[i-1] + \phi[i+1] - \phi[i]$  (Hanson and Kohler, unpublished results) and for "P" straightness as follows: If  $\text{deltaPhi} = \phi[i+1] - \phi[i]$ ,  $\text{deltaPsi} = \psi[i+1] - \psi[i]$ , and  $\alpha = 70^\circ$ , then  $\cos(\text{theta}/2) = \text{approx} - \cos(\text{deltaPsi}/2)\cos(\text{deltaPhi}/2) - \sin(\alpha)\sin(\text{deltaPsi}/2)\sin(\text{deltaPhi}/2)$ . (Hanson and Braun, unpublished results.) Residues close to the phi-psi plane have low values of theta and thus high values of straightness, meaning they are in a region of relatively high structural regularity ---

usually helices or sheets.

See also:

[quaternion](#) [ramachandran](#) [undefined](#)

[top](#) [search](#) [index](#)

## pmesh

Starting with Jmol 11.8, the pmesh command is deprecated. See [isosurface](#) for information about the pmesh file format options.

pmesh ID [object ID]

Selects a specific pmesh (or all pmeshes) for subsequent color commands.

pmesh ID [object ID] ON/OFF {default: ON}

Turn on/off the specified mesh.

pmesh ID [object ID] DELETE

Delete the specified mesh.

pmesh ID [object ID] "filename"

Loads the specified pmesh file, optionally assigned id pmeshID. Double quotes are required. A default directory can be set using [set defaultDirectory](#), and for applets a proxy server can be set for non-host file loading using [set appletProxy](#).

pmesh ID [object ID] DOTS or NODOTS {default: NODOTS} "xyz.pmesh.gz" {default: current}

Controls whether or not dots are shown at the polygon vertices.

pmesh ID [object ID] FILL or NOFILL {default: FILL} "xyz.pmesh.gz" {default: current}

Controls whether the polygons are filled (the default).

pmesh LIST

Lists all pmesh objects

pmesh ID [object ID] MESH or NOMESH {default: NOMESH} "xyz.pmesh.gz" {default: current}

Controls whether the edges of the polygons are drawn.

Examples: [in new window using caffeine.xyz](#)

```
pmesh myPlane "10x10pmesh.txt"
# load a pmesh with ID myPlane
color pmesh translucent yellow
# make it translucent yellow
pmesh myWave "wave.pmesh"
# load another pmesh, with ID myWave
pmesh dots
# turn on dots on all loaded pmeshes
pmesh myWave
# select pmesh myWave
color pmesh white
# color it white
pmesh myPlane mesh
# only display the mesh for myPlane
pmesh myPlane nodots
# no dots for myPlane
```

See [examples-11/pmesh.htm](#)

See also:

[\[plane expressions\]](#) [isosurface](#) [lcaoCartoon](#) [mo](#) [polyhedra](#)

[top](#) [search](#) [index](#)

## polyhedra

Polyhedron construction  
[number of vertices]  
[basis]  
[selection sets]

[display options]  
Polyhedron modification

Jmol will form a wide variety of polyhedral structures. The **polyhedra** command can be used for either construction of new polyhedra or modification of existing polyhedra.

#### Polyhedron construction [back](#)

Used for construction of new polyhedra, parameters fall into five subsets:

**polyhedra** [number of vertices] [basis] [selection sets] [display options]

In order to construct polyhedra, at least one of the first two of these subsets must be present -- the number of vertices or the basis. In addition, some display options (ON, OFF, DELETE -- see below) are incompatible with polyhedron construction.

[number of vertices] [back](#)

Polyhedra involve a central atom and from 3 to 20 outer "vertex" atoms. The number of required vertex atoms is specified by an integer (**polyhedra 6**). More than one number can be specified, separated by a comma or space character (**polyhedra 4,6**). If no number of vertices is indicated, any number from 3 to 20 is assumed.

[basis] [back](#)

Polyhedra can be formed based either upon the number of atoms bonded to the central atom (**polyhedra 4 BONDS**) or upon the number of atoms within a specified distance from the central atom (**polyhedra 4,6 RADIUS 2.0**). (The keyword "RADIUS" is optional, but if it is absent the radius must be indicated as a decimal number, not an integer, so as to distinguish it from a vertex count.) If no radius is indicated, BONDS is assumed. If both BONDS and RADIUS are specified, then polyhedra of the given type(s) are constructed only to connected atoms that are within the specified radius of the central atom.

[selection sets] [back](#)

Potential polyhedra centers and vertex atoms are specified in the form of a standard Jmol embedded [atom expression](#), such as {titanium} or {atomno<12 and not nitrogen} and as such must be specified in parentheses. The first set specifies the centers; the second set specifies the vertex atoms. The optional keyword TO can precede the vertex set for clarity, as in **polyhedra 4,6 BONDS {titanium} TO {oxygen or nitrogen}**. If no atom sets are designated, the assumption is "(selected) TO (\*)". A single designation indicates centers, with "TO (\*)" implied. If only TO and an atom set is specified, then the centers are taken as the currently selected set of atoms.

[display options] [back](#)

Three sets of display options can be included when constructing polyhedra.

- Polyhedra can be displayed either as traditional "FLAT" faces (the default) or as "COLLAPSED" faces, which display more clearly the central atom. An empirically adjustable parameter, **distanceFactor** can be set to a higher value to include more faces in a polyhedron if some do not form. Its default value is 1.85. For collapsed polyhedra, the distance in angstroms from the central atom to the point of collapse can be specified using **faceCenterOffset=x.x** where x.x is a distance in Angstroms.
- Polyhedra can be displayed either with no edges (NOEDGES, the default), or with a thick line on all edges (EDGES), or on just the front edges (FRONTEDGES). The front-edge-only option is only meaningful when the polyhedra are translucent (see below).
- Polyhedra can be colored by indicating a valid color (e.g. **red** or **yellow**) or a hexadecimal RGB color triple (e.g. [xFFF000] or [xFFFF00]). The keywords TRANSLUCENT and OPAQUE can also be used. Alternatively, after polyhedra are created they can be colored using the [color polyhedra](#) command.

#### Polyhedron modification [back](#)

The **polyhedra** command can also be used to modify already-constructed polyhedra. Used in this way, the command can take only one atom set expression, and it must not indicate the number of vertices or the basis. If the atom set expression is omitted, "(selected)" is assumed.

To turn on, turn off, or delete selected polyhedra, use **polyhedra ON**, **polyhedra OFF**, or **polyhedra DELETE**, respectively. Any of the display options (FLAT, COLLAPSED, EDGES, FRONTEDGES, or NOEDGES) can also be similarly modified. Colors or translucency, however, cannot be applied this way. To color a set of polyhedra that are already formed or to make them translucent, first select the set of centers of the polyhedra to modify, then use the [color polyhedra](#) command.

Examples: [in new window using caffeine.xyz](#)



```
select *;polyhedra {*} DELETE;polyhedra 4 BONDS; color polyhedra grey
select atomno=19;polyhedra (*) DELETE;polyhedra 4 RADIUS 2.0 ;color polyhedra yellow
polyhedra {*} DELETE;polyhedra 4 RADIUS 2.0 {*} COLLAPSED #all three
polyhedra {*} DELETE;polyhedra 4 RADIUS 2.0 {*} TO {not within (1.1225, carbon)} #note how this disallows one of the three
select *;color polyhedra translucent; # now we can see the carbons inside
polyhedra {*} EDGES; # highlight the edges
select *; color polyhedra translucent orange;
polyhedra {*} OFF;
polyhedra {*} ON;
polyhedra {*} DELETE;
```

Examples: [in new window using kaolin.mol](#)



```
# build tetrahedrons around silicon
polyhedra BONDS {silicon}
# make some of them green
select atomno<50; color polyhedra translucent green
```

```
# delete some of them
polyhedra {atomno>75 and atomno<100} DELETE
# now build octahedrons where oxygens are within 2.0 Angstroms of a central aluminum atoms
polyhedra RADIUS 2.0 {aluminum} FRONTEDGES
select aluminum and atomno > 75; color polyhedra red
```



See [examples-11/poly.htm](#)

See also:

[\[plane expressions\]](#) [isosurface](#) [lcaoCartoon](#) [mo](#) [pmesh](#)

[↑top](#) [🔍search](#) [📖index](#)

## PRINT

Prints a [Jmol math](#) expression to the console window, status message callback function, and jmolScriptWait return value.

Note:

The PRINT command does not require @{ ... } around Jmol math expressions.

[↑top](#) [🔍search](#) [📖index](#)

## PROMPT

(v. 12.0 -- new)

The **prompt** command pauses a script until the user presses OK. See also the [prompt\(\)](#) function.

prompt

If no parameter is given, displays a trace of the script stack leading to this command. This may be useful for debugging script files.

prompt "message"

Displays the message and waits for the user to press OK.

Note:

The PROMPT command does not require @{ ... } around Jmol math expressions.

[↑top](#) [🔍search](#) [📖index](#)

## quaternion or quaternions (v. 12.0 (deprecated))

In Jmol 12.0, this command has been superceded by the [plot QUATERNION](#) command.

See also:  
[plot ramachandran](#)

[top](#) [search](#) [index](#)

## quit

When the [set scriptQueue](#) is turned on, each script waits for the previous to complete. When a LOOP command is involved and the script queue is enabled, the only way to interrupt the looping script is with another script. So, to account for this issue, the roles of **quit** and **exit** have been expanded. Either **quit** or **exit** at the very beginning of a script command halts any previous still-running script. Processing then continues with the second command on the line. Anywhere else in the command, **quit** and **exit** abort that script.

See also:  
[delay exit goto loop pause resume set \(files and scripts\) step](#)

## ramachandran or rama (v. 12.0 (deprecated))

In Jmol 12.0, this command has been superceded by the [plot RAMACHANDRAN](#) command.

See also:  
[plot quaternion](#)

[top](#) [search](#) [index](#)

## refresh

Forces a screen repaint during script execution. (Unnecessary, and thus deprecated.)

See also:  
[define initialize reset restore save zap](#)

[top](#) [search](#) [index](#)

## reset

Resets all model orientation or the value of a variable

reset  
Resets all models to their original position: zoom 100; center; translate x 0; translate y 0. Note that if the [invertSelected](#), [rotateSelected](#), [translateSelected](#) commands have been given, these changes are not reset, because these changes are recorded in the underlying coordinate set. If it is desired to save and restore atom positions after moving atoms relative to each other, the following commands may be issued: select \*;translateSelected {0 0 0};save state;...(move atoms here)...;restore state.

reset AROMATIC  
Resets all aromatic bonds to the type AROMATIC if they are AROMATICSINGLE or AROMATICDOUBLE. Used in conjunction with [connect](#) and [calculate aromatic](#) (Jmol 11.4).

reset FUNCTIONS  
Deletes all user-defined functions.

reset variableName  
Resets the variable with the given name to the "unset" state.

reset ALL  
Deletes all user-defined variables.

See also:  
[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [case default define echo for if initialize message refresh restore save set switch while zap](#)

[top](#) [search](#) [index](#)

## restore

Restores information saved using the [save](#) command.

restore BONDS saveName  
Restores bonding information changed via the [connect](#) command; if no save name is given, then the last-saved information is restored.

restore ORIENTATION saveName timeSeconds  
Restores a previously saved orientation. If no save name is given, then no time in seconds can be given, and the last-saved orientation is restored immediately. If a save name is given, then the number of seconds over which the restoration should be done may be given as well.

restore SELECTION saveName  
Restores a previously saved selection. If no save name is given the last-saved selection is restored. If no saved selection of this name is found, then the action is the same as **select none**.

restore STATE saveName  
Retores a previously saved state of the applet. Some of the more complex objects, such as dipoles, isosurfaces, IcaoCartoons, and molecular orbitals are not saved.

See also:  
[define initialize refresh reset save zap](#)

[top](#) [search](#) [index](#)

## restrict

Selects the atoms identified by the [atom expression](#) and deletes all characteristics of all atoms and bonds that are outside the selection set by setting those characteristics "off". For wireframe and backbone, **restrict** unconditionally follows [set bondmode OR](#), ignoring the current bondmode setting, thus removing any bonds or backbone rods involving any atoms not in the restricted set. The **restrict** command is a holdover from RasMol, kept in Jmol for compatibility only. The [display](#) command is far more flexible, as it preserves the shape characteristics (size, color, translucency) of the hidden atoms rather than simply deleting the shapes.

restrict {default: ALL}  
Restricts to all atoms;possibly not H atoms.

restrict [\[atom-expression\]](#)  
Restricts atoms based on an [atom expression](#).

restrict BONDS [\[atom-expression\]](#)  
Restricts atoms based on an [atom expression](#) while respecting the setting of **bondmode**. (Jmol 12.0)

where  
**[atom-expression]** is any [expression](#) that evaluates to a set of atoms

See also:  
[display hide select subset](#)

## resume

Resumes script execution after a [pause](#).

resume

See also:  
[delay](#) [exit](#) [goto](#) [loop](#) [pause](#) [quit](#) [step](#)

## RETURN

Returns from a [function](#) or a script being run with the [script](#) command. In the case of a function return, may include an optional return value.

return returnValue

**Note:**  
 The RETURN command does not require @{ ... } around Jmol math expressions.

See also:  
[break](#) [case](#) [catch](#) [continue](#) [default](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [switch](#) [try](#) [var](#) [while](#)

## ribbon or ribbons

Ribbons offer a representation the protein backbone or nucleic acid helix using a flat band. For proteins, control points are chosen to be the center of the peptide bond, and the ribbon is drawn in the direction of the carbonyl oxygen (thus roughly defining the peptide planes). For nucleic acids, the control points are the midpoints between adjacent backbone phosphorus atoms, and the ribbon is drawn in the direction of the C6 carbon. A hermite curve is used.

ribbon ON/OFF {default: ON}  
 ribbon ONLY

Turns ribbon rendering on and all other rendering off.

ribbon [\[ribbon-radius\]](#)

Normally, ribbons vary in width according to the amino acid atom positions. This command sets the width of the ribbon to be a constant value (a decimal, in Angstroms). Starting with Jmol 12.0, a negative number also implies **ONLY**.

where  
[\[ribbon-radius\]](#) is half of the overall width of the ribbon -- (decimal, <=4.0)

Examples:

See [structure.htm](#)

See also:  
[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

## rocket or rockets

[↑](#) [top](#) [?](#) [search](#) [i](#) [index](#)

[↑](#) [top](#) [?](#) [search](#) [i](#) [index](#)

[↑](#) [top](#) [?](#) [search](#) [i](#) [index](#)

[↑](#) [top](#) [?](#) [search](#) [i](#) [index](#)

Creates a crude "rocket" cartoon. See also [cartoon](#) in association with [set cartoonRockets](#) for a more precise cartoon with rockets. Jmol 11.4 introduces the [set rocketBarrels](#) option, which removes the arrow heads from rockets.

rocket ON/OFF {default: ON}

rocket ONLY

Turns rocket rendering on and all other rendering off.

rocket [\[rocket-radius\]](#)

Starting with Jmol 12.0, a negative number also implies **ONLY**.

where  
[\[rocket-radius\]](#) is half of the overall width of the rocket barrel -- (decimal, <=4.0)

Examples:

See [structure.htm](#)

See also:  
[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

## rotate

(v. 12.0 -- adds COMPARE, HELIX, SYMOP, TRANSLATE options)

The **rotate** and **spin** commands allow single angle rotation or continued rotation (spinning) in one of two specific frames -- the standard "fixed" applet window frame or the internal "molecular" model frame (the axes in the data file) -- around any pair of points of one or more of the following types: absolute coordinate positions set in braces as {x, y, z}, the geometric center of a subset of atoms in the model specified in an atom set in parentheses, for example **{atomno < 10 and not oxygen}**, or the geometric center of a previously defined draw object, indicated by the drawing object name preceded by a dollar sign (for example, **\$line1**). In addition, rotation may be specified to be around six standard axes (x, y, z, -x, -y, and -z) as well as several other options. The keyword MOLECULAR may be necessary for explicit model rotation about the internal molecular axes. Defaults are allowed. The default axis for spinning or rotating is the y (vertical) axis; the default angle of rotation is 10 degrees; the default rotation rate is a slow spin of 10 degrees per second. Options may be given in any order and include:

	By itself, the <b>rotate</b> command rotates 10 degrees around the y (vertical) axis in a counterclockwise fashion.
<b>x, y, z, -x, -y, -z</b>	Rotation about one of the six standard axes. The implied axes are the axes of the window, not the molecule.
<b>(decimal)</b>	The first decimal number parameter indicates the number of degrees for a rotation, either positive (right-hand rotation) or negative (left-hand rotation), or (for the TRANSLATION option) the number of Angstroms total; a second decimal number parameter, implies the SPIN option and if positive indicates degrees per second for spinning (or, for the TRANSLATION option, angstroms per second). If the second parameter is negative, -n, the spinning is carried out over n seconds.
<b>{atom expression or point} {atom expression or point}</b>	Rotation about an axis pointing from the first point to the second in right-hand direction. For example, <b>rotate {0 0 0} {1 0 0} 10</b> is the same as <b>rotate x 10</b> , and <b>spin {atomno=1} {atomno=2}</b> spins the model around the axis connecting these two atoms.
<b>\$drawID</b>	the two atoms desired for rotation can be defined based on a <a href="#">drawn</a> object such as a line or vector by specifying the ID of that drawn object.
<b>{3x3 matrix}</b>	A 3x3 matrix can be used to specify the axis and angle of a rotation; MOLECULAR is implied.
<b>{4x4 matrix}</b>	A 4x4 matrix can be used to specify a rotation axis and angle along with a translation; MOLECULAR is implied.
<b>AXISANGLE {x y z}</b>	Rotation about an axis defined as a vector from the origin {0,0,0} to the specified {x,y,z} coordinate. (Jmol 11.6)
<b>AXISANGLE {x y z theta}</b>	Rotation about an axis through {0 0 0} and {x y z} by the specified number of degrees. (Jmol 11.6)
<b>BRANCH {atom 1} {atom 2}</b>	Rotates the atoms that are in the molecular branch containing {atom 2} but not {atom 1} about the axis connecting these two atoms. For example, to spin a methyl group having carbon atom C12 about its connected atom by 360 degrees at a rate of 30 degrees per second, one might use <b>spin [_C and connected(C12)] {C12} 360 30</b> . (Jmol 11.6)

[↑](#) [top](#) [?](#) [search](#) [i](#) [index](#)

<b>COMPARE {atomSet1} {atomSet2 or positions}</b>	For atoms, determines the best-fit correlation between atoms in atomSet1 and atoms in atomSet2 and then rotates and/or translates atomSet1 to align with atomSet2. The two atom sets must have the same number of atoms and must have a direct 1:1 correlations of atoms. (For a more general comparison, see the <a href="#">compare</a> command. An array of points, such as @{ [2.1].xyz.all} may take the place of the second atom set. This allows, for example, saving of a set of coordinates with <b>tempCoords = {2.1}.xyz.all</b> and restoring of those coordinates with <b>ROTATE COMPARE {2.1} @tempCoords</b> some time after a selected atom rotation.
<b>HELIX</b>	For any operation involving a translation (4x4 matrix, COMPARE, SYMOP, or TRANSLATE options) carries out the rotation about the optimum helical path.
<b>INTERNAL</b>	Same as <b>MOLECULAR</b> .
<b>MOLECULAR</b>	In the case of simple axes such as <b>x</b> or <b>y</b> , indicates that the axis is defined in terms of molecular coordinates.
<b>QUATERNION {x y z w}</b>	Rotation about an axis defined by the specified quaternion in the form of a four-vector. For example, to reset the model to the original orientation without changing the zoom, one can use <b>rotate QUATERNION @{ !quaternion(script("show rotation")) }</b> . The keyword QUATERNION is optional.
<b>SELECTED</b>	Rotate only the currently selected atoms.
<b>SPIN</b>	Provide an animation of the rotation. Implied when two decimals -- a number of degrees and a rate -- are given.
<b>SYMOP n</b>	Carry out the symmetry operation number <b>n</b> (or, if <b>n</b> is a negative number, the reverse of that operation) on the currently selected atoms.
<b>TRANSLATE</b>	Add a translation to the animation (providing a screw-like motion) MOLECULAR and SELECTED are implied. If only one point is specified, then this command allows for an animation of a movement of the selected set of atoms along the vector connecting {0 0 0} and that point. In that case, the extent is in Angstroms and rate is in AngstromsPerSecond; otherwise, as for other rotations, extent is in degrees and rate is in degreesPerSecond.

**Examples:**

```
# set to final state
moveto 0.0 { -46 -870 492 142.18} 300.0 0.0 0.0 {61.260506 38.7915 44.659} 50.97134 {0.0 0.0 0.0} 281.9351 50.269466 50.0
qfinal = quaternion(script("show rotation"))
# set to initial state
moveto 0.0 { -142 866 480 107.35} 100.0 0.0 0.0 {61.260506 38.7915 44.659} 50.97134 {0.0 0.0 0.0} 8.481314 102.86942 50.0;
qinitial = quaternion(script("show rotation"))
# calculate steps
dq = qfinal / qinitial / 20
# now loop and save
write image "image0.jpg"
for (i = 1; i <= 20; i = i+1)
  rotate quaternion @dq
  f = "image"+i+".jpg"
  write image @f
end for
```

**See also:**[spin](#)**rotateSelected**

This command takes all the parameters of [rotate](#) but only carries out the operation on the currently selected atoms. In addition, if the keyword SPIN is used, the command starts only the atoms spinning. Note that in its current implementation, this rotate DOES slightly modify atom coordinates; any measurements made after rotating may be off by a fraction of a percent, and continued spinning for a long time may introduced significant errors in relative distances and angles.

**Examples:** [in new window using caffeine.xyz](#)

```
select connected(C19) and not N8;
rotateSelected {N8} {C19} 30 # CH3 rotates 30 degrees around that bond other options might be rotateSelected {0 0 0} {1 1 1} 30, rotateSelected
axisangle {1 1 1 30}, or rotateSelected molecular x 30
```

[↑top](#) [🔍search](#) [📖index](#)**See also:**[animation frame invertSelected model move moveto set \(misc\) spin translate translateSelected zoom zoomto](#)[↑top](#) [🔍search](#) [📖index](#)**save**

Saves a variety of sorts of information for later [restoring](#) either in the current model or a different model.

save BONDS saveName

Saves bonding information, including atom connections, color, width, and visibility. A save name is optional.

save ORIENTATION saveName

Saves a full set of orientation information, including center and position of rotation. A save name is optional but recommended, as it allows restoring of the orientation over a specified number of seconds.

save SELECTION saveName

Saves the current set of selected atoms for restoring later, either for the currently loaded model or for another model. (If used for only the current model, without a **load** command between **save** and **restore**, **save/restore SELECTION xxx** works the same as **define xxx selected/select xxx**.) Note that when more than one model are loaded, the set of selected atoms is for the entire set of loaded models, not just the currently displayed one. Applying a restore to a completely different model should be done with care or it may not have the intended results.

save STATE saveName

Saves the current state of the applet. Some more complex objects, such as dipoles, isosurfaces, lcaoCartoons, and molecular orbitals are not saved.

**See also:**[define initialize refresh reset restore zap](#)[↑top](#) [🔍search](#) [📖index](#)**script or source**

(v. 12.0 -- adds LOCALPATH and REMOTEPATH options.)

Loads and executes the specified script file/url. The hash/pound/sharp character (#) character marks a comment to the end of the line or a semicolon. The semicolon character (;) separates multiple statements on the same line. A script file may load another script file, up to 10 deep. Within the script, for any files indicated, prepending the file name with **\$SCRIPT\_PATH\$** indicates to use the path of script file, not the current path, to find the file.

script [[file-name](#)]

Loads and executes the specified script file/url. A hash/pound/sharp character (#) character marks a comment to the end of the line or a semicolon. A semicolon character (;) separates multiple statements on the same line. A script file may load another script file, up to 10 deep. Starting with Jmol 12.0, the command word SCRIPT is not necessary if the file is quoted or is of the simple format xxxxx.yyy.

script LOCALPATH "path" [[file-name](#)]

When reading a script created with [write STATE](#), the LOCALPATH keyword instructs Jmol to strip all paths beginning with "file:/" down to the indicated path. So, for example, **script LOCALPATH "" "myfile.spt"** indicates that all local files referenced in the state script should be read from the current default directory. LOCALPATH can be used with scripts other than state scripts created by Jmol. The mechanism is simply looking for instances of "file/"some\_file\_name". If this construction is found in any script read, the replacement will be made. (Jmol 12.0)

script REMOTEPATH "path" [[file-name](#)]

When reading a script created with [write STATE](#), the REMOTEPATH keyword instructs Jmol to strip all paths beginning with "http:", "https:", or "ftp:" down to the indicated path. So, for example, **script REMOTEPATH "data" "myfile.spt"** indicates that all remote files referenced in the state script should be read from the subdirectory "data/" within the current default directory. REMOTEPATH can be used with scripts other than state scripts created by Jmol. The mechanism is simply looking for instances of "file/"some\_file\_name". If this construction is found in any script read, the replacement will be made. (Jmol 12.0)

script [[file-name](#)] CHECK

Just checks the file for proper syntax and the presence of files.

script [[file-name](#)] COMMAND n

Executes command n of the designated script file.

script [\[file-name\]](#) COMMANDS n - m  
 Executes commands n through m of the designated script file. The second number may be omitted to indicate "to the end of the file": **script "myfile.spt" COMMANDS 10 -**

script [\[file-name\]](#) LINE n  
 Executes line n of the designated script file.

script [\[file-name\]](#) LINES n - m  
 Executes lines n through m of the designated script file. The second number may be omitted to indicate "to the end of the file": **script "myfile.spt" LINES 10 -**

script APPLELET appletName @ {Jmol math expression}  
 Runs the script command result of the [Jmol math expression](#) in one or more applets. The math expression may be a simple single variable name or quoted string or a more complex expression. If the math expression is a quoted string starting with **script** such as **"script dothis.spt"**, then the indicated script file will be run in each applet. The applet name can be any of the following.

*	all applets on the same web page as the originating applet (the script will run last in the originating applet)
>	all OTHER applets
.(period)	just this applet
name	the applet named <b>name</b> or, if that does not exist, <b>jmolAppletname</b> . Note that the function <code>getProperty("appletInfo.registry")</code> provides a list of applets on all pages, not just the same page. These other applets may be targeted if their full name (which includes a unique number extension) is given.
"name1,name2,..."	all applets matching the specified names. Note that quotes ARE required in this case.

script INLINE @ {Jmol math expression}  
 Runs the script command result of the [Jmol math expression](#) rather than from a file. The math expression may be a simple single variable name or a more complex expression. For example, **var bgColor="red";script INLINE @{"background " + bgColor}** or **var s = "[arg]";script INLINE @{"select " + s}**.

script javascript:functionCall()  
 Applet only: Evaluates the return of the indicated JavaScript function as the script to be executed. Execution is blocked if the web page parameter `_jmol.noEval` = true. Note that this is different from the [javascript](#) command, which simply evaluates the specified JavaScript command. Here the function is evaluated on the embedding page, and the return from that function, which is presumed to be Jmol script, is evaluated within Jmol.

where  
**[file-name]** is any valid filename or URL -- (string)

See also:  
[getProperty set \(callback\)](#)



## select

Selects the atoms identified by the expression. If no expression is given then all atoms are selected. Starting with Jmol 11.6, a second property expression can be added for more general selection of atoms based on atom properties.

select {default: ALL}  
 Selects all atoms (possibly not H atoms).

select [\[atom-expression\]](#)  
 Selects atoms based on an [atom expression](#). To select atoms specific to a specific model when more than one model is present, use "n" where "n" is the model number. For example, to select all atoms of model 3, use **select \*/3**.

select [\[atom-expression\]](#) (property expression)  
 Starting with Jmol 11.6 the **select** command allows for full utilization of Jmol math for atom selection. If the first parameter is an atom expression, such as **{}** or **{10-30}**, a second parameter can be included. This second parameter must evaluate to a TRUE/FALSE expression involving the individual atoms of the atom expression. Parentheses around the property expression are optional. For example: **select (\*,ca) (atomY < atomX)** selects for all alpha carbons for which their X coordinate is less than their Y coordinate. Note that in this context, "x", "y", and "z" are variable names, not coordinates. Use "atomX", "atomY", and "atomZ" if you need to refer to atom coordinates. The variable `_x` is assigned to the individual atom being tested. This variable can be used in nested select() functions within the select command to represent the atom being tested. For example,

**select (\*,ca) (phi < select(y; (\*,ca); y.resno = \_x.resno + 1),phi)** selects alpha carbons of amino acid residues that have phi angles less than that of the phi angle of the next amino acid in the chain.

where  
**[atom-expression]** is any [expression](#) that evaluates to a set of atoms

Examples: [in new window using 1a3n.pdb](#)

```
select carbon;color white
select protein;ribbons on
select *:D;color blue
select [HIS]92:D.spacefill 300
select [HIS]92:D.N.spacefill 600
select [HIS]92:D.C?;color orange
select [HIS]92:N;color [255,196,196]
select elemno<7;spacefill 200
select within(group, within(10.0, :a));color green;select :a;color red
select within(chain, [HIS]92);color white;
select within(chain, within(3.0,[HIS]92:D));color purple;
select within(chain,within(5.0,[HIS]92));color white
select 95:a:L # selects chain L, residue 95, insertion code a
```

See [select.htm](#)

See also:  
[display](#) [hide](#) [restrict](#) [subset](#)



## selectionHalos

When ON, Jmol displays [halos](#) around atoms when they are selected. The radius of the halo is always from 4 to 10 pixels larger than the current setting for [spacefill](#) or the current setting of halo radius using the [halos](#) command. The color of any specific halo is determined as described for [color selectionHalos](#).

selectionHalos ON/OFF {default: ON}



## set

Jmol allows a wide range of settings to be changed using the SET command -- see the categories below for details. Starting with Jmol 11.2, in most cases the SET command is no longer necessary -- any simple value you can set with SET can be set simply using an assignment of a value to a variable name:

strandCount = 6

instead of **SET strandCount 6**. However, it is recommended that you use SET for all [Jmol parameters](#), just as a way of clearly indicating in scripts that you are setting a Jmol parameter and not a user variable. In all cases below, "ON" and "TRUE" are equivalent, and "OFF" and "FALSE" are equivalent. Different words may be used simply because they seem more appropriate for a particular parameter.

set  
**set** by itself lists all Jmol parameters that can be set and all Jmol read-only variables (starting with underscore), along with their current values.

set xxx?  
**set** followed by characters ending in question mark lists all Jmol parameters starting with those characters and all Jmol read-only variables starting with underscore and those characters.

See also:  
[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [case default](#) [echo](#) [for if message](#) [reset](#) [switch](#) [while](#)

[variables](#) [↑top](#) [🔍search](#) [📖index](#)

set (antialiasing)

Antialiasing is the smoothing of jagged lines and sharp boundaries in an image. Jmol 11.4 introduces antialiasing independently for display and for image creation using the [write](#) command. If the display is antialiased, the option also exists to antialias translucent objects or not. Memory requirements are doubled and rendering performance will be diminished when antialiasDisplay is turned on.

set antialiasDisplay OFF

Turning this parameter ON results in smoothing of the model rendering in the window.

set antialiasTranslucent ON

Turning this parameter OFF removes smoothing of the translucent components of the window in addition to the opaque components when antialiasDisplay is turned on.

set antialiasImages ON

Turning this parameter OFF disables smoothing of the images created using the [write](#) command.

set (bond styles)

This group of commands sets the appearance of various optional bond effects for the model.

set bondMode AND

The default Jmol condition. When script commands affect a set of atoms, BOTH atoms must be in the set for the bonds between them to also be affected. Similarly affects the display of backbone units in the selection of protein residues and nucleic acid bases.

set bondMode OR

When script commands affect a set of atoms, EITHER atom may be in the set for the bonds also to be affected. Similarly affects the display of backbone units in the selection of protein residues and nucleic acid bases.

set bondModeOr FALSE

Setting this parameter TRUE is equivalent to **set bondMode OR**; it can be tested using Jmol math.

set bondRadiusMilliAngstroms (integer)

Sets the default bond radius in milliAngstroms. Immediately applies this to all bonds in all models.

set bondTolerance (decimal)

When autobonding, the value of bondTolerance is added to the two bond radii of atoms being tested for a bond. A larger bondTolerance allows atoms that are further apart than the sum of their listed radii to still be bonded. This parameter should be adjusted prior to file loading for proper maintaining of the Jmol state.

set dipoleScale (-10.0 to 10.0)

Sets the overall scale of all displayed dipole vectors.

set hbondsRasmol TRUE

For PDB files, generally Jmol will use a hydrogen bond calculation based on RasMol; for other file types, Jmol uses its own calculation. Setting this parameter FALSE causes hydrogen bonds to be calculated for all models using the Jmol calculation and involving [set hbondsAngleMinimum](#) and [set hbondsDistanceMaximum](#).

set hbondsSolid FALSE

Setting this parameter TRUE causes hydrogen bonds to be displayed as solid lines rather than dotted lines.

set hbondsBackbone FALSE

Hydrogen bonds between protein amino acid residues or nucleic acid base pairs are displayed as lines. These lines can be displayed whether or not the H atoms are present in the file, and can be drawn either between the two non-hydrogen atoms involved in the bond (O or N, typically, the default) or, alternatively, between the two backbone alpha-carbon atoms, depending upon the desired effect.

set minBondDistance (decimal)

Sets the minimum bond distance for autobonding. Should be set prior to file loading for proper maintenance of the Jmol state.

set showMultipleBonds ON

In some file formats (.mol files, for example) the connection data may indicate the bond type--single, double, triple, or quadruple. Use **set multipleBonds OFF** when you want all bonds to appear as single bonds.

set ssbonds BACKBONE or SIDECHAIN

[variables](#) [↑top](#) [🔍search](#) [📖index](#)

Sulfur-sulfur bonds in cysteine bridges of proteins are displayed as lines. These lines can either be between the two sidechain sulfur atoms (the default) or between the two backbone alpha-carbon atoms, depending upon the desired effect.

set ssBondsBackbone FALSE

Setting this parameter TRUE is an alternative method of setting disulfide bonds to backbone; it can be tested using Jmol math.

Examples:

See [bonds.htm](#)

See also:

[bondorder](#) [connect](#) [hbonds](#) [set \(files and scripts\)](#) [ssbonds](#) [wireframe](#)

[variables](#) [↑top](#) [🔍search](#) [📖index](#)

set (callback)

Jmol 11.0 introduces dynamic JavaScript callback function definition. You can specify the functions to receive callbacks, and you can change the functions at any time. To turn off callbacks of a given type, specify "NONE". The function name must be present in JavaScript on the page containing the applet. Specifying "alert" will send the message to the user via a JavaScript alert. Note that quotation marks are required around the function name. If the filename starts with "jmolscript:" then instead of JavaScript being run, Jmol executes the Jmol script that follows. For example, **set hoverCallback "jmolscript:script hover.spt"** executes the Jmol script "script hover.spt" when an atom is hovered over. The code in the file hover.spt can then respond to that hovering action without the need for JavaScript.

set AnimFrameCallback "function name"

Sends a message indicating a change of [frame](#). For compatibility with Chime, the second parameter of this function returns a number ONE LESS than the actual frame number. Starting with Jmol 11.2, this function returns nine parameters. The product of the last two parameters indicates the true direction of animation.

function animFrameCallback(app, frameNo, fileNo, modelNo, firstNo, lastNo, isAnimationRunning, animationDirection, currentDirection)

app	String	The name of the applet
frameNo	int	The current frame number (0-based)
fileNo	int	The current file number (1-based)
modelNo	int	The current model number within the current file (1-based)
firstNo	int	The first frame of the animation range, expressed as fileNo*1000000+modelNo
lastNo	int	The last frame of the animation range, expressed as fileNo*1000000+modelNo
isAnimationRunning	int	0 (animation is off) or 1 (animation is on)
animationDirection	int	1 (animation direction +1) or -1 (animation direction -1)
currentDirection	int	1 (forward) or -1 (reverse)

set EchoCallback "function name"

Sends a message each time the [echo](#) command is executed. If an EchoCallback function is not defined, these messages go to the MessageCallback function.

set EvalCallback

Generally the Jmol applet is allowed to use the eval() function of the host page JavaScript using the [javascript](#) command (unless execution of JavaScript by the applet has been specifically disallowed by setting \_jmol.noEval = true prior to the jmolApplet() call). The setting of **evalCallback** function must be made prior to jmolAppletCall() using **jmolSetCallback("evalCallback","someFunctionName")**. It cannot be set using **set evalCallback**. The callback sends the JavaScript to be evaluated back to the web page for evaluation rather than initiating that evaluation within Jmol. This could be important for the signed applet in order to isolate threads or for debugging applet calls to eval(). It is used in <http://chemapps.stolaf.edu/pe/protexpl>.

set HoverCallback "function name"

Sends a message indicating what atoms is being hovered over, indendently of whether [hover](#) is ON or OFF.

set LoadStructCallback "function name"

Sends a message each time a file is [loaded](#).

set MeasureCallback "function name"

Sends a message indicating the status of measurements made by the user. If a MeasureCallback function is not defined, these messages go to the MessageCallback function.

set MessageCallback "function name"

Sends a wide variety of messages during script execution.

set MinimizationCallback "function name"

Sends a message that indicates the status of a currently running [minimization](#).

set PickCallback "function name"

Sends a message that depends upon the current status of [set picking](#).

set ResizeCallback "function name"

Sends a message indicating changes of the window size.

set ScriptCallback "function name"

Sends messages indicating the status of script execution. Line-by-line script commands are sent if one has **set debugScript TRUE**. If a ScriptCallback function is not defined, these messages go to the MessageCallback function.

set SyncCallback "function name"

The SyncCallback method allows a JavaScript function to intercept and modify or cancel an applet-applet [sync](#) message. If the called function returns "" or 0, the synchronization is canceled; any other string is substituted for the script and sent to the other currently synchronized applets.

See also:  
[getProperty script](#)

[variables](#) [↑top](#) [🔍search](#) [iindex](#)

### set (debugging) (v. 11.0 introduces an adjustable logging level and showScript option)

set debug OFF

Turning this parameter ON sets **debugScript** ON and **logLevel** to 5; setting it OFF returns these settings to their standard values (OFF and 4, respectively).

set debugScript OFF

Turning this parameter ON enables debugging (going to a JavaScript message [callback](#)).

set delayMaximumMs 0

Sets the maximum delay that scripts will use, primarily used for testing scripts. The default setting of this parameter, 0, disables maximum delay checking.

set fontCaching TRUE

A debugging parameter.

set historyLevel (integer)

Primarily for debugging complex scripts. Sets the level of depth in scripts for which the command [history](#) is recorded. A setting of 0 (the default) indicates that commands in scripts run using the [script](#) command should not be recorded. A setting of 1 indicates that such commands should be recorded for script commands given at the top level. A setting of "n" indicates that all commands executed from script commands through level "n" should be recorded. A setting of -1 turns off all history recording.

set logLevel (0 - 5)

Jmol 11.0 allows you to set the amount of logging sent to the Java console (as opposed to the Jmol [console](#)). The default level is 4, "information, warnings, and errors". Levels include:

0	no messages whatsoever
1	fatal errors only
2	all errors
3	all warnings and errors
4	information, warnings, and errors
5	full debugging

set scriptReportingLevel (integer)

Sets the maximum script depth (how many scripts have been called) for which certain messages should appear in the console and be reported back via callbacks. A value of 0 (default) displays messages only from the topmost level -- as, for example, commands entered from the console; a value of 1 displays messages from the top level and messages due to commands such as "script t.spt" but not from scripts that are called from t.spt; etc. A value of -1 turns off all reporting. Affected commands include [connect](#), [select](#) (and related commands), and [set](#). This parameter is particularly useful when scripts utilize [message/goto](#) to control program flow.

set showScript OFF

Turning this parameter ON causes Jmol to show the script commands from a script file as they are executed. A slight difference may be observed when showScript is set in comparison to normal operation in that when showScript is set, an automatic refresh is executed after every command.

set showScript milliseconds

Shows the script commands from a script file as they are executed and pauses the specified number of milliseconds between commands. Setting the delay to 0 turns script showing off.

[variables](#) [↑top](#) [🔍search](#) [iindex](#)

### set (files and scripts) (v. 11.0 -- introduces several new settings)

The following commands relate to how files and scripts are loaded and how scripts are executed.

set allowEmbeddedScripts

When set TRUE (default), Jmol will read scripts that are contained in certain file types and append them to the script set using **set defaultLoadScript** (below). Embedded scripts are indicated by "jmolscript:" followed by valid Jmol script commands in the following file locations:

CIF	any line(s)
MOL	line 3
PDB	any REMARK line(s)
XYZ	line 2

Support for additional file types will be provided upon user request.

set appendNew TRUE

Setting this parameter to FALSE causes Jmol to add atoms to the last model in a file set rather than add a whole new model when [load APPEND](#) is used.

set appletProxy "URL"

Sets the URL for a proxy server when [loading](#) a file or reading a [pmesh](#) or [isosurface](#) or when reading a file-based [script](#). A proxy server is a server-side application (typically written using PERL, PHP, or ColdFusion) on the same host as the JAR file that can deliver files from other servers on the internet. Jmol appends "?url=" followed by the URL of the requested data file to the indicated proxy server name. NOTE that Java security requires that this call be to the same server that hosts the JAR file.

set applySymmetryToBonds OFF

Turning this parameter ON instructs Jmol, when applying symmetry to atoms, as in "load xxx.cif {1 1 1}", to also apply symmetry to the bonds indicated in the file. The flag is useful when normal Jmol autobonding would not properly connect atoms, but the model is "molecular" -- the base atom coordinates are correct for whole molecules. The flag should not be used in cases where the application of symmetry operations creates new bonds that were not present in the original set, as for quartz.cif, where there is only one bond initially, and after applying symmetry new bonds are created that are between atoms that were created using two different symmetry operations. If bonds are not indicated in a file, this flag is ignored, and Jmol uses its autobonding algorithm to create bonds.

set atomTypes "..."

The MOL2 and MDTOP file formats specifically do not contain enough information within the file to determine the element for all atoms. (The atom types depend upon which force field was employed, and not all force fields allow the direct decoding of element from atom type.) The **atomTypes** setting allows the user to correlate atom types with specific elements for these specific readers. The syntax is "abcd=>C;efgh=>H;...", indicating that atom type "abcd" is carbon, atom type "efgh" is hydrogen, etc. The matching is case sensitive.

set autobond ON

Some file formats that Jmol reads, such as XYZ, do not contain bonding information. In these cases, the default action for Jmol is to generate bonds automatically based on an algorithm. When given prior to loading a model, the **set autobond OFF** command causes Jmol to not do any automatic bond creation when subsequent models are loaded.

set autoLoadOrientation FALSE

Specifically for Spartan and Sygress readers for Jmol 11.8, this setting (default FALSE) sets the default orientation (when the file is loaded or [reset](#) is issued) to that given in the file. Starting with Jmol 12.0, the setting is ignored, and the default orientation is automatically used unless FILTER "NoOrient" is included.

set currentLocalPath "path"

Sets the path to files specifically for the dialog box that pops up when a file name is "?" in a command such as [load](#) or [write](#). Automatically set to this path when the user navigates to a new directory during that dialog.

set dataSeparator "separator text"

Issued prior to the [data](#) command, sets the text that will separate one set of file data from another, thus allowing the inline loading of multiple file types.

set defaultDirectory "directory path"

(APPLET ONLY) Sets the default directory to use for reading all files. This will generally be a relative path such as "../data" or "../files". Note that Java security requires that if the applet is run from a hard drive rather than via the internet, all files read must be either in the directory containing the JAR file or in a subdirectory of that directory. If the applet is unsigned and loaded from the internet, then Java security typically requires that all files read are from the host from which the JAR file was read. But see [set appletProxy](#), above. This flag is applicable only to the Jmol applet, not to the Jmol application.

set defaultLattice {i j k}

For crystallographic systems, sets the default lattice to be loaded. {1 1 1} loads the standard single unit cell (555). {2 1 1} loads two unit cells along the i direction, cells 555 and 655. {2 2 2} loads a set of eight unit cells; {3 3 3} loads a set of 27. This last is probably most useful, because it loads one unit cell surrounded by all 26 cells sharing its 6 faces, 12 edges, and 8 corners.

set defaultLoadScript "script"

Sets a script to run after any file is loaded. The script must be in quotations. If the script itself needs quotation marks, then it should be placed in a file and indicated as follows: **set defaultLoadScript "script myscript.scr"**.

set edsUrlCutoff "url"

Sets the URL to use for obtaining the Uppsala Electron Density server secondary file that gives cut-off/sigma information.

set edsUrlFormat "url"

Sets the URL to use for obtaining the Uppsala Electron Density server MAP file.

set forceAutoBond OFF

**set forceAutoBond ON** tells Jmol to disregard any bonding information in a file and use its own internal algorithm for determining connectivity. Its effect is for all future file loads until set OFF. This setting is particularly useful for some PDB and mmCIF files that already have a threshold amount of bonding, so that a full set of bonding can be created automatically at load time. This is necessary for proper assignment of secondary structure.

set history nLines

Sets the number of lines of command history to record (minimum 2) and turns history recording ON. **set history 0** turns off the command history feature but does not actually set the number of lines to zero. See also [history](#) and [show history](#).

set loadFormat "URL"

The load format, by default "http://www.rcsb.org/pdb/files/%FILE.pdb", allows setting of the URL that will be used when "=" is used in front of a file name in a [load](#) command, for example: **load =1crn**.

set scriptQueue ON

Turning this parameter OFF disables script queuing. Setting this parameter OFF should never be necessary, but it is provided here as an option. When script queuing is enabled (the default), scripts that are [looping](#) require [quit](#) or [exit](#) to be executed in a subsequent script in order to complete. When script queuing is turned off, scripts from different threads may collide and cause unpredictable behavior or crashing of Jmol.

set smallMoleculeMaxAtoms 40000

This parameter sets the maximum number of atoms for default rendering of the model. Models with this number or fewer atoms will be rendered with the default spacefill rendering and the default bond diameter; models with more than this number of atoms will be displayed by default with **spacefill 0**; **wireframe 1** to conserve resources.

See also:  
[bondorder](#) [cd](#) [connect](#) [exit](#) [hbonds](#) [initialize](#) [load](#) [quit](#) [set \(bond styles\)](#) [ssbonds](#) [wireframe](#) [zap](#)

set (highlights)

This command group allows for annotation and highlighting of atoms in terms of labels and "halos."

set display SELECTED/NORMAL  
(deprecated; see [selectionHalos ON/OFF](#))

set frank

See [frank](#)

See also:  
[backbone](#) [background](#) [cartoon](#) [dots](#) [echo](#) [ellipsoid](#) [font](#) [geoSurface](#) [hover](#) [label](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(labels\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [set echo](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

[variables](#) [↑](#) [top](#) [🔍](#) [search](#) [📘](#) [index](#)

set (labels)

This command group sets parameters associated specifically with atom labels. If the atom expression is not indicated, the action is applied to the currently selected atoms. (Jmol 11.4)

set fontScaling OFF

When **fontScaling** is set ON, any labels created after that are rescaled when the model is [zoomed](#).

set fontSize [[font-size](#)] {default: 8}

Sets the font size for atom labels for the currently selected atoms.

set labelAlignment LEFT, RIGHT, or CENTER

Sets the label text alignment within a multi-line label as left-, right-, or center-justified. (For overall label alignment, see [set labeloffset](#), below.)

set labelAtom ON/OFF {default: ON} { atom expression }

If a selected label is rotated behind an atom, it is hidden by that atom (default). If an atom expression is given, an indicator of ON or OFF must also be given. OFF is the same as "SET LABELFRONT".

set labelFront ON/OFF {default: ON} { atom expression }

The selected labels will always appear in front of all atoms. If an atom expression is given, an indicator of ON or OFF must also be given. OFF is the same as "SET LABELATOM".

set labelGroup ON/OFF {default: ON} { atom expression }

Selected labels appear in an invisible plane just in front of the atoms of their group only. Applicable only to PDB/mmCIF files. If an atom expression is given, an indicator of ON or OFF must also be given. OFF is the same as "SET LABELATOM".

set labelOffset [[x-offset](#)] [[y-offset](#)] { atom expression }

Sets the label offset relative to the atom being labeled. A positive number indicates the number of pixels between the atom center and the beginning of the label. A negative number indicates the number of pixels between the atom center and the end of the label, with the entire label to the left of the atom. Zero indicates centered.

set labelPointer OFF { atom expression }

Turning this parameter ON instructs Jmol to add lines pointing from the selected atoms to their respective labels, using the color of the label text ([color label xxxx](#)).

set labelPointer BACKGROUND { atom expression }

Turns on label pointers to selected atoms, drawing them in the color of the label background.

set labelToggle { atom expression }

Toggles the labels on or off for the specified set of atoms.

where  
**[font-size]** is approximately the same as Rasmol -- (integer, 6 to 63)  
**[x-offset]** is the x-offset -- (integer)  
**[y-offset]** is the y-offset -- (integer)

See also:  
[echo](#) [font](#) [hover](#) [label](#) [set \(highlights\)](#) [set echo](#)

[variables](#) [↑](#) [top](#) [🔍](#) [search](#) [📘](#) [index](#)

set (language)

Starting with Jmol 11.2, you can set the language for the applet popup menu to a new lanuage just by selecting the desired language from the language submenu.

set language "[two-letter code]"

The command **set language "xx"**, where **xx** is a two-letter abbreviation for a language, allows this to be done programmatically. Supported languages are listed on the popup menu.

set languageTranslation ON

Setting this parameter OFF turns off language translation. This may be important for pages that need to parse messages coming from Jmol in American English.

[variables](#) [top](#) [search](#) [index](#)

## set (lighting)

(v. 11.0 -- clearer naming)

This commands in this group determine the overall lighting effects, size, and rotation for the model. Note that in a multi-applet environment, changing lighting values (ambientPercent, diffusePercent, specular, specularExponent, specularPercent, or specularPower) changes them for ALL applets. Effect on another applet may not appear until that model is rotated by the user or some command is issued to that applet that requires updating the display.

set ambientPercent (integer 0 to 100)  
Sets the amount of "ambient" light filling the shadows created by the apparent light source. An ambientPercent value of 0 creates an effect of a spotlight on a stage; a value of 100 removes the shadow entirely, creating a flat, nonrealistic effect.

set diffusePercent (integer 0 to 100)  
Sets the amount of "diffuse" light apparently emanating from the spotlight, but not hitting and reflecting off the model directly. Setting the diffusePercent value to 0 turns this effect off; giving the effect of the model in a black-walled room where no light reflection is possible, effectively turning off all but specular reflections.

set phongExponent (integer 0 to 1000)  
Sets the specular strength using a number which is 2<sup>(specularExponent)</sup> [Jmol 12.0]

set specular OFF  
Turns off the specular reflection (the shiny dot on a curved surface).

set specularExponent (integer 1 to 10)  
Ranging from 1 to 10, the specular exponent determines the tightness of the specular spot, with 1 being very broad and 10 being very tight. Technically, log<sub>2</sub>(phongExponent).

set specularPercent (integer 0 to 100)  
Sets the size of the apparent reflection from a light source.

set specularPower (integer 0 to 100)  
Sets the density of dots in the specular reflection, producing the effect of a very dim light (10) to a very bright light (100).

set zShadePower (integer)  
This parameter can be adjusted (typically 1, 2, or 3) to create a stronger effect of fog when **zShade** is set ON.

Examples:

See [lighting.htm](#)



See also:

[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

[variables](#) [top](#) [search](#) [index](#)

## set (measure)

Sets characteristics of the measurement labels and lines. See also [measure](#).

set defaultDistanceLabel "format"  
Sets the format of the labels for [distance measurements](#).

set defaultAngleLabel "format"  
Same as for defaultDistanceLabel, but for angles.

set defaultTorsionLabel "format"  
Same as for defaultDistanceLabel, but for torsions.

set dynamicMeasurements ON

This parameter should be set TRUE if molecules or atoms are being moved relative to one another using [translateSelected](#) or [rotateSelected](#) so that the numbers associated with distances and angles updates properly as atoms are moved.

set measurements [\[width-in-angstroms\]](#)  
Sets the width of the measurement line in angstroms.

set measurements [\[linewidth-pixels\]](#)  
Sets the width of the measurement line in pixels.

set justifyMeasurements FALSE  
Turning this parameter TRUE right-justifies measurement labels

set measurements DOTTED  
Sets the measurement line to be dotted.

set measurementLabels ON  
Turning this parameter OFF turns off measurement LABELS only, leaving the lines. (To turn off both, use **set showMeasurements OFF**.)

set measurementUnits [\[distance-unit\]](#)  
Sets the units for measurement display to be Angstroms, nanometers, or picometers.

set showMeasurements TRUE  
Setting this parameter FALSE turns off measurement lines and labels BOTH. (To turn off just the label, use **set measurementLabels OFF**.)

where  
[distance-unit] is ANGSTROMS, AU, BOHR, NM, NANOMETERS, PM, or PICOMETERS  
[width-in-angstroms] is a (decimal, <2.0)  
[linewidth-pixels] is an (integer)

[variables](#) [top](#) [search](#) [index](#)

## set (misc)

(v. 11.0 -- new options for temperature, measurements, and backgroundmodel)

In all cases below, "ON" and "TRUE" are equivalent, and "OFF" and "FALSE" are equivalent.

set allowGestures FALSE  
This parameter is primarily intended for a touch-screen context. Setting this parameter TRUE allows single-point gestures to be detected. Currently the only single-point gesture supported is a "swipe" or "flick" created by a motion of the mouse or finger that is continuing along a line at the time the mouse or finger is released.

set allowKeystrokes FALSE  
Set this parameter TRUE to allow key strokes in the applet or application window to be interpreted as script commands. These keystrokes will be displayed if the [showKeystrokes](#) setting is TRUE.

set allowModelKit TRUE  
Set this parameter FALSE to disable [modelKitMode](#).

set allowMultiTouch TRUE  
Set this parameter FALSE to disable multi-touch gestures (two-finger spread for zoom, two-finger drag for translation) within a multi-touch environment.

set allowRotateSelected FALSE  
When set TRUE, this parameter allows user rotation of the molecule containing the selected atom using the mouse (holding ALT down while dragging). The coordinates of the rotated molecule will be slightly degraded in this process.

set animationFps (integer)  
Same as "animation FPS" -- sets the animation delay in frames per second.

set autoFPS FALSE  
Setting this parameter TRUE adjusts the rate of frame changes in an animation to a lower rate if the rendering cannot keep up with the frame changing.

set axesColor "color\_name"  
Sets the color of all axes. (Same as [color axes](#) [file](#).)

set axis1Color "color\_name"  
Sets the color of the X axis.

set axis2Color "color\_name"  
Sets the color of the Y axis.

set axis3Color "color\_name"  
 Sets the color of the Z axis.

set atomPicking TRUE  
 Setting this parameter FALSE disallows picking of atoms. See also **set bondPicking** and **set drawPicking**.

set backgroundModel (integer >= 1) or "file.model"  
 Sets a particular model or animation frame to be fixed in place during an [animation](#) or when displaying models [loaded](#) from a multi-model file. **set backgroundModel 0** turns this feature off. For a multiframe context, the model must be given in "file.model" format. Certain restrictions apply to scripts when a background model is displayed; in that case it may be important to turn the model off and then back on during selected scripting.

set bondPicking FALSE  
 Setting this parameter TRUE allows picking of bonds. If pickCallback is enabled, a message is sent to the callback function providing detailed information about the bond that was picked. See also **set atomPicking** and **set drawPicking**.

set chainCaseSensitive FALSE  
 Jmol can be set to read the chain designations in PDB, mmCIF, and related files either with or without case sensitivity. With the default **set chainCaseSensitive FALSE**, the chain designations are interpreted as case-insensitive. With **set chainCaseSensitive ON**, the chain designation is read in a case-sensitive manner -- chain "A" is different than chain "a". This supports [PDB format](#) model files with more than 26 chains. The default startup up mode is OFF -- chain designation "a" in a SELECT command will refer to chain "A" in a file.

set colorRasmol FALSE  
 Setting this parameter TRUE provides an alternative way to set the default color scheme to RasMol; testable with Jmol math.

set defaultColorScheme JMOL or RASMOLO  
 Sets the default color scheme to be the traditional Rasmol/Chime scheme or the newer, more subtle, Jmol scheme. This command does not actually change the display for an object unless that object is currently being displayed using the default color scheme. See the [Jmol Colors page](#) for default color scheme details.

set defaultDrawArrowScale (decimal)  
 Sets the default scale for the arrow tip for [arrows](#).

set defaults JMOL or RASMOLO  
 Sets the overall defaults to be Jmol standard or more similar to Rasmol, including default color scheme, axes orientation, zero-based XYZ numbering, no spacefill, and simple wireframe. Applied immediately; should be used prior to file loading.

set defaultVDW JMOL or BABEL or RASMOLO or USER  
 Sets the default van der Waals radius set for [spacefill](#) and related commands. For a detailed listing of the values used, see [vdw comparison.xls](#). USER refers to a set of data provided by the user using the [data](#) command.

set dotDensity -3 to 3  
 When Jmol displays [dots](#), the density of dots is determined by the scale of the display. At a small scale, Jmol may display as few as 12 dots; as zoom increases, this number increases to 42, then 162, and finally, at high zoom, it becomes 642 dots. The **dotDensity** setting allows control over how many dots are displayed at the various scale level cutoffs. (The actual calculation does not use zoom; rather, it uses a measure of pixels per micron). Setting **dotDensity** to -3 results in Jmol always displaying 12 dots; setting it to 3 results in Jmol always displaying 642 dots. Settings in between these values decrease or increase the number of dots relative to the default setting (0).

set dotScale (integer)  
 Sets the size of [dots](#).

set dotsSelectedOnly FALSE  
 Setting this parameter TRUE instructs the [calculate surface](#) command to disregard atoms that are not selected when calculating the position of the surface (which then determines the parameter [surfaceDistance](#) for each atom). Also tells Jmol to ignore nonselected atoms when creating a [dot surface](#). **set dotsSelectedOnly TRUE** would allow, for example, a continuous set of dots around a ligand even if it is in contact with a protein.

set dotSurface ON  
 Setting this parameter OFF instructs Jmol to draw complete spheres of [dots](#) around each selected atom rather than only the dots that "dot surface."

set dragSelected OFF  
 When ON, allows the user to move selected atoms by pressing ALT-SHIFT-LEFT and dragging; when combined with **set pickingstyle DRAG**, just LEFT-dragging moves the atoms, and the ALT and SHIFT keys are not required.

set drawHover OFF  
 When ON, and the user hovers over a point associated with [draw](#) object, the name of the object is displayed next to the point.

set drawPicking OFF  
 When ON, Jmol reports picking of points associated with [draw](#) objects to the console, the messageCallback function, and the pickCallback function. In addition, starting with Jmol 11.6, setting drawPicking true enables measurement of distances and angles involving drawn objects such as points, lines, arrows, and planes. Measurements of this type only appear transiently; they are not saved. For Jmol 11.6, see also **set atomPicking** and **set bondPicking**.

set exportDrivers "driver\_list"  
 Sets (or allows you to inspect, expand, or limit) the list of export drivers available to Jmol. As of Jmol 11.4.1, "Maya;Vrml;Povray".

set formalCharge (integer)  
 Sets the formal charge for the selected atoms.

set fractionalRelative FALSE  
 Sets the meaning of {1/2 1/2 1/2} to be relative to the current [unitcell](#) (TRUE) or not (FALSE, default).

set helixStep (integer)  
 Sets the step for calculating straightness and for [draw helix axis](#). The default value is 1, but a value of 2, for example, allows calculating straightness and axes for structures based on pairs of residues.

set helpPath "URL"  
 Sets the web page for the [help](#) command.

set hoverDelay (decimal)  
 Sets the length of time in seconds before a hovering action is acknowledged.

set hoverLabel (string)  
 Sets the label displayed upon hovering.

set imageState ON  
 The **imageState** property, when TRUE, allows Jmol to insert into JPG and PNG files its state. This allows images to serve both as 2D and 3D models.

set isKiosk OFF  
 For a multi-touch screen, set this parameter ON if the Jmol applet is running in a browser with the kiosk mode enabled, indicating there are no other applets or applications running.

set isosurfacePropertySmoothing ON  
 In the case of an isosurface that is mapped using atom-based property data, the default action (as of Jmol 11.4) is to smooth out the coloring based on an averaging of color weighted by a factor of 1/distance<sup>4</sup> to atoms. Turning this parameter OFF tells Jmol not to do this and instead produce a patchwork mapping that assigns a color based on the property of only the nearest atom (Jmol 11.2 behavior).

set loadAtomDataTolerance (decimal)  
 Sets the maximum distance away from a point that that an atom can be found when applying atom data using the [load \[property\]](#) command. Default 0.01 Angstroms. Applies the data to all atoms in similar unit cells if the data being read. This allows applying the same data to atoms in all unit cells. (Jmol 11.8)

set measureAllModels OFF  
 In situations where there are multiple models, typically only one model is displayed at any given time. In that situation, if a user clicks on a pair of atoms to measure a bond distance, then only one measurement is made. Starting with Jmol 11.0, using **set measureAllModels ON**, when the user makes measurements in any one frame, ALL similar measurements in all models in hidden frames are generated at the same time. **set measureAllModels ON** thus allows for very efficient measuring and investigation of differences in a bond distance or bond angle or dihedral angle across multiple models.

set messageStyleChime FALSE  
 Changes the style of message reporting by script callbacks to one similar to Chime.

set minimizationCriterion (decimal)  
 Sets the criterion for stopping a [minimization](#). The default value is 0.001; the minimum value is 0.0001.

set minimizationRefresh TRUE  
 Set this flag FALSE to not display [minimizations](#) as they occur.

set minimizationSilent FALSE  
 Turns off messages reporting energies for the minimization.

set minimizationSteps (integer)  
 Sets the number of steps after which a [minimization](#) will stop even if the desired criterion has not been reached. Default value is 100.

set mouseDragFactor (decimal)  
 Sets the sensitivity of the mouse when dragging. Introduced in Jmol 12.0, Jmol 12.0 will change the default sensitivity of the mouse wheel to work less aggressively. To use the older Jmol 11.8 sensitivity, use set mouseWheelFactor 1.15; the default for Jmol 12.0 is 1.02. Similarly, Jmol 12.0 will have a less aggressive drag sensitivity for the mouse, allowing the mouse to work more appropriately, especially in a touch-screen environment, (where the "mouse" is a finger). The default value of 1.0 for set mousedragFactor allows a 180-degrees rotation when the pointer drags across the full window width.

set mouseWheelFactor (decimal)  
 Sets the sensitivity of the mouse wheel. Introduced in Jmol 12.0, Jmol 12.0 will change the default sensitivity of the mouse wheel to work less aggressively. To use the older Jmol 11.8 sensitivity, use set mouseWheelFactor 1.15; the default for Jmol 12.0 is 1.02. Similarly, Jmol 12.0 will have a less aggressive drag sensitivity for the mouse, allowing the mouse to work more appropriately, especially in a touch-screen environment, (where the "mouse" is a finger). The default value of 1.0 for set mousedragFactor allows a 180-degrees rotation when the pointer drags across the full window width.

set multiprocessor FALSE

Turns on parallel multiprocessing. If this setting cannot be set true, then it means you do not have a multiprocessor machine. Used in association with the [parallel](#) and [process](#) commands.

set pdbGetHeader FALSE

Setting this flag TRUE tells Jmol to keep the header portion of the last-loaded PDB file in memory for later retrieval using getProperty("fileHeader"). If FALSE (default), the header is not saved, and a call to this function is slower, since it requires reloading the PDB file and parsing it for its header.

set pdbSequential FALSE

Setting this flag TRUE tells Jmol to assume the groups listed for a given chain in a PDB file are in order already and that Jmol should not check for proper inter-group bonding when assigning polymer sequences that form the basis of secondary structure. This flag allows for custom PDB files where the groups of a chain may not be physically adjacent, yet it is still desired to represent them as single structures.

set percentVdwAtom (integer)

The default size of an atom created when a file is loaded as a percent of the atom's van der Waal radius (Jmol standard value: 20).

set pickingSpinRate (integer)

The rate of spinning that occurs when a user clicks on the end of a line created using [draw](#) (default: 20).

set pointGroupDistanceTolerance (decimal)

Sets the tolerance for two atoms being considered identical after application of a rotation. See [calculate pointgroup](#) for details.

set pointGroupLinearTolerance (decimal)

Sets the tolerance for two axes being considered identical prior to application of a rotation. See [calculate pointgroup](#) for details.

set pickLabel (string)

Sets the format of the message sent to the console and callback functions when an atom is clicked.

set preserveState TRUE

This option turns off many memory-consuming features of Jmol that are necessary for preserving the state. It can be used in situations where memory is at a premium and there is no desire to write or save the current Jmol state. (Jmol 12.0)

set propertyAtomNumberColumnCount (integer)

An integer value of 0 or greater. Sets the number of columns to be read for **propertyAtomNumberField**.

set propertyAtomNumberField (integer)

An integer value of 0 or greater. Sets the column (when **propertyAtomNumberColumnCount** > 0) or free-format field (otherwise) for the atom numbers in a [data](#) set. These are the atom numbers designated in a PDB file or numbers starting with 1 otherwise. Setting the field to 0 indicates that there is no such field, and values should be read in sequentially.

set propertyColorScheme "colorSchemeName"

Sets the color scheme associated with properties. SchemeName, in quotes, may be one of "roygb" (default rainbow), "bwr" (blue-white-red), "rwb" (red-white-blue), "low" (red-green), or "high" (yellow-blue) prior to Jmol 11.4; it may be any defined color scheme in Jmol 11.4. In Jmol 11.4, this parameter is largely replaced by **color "schemeName"**. Jmol 12.0 adds "bw" (black-white) and "wb" (white-black).

set propertyDataColumnCount (integer)

An integer value of 0 or greater. Sets the number of columns to be read for **propertyDataField**.

set propertyDataField (integer)

An integer value of 0 or greater. Sets the column (when **propertyAtomNumberColumnCount** > 0) or free-format field (otherwise) for the property data in a [data](#) set. Setting this value to 0 indicates that values are simply to be read sequentially from the data.

set quaternionFrame A,B,C,N,P,Q,RC,RP,X

Specifies the axes used to define the right-handed [quaternion](#) frame for proteins and nucleic acids. (Jmol 11.6) These frames (xyz axes and origin) define orientations of amino acid residues in proteins and nucleic acid residues in RNA and DNA. The minimum definition of a frame is an origin (only used for [draw quaternion](#)), a point that will define an axis (usually X, but in some cases Y), and a third point that will be used to generate the other two axes. Specifically, if O, A, and B are three points and VO<sub>A</sub> defines the direction of the X axis, then V<sub>C</sub> = VO<sub>A</sub> x VO<sub>B</sub> defines the direction of the Z axis, and V<sub>C</sub> x VO<sub>A</sub> defines the direction of the Y axis. If VO<sub>B</sub> defines the direction of the Y axis, then V<sub>C</sub>, as above, defines the direction of the Z axis, and VO<sub>B</sub> x V<sub>C</sub> defines the direction of the X axis. Different frames can be used for different purposes, ranging from analyzing amino acid side-chain orientations to quantifying the "straightness" of helices and sheets. Most useful are relative and absolute differences of quaternions, which define local helical axes or the axis, angle, and translation required to move from one orientation to another.

A	"alternative"	for proteins, same as N (see below)	for nucleic acids, an alternative backbone quaternion with O = P[i], B(Y) = C4'[i], and A = C4'[i-1].
---	---------------	-------------------------------------	---

			which is closely associated with eta and theta dihedral angles.
B	"backbone"	for proteins, O = CA[i], A(X) = CA[i+1], B = CA[i-1]	for nucleic acids, O = P[i], A(X) = P[i+1], B = P[i-1]
C	"carbon"	for proteins, O = CA, A(X) = C, and B = N, thus defining the orientation of the peptide sidechains	for nucleic acids, same as N, but with the origin shifted to the "center of heavy atoms" for the base (all atoms associated with the base other than H or C4')
N	"nitrogen"	a frame based on the peptide N atom and useful specifically in the area of solid state NMR spectroscopy. O = N, and if V <sub>C</sub> = V <sub>N-NH</sub> , and V <sub>D</sub> = V <sub>N-CA</sub> , then V <sub>B</sub> (Y) = V <sub>A</sub> x V <sub>D</sub> , M = axisAngle(V <sub>B</sub> , -17 degrees), then V <sub>A</sub> = MV <sub>C</sub> , and V <sub>C</sub> = V <sub>A</sub> x V <sub>B</sub> defines the direction of the Z axis.	for nucleic acids, O = N9(purines) or N1(pyrimidines), B(Y) = O + V <sub>C1'O</sub> , and A = VO <sub>C2</sub> (toward the Watson-Crick edge)
P	"peptide/phosphorus"	for proteins, O = C, A(X) = CA, and Y = N[i+1], thus defining the orientation of the peptide plane	for nucleic acids, O = P[i], A(X) = O3'[i-1], and B = O5'[i], thus defining the orientation of the phosphorus tetrahedron
Q	"Quine"	defined as follows: O = (C[i] + N[i+1])/2 and if V <sub>X</sub> = V <sub>CA[i]-C[i]</sub> and V <sub>B</sub> = V <sub>N[i+1]-CA[i+1]</sub> , then V <sub>Z</sub> = V <sub>X</sub> x V <sub>B</sub> , and V <sub>Y</sub> = V <sub>Z</sub> x V <sub>X</sub> . This frame was an early definition of the orientation of the peptide plane and suffers from the fact that these two vectors are very nearly colinear, and can produce Z directions that are misdirected; provided for historical reference only (J. R. Quine, Journal of Molecular Structure: THEOCHEM, Volume 460, Issues 1-3, 26 February 1999, pages 53-66).	not used for nucleic acids
RC and RP	"ramachandran"	These frame selections specify that straightness should be calculated from Ramachandran angle approximations rather than actual quaternionFrames "C" and "P", respectively	no nucleic acid equivalent
X	"experimental"	Jmol development testing	Jmol development testing

set rangeSelected

Starting with Jmol 11.0, when this flag is set to TRUE, the range for [color](#) temperature is set to be adjustable based on the selected set of atoms being colored; when false (the default), the range of colors is set based on the range of values in the entire model. Starting with Jmol 12.0, this parameter applies to any property.

set repaintWaitMs 1000

Sets the number of milliseconds to wait before a timer expires signalling that Jmol should not wait any longer for a repaint operation. In some large memory-intensive operations, it is sometimes advisable to set this parameter to a higher number.

set saveProteinStructureState TRUE

Generally when Jmol [writes](#) the state, helix/sheet/turn data is saved. In some cases this may not be desired. Setting this flag FALSE prevents the saving of this information in the state.

set selectAllModels TRUE

Generally when selections are made in Jmol all matching atoms in all models are selected, regardless of which model is currently in [frame](#). When set FALSE, this flag indicates that the subset of atoms to select should be automatically set to the current model when frame changes occur.

set selectHetero ON

When turned OFF, selections do not select HETATM atoms in PDB files

set selectHydrogen ON

When turned OFF, selections do not select hydrogen atoms

set showKeyStrokes TRUE

When set TRUE, and [set allowKeyStrokes](#) is TRUE, key strokes in the applet or application window to be interpreted as script commands and displayed in the bottom left corner of the window.

set smartAromatic ON

Turning the smartAromatic parameter OFF reverts to a Jmol 10 style of drawing aromatic bonds as a paired solid and dashed line when loading subsequent files. (This command has no immediate effect. Use [reset aromatic;calculate aromatic](#) to see the effect of smartAromatic; Jmol 11.4)

set spinFps [[frames-per-second](#)]

determines the number of frames per second between displays of the molecule -- a small number here results in a jerky stepwise rotation.

set spinX [[degrees-per-second](#)]

The **set spinX**, **set spinY**, and **set spinZ** commands allow for the setting of the spin axes -- x, y, and z -- independently as well as the rate of spin. The actual spinning axis is a complex combination of the three settings. No actual spinning occurs until the **spin ON** command is issued or the user turns spinning on using the mouse menu. Jmol 11.0 adds a much richer variety of spinning possibilities to this Chime/Rasmol-standard capability, allowing simple spinning around arbitrary molecular- and window-based axes. See the **rotate** command.

set spinY [[degrees-per-second](#)]

see **set spinX**

set spinZ [[degrees-per-second](#)]

see **set spinX**

set stateVersion (integer)

Displays the version of Jmol used to create the most recently run state script (a parameter rather than a read-only variable only because a script has to create it).

set statusReporting ON

When set OFF, this parameter prevents Jmol from recording status messages and reporting them to a requesting web page. When OFF, the JavaScript method jmolScriptWait() cannot return status from the Jmol applet.

set stereoDegrees (decimal)

Same as **stereo (degrees)**, sets the angle for stereo images; can be checked using Jmol math.

set strutDefaultRadius 0.3

For **calculate STRUTS** and the **struts** command, sets the default radius for new struts. (Jmol 12.0)

set strutLengthMaximum 7

For **calculate STRUTS**, sets the maximum length for a new strut. (Jmol 12.0)

set strutsMultiple OFF

For **calculate STRUTS**, turn this flag ON to allow more than one strut on a given atom. (Jmol 12.0)

set strutSpacing 6

For **calculate STRUTS**, sets the minimum spacing (in terms of residues) for new struts. (Jmol 12.0)

set syncMouse OFF

When **sync** is ON, delivers mouse actions to the target applet.

set syncScript OFF

When **sync** is ON, delivers script events to the target applet.

set useMinimizationThread ON

Generally the **minimize** command runs in a separate thread. Setting this parameter FALSE instructs Jmol to use the same thread as is being used for commands. This is important in situations where one wants to wait for completion of the minimization before continuing.

set useNumberLocalization ON

Currently only applicable in the display of unit cell parameters, the **useNumberLocalization** setting determines whether or not local number formatting will be used (comma as decimal point, for example).

set vectorScale (decimal)

Sets the scale for vibration vectors.

set vibrationPeriod (decimal)

Sets the default period of vibrations (in seconds). Setting this value to 0 disables vibration modeling.

set vibrationScale (decimal)

Sets the amplitude of vibrations.

set waitForMoveto ON

Setting this parameter OFF allows **moveTo** operations to continue independently while script commands are processing. An ongoing moveTo operation can then be stopped at any time using **moveTo STOP** [Jmol 12.0]

set wireframeRotation OFF

Turning this parameter ON sets Jmol to not display spacefill and only display bonds as simple lines during user-based model manipulation.

where

[[frames-per-second](#)] is an (integer)

[[degrees-per-second](#)] is an (integer)

See also:

[animation](#) [data](#) [draw](#) [frame](#) [invertSelected](#) [model](#) [move](#) [moveto](#) [rotateSelected](#) [show](#) [spacefill](#) [spin](#) [translate](#) [translateSelected](#) [write \(model\)](#) [write \(object\)](#)  
[zoom](#) [zoomto](#)

[variables](#) [top](#) [search](#) [index](#)

## set (navigation)

This commands in this group set various parameters in relation to navigating through the model. See [pdf documentation](#) for details regarding the Jmol perspective/navigation model.

set hideNavigationPoint FALSE

When navigating, a small square-with-crosshairs pointer usually appears to help guide the user. Setting this parameter TRUE hides that pointer.

set navFPS (integer)

Sets the nominal speed of [navigation](#). Default value is 10.

set navigateSurface FALSE

An experimental setting that restricts navigation to the inside of a surface. Not intended for general use.

set navigationDepth (percent)

Sets the depth of the observer navigation point using the same basis as standard slab and depth -- 0 being the back plane of the model; 100 being the front plane of the model. Values greater than 100 represent observation points in front of model.

set navigationMode FALSE

When set TRUE, enables user-directed navigation through the model using the keypad arrow keys according to the following table. Setting this parameter TRUE automatically sets the perspectiveStyle to 11, sets zoomEnabled, and sets perspectiveDepth on.

UP/DOWN arrows	go forward/back off
LEFT/RIGHT arrows	turn left/turn right
ALT UP/DOWN arrows	pitch upward or downward
SHIFT LEFT/RIGHT/UP/DOWN arrows	shift navigation point on the screen and in relation to the model

set navigationPeriodic FALSE

When set TRUE, and the model has a unit cell defined and symmetry has been applied using **load {i j k}**, creates the effect of an limitless navigation. The navigation center is kept always within the unit cell. The more full unit cells that have been loaded, the more realistic the effect.

set navigationSpeed (integer)

Sets the rate of forward/reverse navigation when using the up/down arrow keys. The default value is 5.

set navigationSlab (percent)

Sets the depth of the depth of the slab plane relative to the observer navigation point using the same basis as standard slab and depth -- 0 (default) being at the navigation point; positive numbers being toward the user relative to that.

set navX (decimal)

Sets the navigation turning rate in the X direction (yaw) in units of 1/50 degrees per frame.

set navY (decimal)

Sets the navigation turning rate in the Y direction (pitch) in units of 1/50 degrees per frame.

set navZ (decimal)

Sets the navigation rate in the Z direction (forward motion) relative to its standard value of 1.

set showNavigationPointAlways FALSE

When set TRUE, the navigation cross-hair cursor is shown whenever in navigation mode, not just while the user is pressing keypad keys or scripted navigation is occurring.

set visualRange (angstroms)

The visual range is the MINIMUM distance in Angstroms that is viewable by the observer. The default value of 5.0 indicates that with **set navigationMode TRUE** any object in a plane that is less than 5.0 Angstroms across will be clipped automatically under the presumption that that object is behind the observer. This prevents large objects close to the user from eclipsing smaller objects further from the observer so as to give the illusion of having entered the model itself and now being within it.

See also:

[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [moveto](#) [navigate](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#) [zoomto](#)

[variables](#) [↑top](#) [🔍search](#) [📖index](#)

## set (perspective)

This commands in this group determine the overall lighting effects, size, and rotation for the model. Note that in a multi-applet environment, changing lighting values (ambientPercent, diffusePercent, specular, specularExponent, specularPercent, or specularPower) changes them for ALL applets. Effect on another applet may not appear until that model is rotated by the user or some command is issued to that applet that requires updating the display.

set cameraDepth (positive number)

Sets the amount of perspective. The Jmol default is **set cameraDepth 3.0**, but you can adjust that to your liking. Smaller numbers lead to more extreme perspective distortion; higher numbers minimize the effect of perspective. The scale in the vertical plane midway from front to back of the model is identical to with **set perspectiveDepth off** at any zoom level; if **p** the relative distance of a vertical plane from front (0) to back (1) of the model at a zoom of 100, then the scaling factor is (cameraDepth + 0.5) / (cameraDepth + p) for that plane. This gives, for cameraDepth=3.0, 1.17 for p=0, 1.0 for p=0.5 (as for all cameraDepths), and 0.87 for p=1.

set perspectiveDepth ON

Turning this parameter OFF turns off perspective depth. OFF is required for proper function of absolute scale (**set scaleAngstromsPerInch nnn**). Perspective depth is automatically turned off by Jmol when loading a model having a unit cell and symmetry operators and automatically turned ON otherwise.

set perspectiveModel 11

Jmol's perspective model involves a linear perspective that is required for navigation mode. Setting this parameter to 10 returns Jmol to the former perspective model used in Jmol 10. When **set navigationMode** is invoked, the perspective model is automatically set to 11.

set scaleAngstromsPerInch [\[viewing-distance\]](#)

Sets the absolute scale of the model by setting the viewing distance from the user to the model in arbitrary units. The actual scale will depend upon the sizes of both the applet window and the screen.

set rotationRadius (Angstroms)

Sets the nominal rotation radius for the model effectively setting the size of the model at **zoom 100**. Normally set to the radius that will contain within the screen the entire model when rotated relative to the default rotation center.

set windowCentered ON

Turning this parameter OFF allows one to set the center of rotation to a new position without also moving that position to the center pixel of the applet window.

set zoomEnabled ON

When turned OFF, this parameter disables zooming. Same as **zoom ON/OFF**

set zoomLarge ON

When ON (Jmol default setting), Jmol adjusts the zoom based on the LARGER of the two application/applet dimensions, height and width. Turning this parameter OFF causes zoom to be based on the smaller of the two dimensions, which may be useful for the applet if the user is allowed to resize the applet window.

set zShade OFF

**set zshade ON** enables a "fog" or "fade" effect, which shades objects based on the distance from the observer such that distant objects fade into the background. Uses the value of the SLAB and DEPTH settings to determine the point of no effect and full effect, respectively (by default 100 and 0). The effect is enabled regardless of the setting of (**slab ON/OFF**) and is reset to OFF upon **reset** or the loading of a new model. (Jmol 11.8)

where

**[viewing-distance]** is the apparent distance from the model to the user in arbitrary units -- (integer), (decimal)

See also:

[backbone](#) [background](#) [cartoon dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

[variables](#) [↑top](#) [🔍search](#) [📖index](#)

## set (structure)

(v. 11.0 -- several new customizable features)

This command group allows for customization of the rendering of PDB and mmCIF secondary structures. The proposed Jmol 11.0 default is **set cartoonRockets OFF**; **set ribbonAspectRatio 16**; **set hermiteLevel 0**; **set ribbonBorder 0**; **set sheetSmoothing 1**; **set strands 5**; **set traceAlpha ON**.

set cartoonBaseEdges FALSE

Setting this parameter ON displays nucleic acid bases as triangles that highlight the sugar edge (red), Watson-Crick edge (green), and Hoogsteen edge (blue). See Nasalean L, Strombaugh J, Zirbel CL, and Leontis NB in **Non-Protein Coding RNAs**, Nils G. Walter, Sarah A. Woodson, Robert T. Batey, Eds., Chapter 1, p 6.

set cartoonRockets OFF

Setting this parameter ON sets the display of RasMol **cartoons** to be a mixture of three-dimensional sheet cartoons and alpha helix **rockets**. It is not a precise as a cartoon with respect to helices but more precise than rockets in relation to beta-pleated sheets.

set hermiteLevel (integer, -8 to 8)

Determines the amount of curve smoothing used in rendering protein and nucleic acid secondary structures involving **cartoon**, **ribbon**, **rocket**, and **trace**. **set hermiteLevel 0** uses a Jmol 10.2 method of rendering these structures, with rope-like traces and paper-thin ribbons and cartoons. Positive values produce more rounded but slower to render shapes, but only when the model is not in motion. Negative numbers produce the same, but also while rotating. A value of 4 or -4 might be a reasonable compromise between look and performance.

set highResolution OFF

When set ON, and hermiteLevel is set, draws high-resolution hermite curves even if the diameter is small. Otherwise, small-diameter traces are shown in a faster-rendering fashion.

set ribbonAspectRatio (integer)

Sets the thickness of the ribbons in ribbon and cartoon renderings in terms of the width:height aspect ratio; only enabled in conjunction with **set hermiteLevel** to a non-zero value. **set ribbonAspectRatio 0** turns off this feature; 8-16 is recommended; higher positive numbers leading to thinner ribbons.

set ribbonBorder OFF

Turning this parameter ON adds a slight border for ribbons.

set rocketBarrels OFF

Turning this parameter ON removes the arrow heads from **rockets** and **cartoon rockets**, turning them into simple cylindrical barrels.

set sheetSmoothing (0 to 1)

In conjunction with **set traceAlpha**, this parameter determines the "waviness" of beta-pleated sheets. **set sheetSmoothing 0** creates wavy sheets, with the ribbon or trace going directly through the alpha carbons. The default is **set sheetSmoothing 1**, which produces a more averaged, smoother (standard) ripple. Has no effect unless **set traceAlpha**.

set strandCount [\[strand-count\]](#)

Sets the number of strands to display for the **strand** and **meshribbon** shapes both, with a maximum of 20.

set strandCountForMeshRibbon [\[strand-count\]](#)

Sets the number of strands to display for the **meshribbon** shape, with a maximum of 20.

set strandCountForStrands [\[strand-count\]](#)

Sets the number of strands to display for the **strand** shape, with a maximum of 20.

set traceAlpha TRUE

When set TRUE (the default), along with **set sheetSmoothing 0**, directs Jmol to draw **traces** directly through all alpha-carbons. The effect is a wider, more standard alpha helix and a wavier beta-pleated sheet than in Jmol 10.2. Setting **set sheetSmoothing 1**; **set traceAlpha ON** directs Jmol to create smooth out the beta-pleated sheets but still follow the alpha carbons for other structure types. With **set traceAlpha FALSE**, Jmol draws traces through the midpoints of the lines connecting adjacent alpha-carbons, producing tighter alpha helices and smoothed beta-pleated sheets. Also used as the basis for drawing cartoons, meshRibbons, ribbons, rockets, and strands. The sequence **set traceAlpha off**; **set sheetSmoothing 1**; **set hermiteLevel 0** is equivalent to the Jmol 10.2 default. Jmol 11.4 has the default settings: **set traceAlpha TRUE**; **set sheetSmoothing 1**; **set hermiteLevel 0**.

where

**[strand-count]** is the number of strands -- (integer, 0 to 20)

[variables](#) [↑top](#) [🔍search](#) [📖index](#)

## set (visibility)

(v. 11.0 -- several new features)

This command group turns on or off specific sets of atoms and axes/cell-related options.

set axes [\[line-width-or-type\]](#)  
 See [axes](#).  
 set axesMode 0, 1, or 2  
 Allows setting and checking of the current axes mode: (0) window, (1) molecular, (2) unitcell (Jmol 11.4)  
 set axesMolecular OFF  
 See [axes](#); this parameter can be set and checked using Jmol math.  
 set axesScale (decimal)  
 Allows setting of the axes scale; (default 2; Jmol 11.4)  
 set axesUnitcell OFF  
 See [axes](#); this parameter can be set and checked using Jmol math.  
 set axesWindow ON  
 See [axes](#); this parameter can be set and checked using Jmol math.  
 set backgroundColor [\[RGB-color\]](#)  
 Sets the background color to the specified value; can be inspected using Jmol math.  
 set boundingbox [\[line-width-or-type\]](#)  
 See [boundingbox](#).  
 set boundingboxColor "color\_name"  
 This parameter sets the default boundingbox color and also can be read as the current boundingbox color.  
 set defaultTranslucent (decimal)  
 Sets the default translucent level (0.5 is standard).  
 set disablePopupMenu FALSE  
**set disablePopupMenu TRUE** disables the pop-up menu.  
 set displayCellParameters TRUE  
 Turning this parameter FALSE turns off the unit cell parameter list that is included with [unitcell ON](#).  
 set greyScaleRendering OFF  
 Setting this parameter ON displays all colors as grey scale values.  
 set hideNameInPopUp FALSE  
 Setting this parameter TRUE hides the file name and contents in the pop-up menu starting with the NEXT FILE LOADED.  
 set hideNotSelected FALSE  
 Setting this parameter TRUE tells Jmol to hide any atoms whenever they are not selected. **set hideNotSelected TRUE** immediately hides the currently unselected atoms. When turned off, no immediate action is taken, but future selections no longer hide atoms. The flag is automatically cleared when a new model is loaded or when the picking mode is set to any sort of selection (atom, group, etc.).  
 set refreshing TRUE  
 When **refreshing** is set FALSE, no writing to the screen is done at all. Setting refreshing to TRUE should only be done in cases where there is a desire to not show intermediate results as a valid well-tested script runs. If the script throws an error, and refreshing is FALSE, Jmol may appear to be frozen when in fact it is not.  
 set showAxes FALSE  
 Setting **showAxes** TRUE displays the axes at their default line width; setting it false hides the axes.  
 set showBoundingBox FALSE  
 Setting **showAxes** TRUE displays the bounding box as a thin dotted line; setting it false hides the bounding box.  
 set showFrank TRUE  
 Setting **showFrank** TRUE displays the Jmol frank in the lower right-hand corner. Checking this parameter allows checking to see if the frank is on.  
 set showHiddenSelectionHalos FALSE  
 When both this parameter is ON and [selectionHalos](#) are ON, Jmol will display selection halos even if atoms are hidden.  
 set showHydrogens TRUE  
 Turns on and off display of hydrogen atoms.  
 set showSelections FALSE  
 deprecated -- see [selectionHalos](#)  
 set showUnitcell FALSE  
 Setting this parameter TRUE is equivalent to [unitcell ON](#); can be tested using Jmol math.  
 set slabByAtom FALSE  
 Setting this flag TRUE removes whole atoms and bonds when they are partially clipped. (Jmol 12.0)  
 set slabByMolecule FALSE  
 Setting this flag TRUE removes whole molecules when they are partially clipped. (Jmol 12.0)  
 set slabEnabled FALSE

Setting this parameter TRUE is equivalent to [slab ON](#); can be tested using Jmol math.  
 set solventProbe OFF

Turning this parameter ON turns on a solvent "probe" that can be displayed using dots. After **set solventProbe ON**, a subsequent [dots ON](#) shows the surface of the aggregate of selected atoms using dots. This surface is defined by the contact of a spherical probe (representing a solvent molecule) rolled over the surface of the selected atoms. The radius of the probe sphere of 1.4 Angstroms approximates a water molecule. The default radius for Jmol is 1.2 Angstroms, which can be changed using **set solventProbeRadius**. Note that no change in display occurs after **set solventProbe ON** until the next [dots ON](#) command is encountered. Does not effect the function of [isosurface solvent](#), which draws a surface regardless of the setting of this flag. For a detailed discussion of molecular surfaces, see <http://www.netsci.org/Science/Compchem/feature14.html>.

Examples: [in new window using 1crn.pdb](#)



```
set radius 2.0
set solvent ON
select 1-10
dots ON
```

set solventProbeRadius [\[probe-radius-in-angstroms\]](#) {default: 1.2}

Sets the radius of the solvent "ball" that would roll around the structure defining its outline. After **set radius**, you must (re)issue [dots ON](#) for it to take effect, and the solvent probe option for dots must be set on using **set solvent ON** (below).

set unitcell

See [unitcell](#).

set unitCellColor "color\_name"

This parameter sets the default unit cell color and also can be read as the current unit cell color.

where

[\[line-width-or-type\]](#) is a line width or type for a drawing object -- ON, OFF, DOTTED, (integer, 1 to 19), (decimal, <2.0)  
[\[RGB-color\]](#) is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [xRRGGBB]  
[\[probe-radius-in-angstroms\]](#) is a (decimal)

See also:

[axes backbone background boundingbox cartoon dots ellipsoid geoSurface meshribbon ribbon rocket set \(highlights\) set \(lighting\) set \(navigation\) set \(perspective\) spacefill strand trace vector wireframe](#)

[variables](#) [↑top](#) [🔍search](#) [📖index](#)

## set echo

Sets positioning and visibility parameters associated with text written to the screen using the [echo](#) command. In addition, Jmol 11.6 adds **set echo IMAGE**, allowing images to be displayed instead of text. Characteristics of the text that are settable include:

font size, face, style, and scaling factor	See <a href="#">font ECHO</a> .
font color	Includes translucency. See <a href="#">color ECHO</a> .
background color	Includes translucency. See <a href="#">background ECHO</a> .
model	Starting with Jmol 11.4, echo text can be associated with a specific model, with visibility controlled by the <a href="#">frame</a> command. See below.
x and y position within the window	A "2D" echo. A block of text can be displayed anywhere within the applet or application window. An automatic adjustment is made to prevent text from going outside of the bounds of the window. See below.
TOP, MIDDLE, BOTTOM	Three special 2D positions are also defined for quick placement of text. These three special echos can only have one of three predefined positions: LEFT, CENTER, or RIGHT.
depth	The default Z-position for echoed text and images is in front of the model. However, this can be set to any <a href="#">depth</a> in relation to the model, from 0 (rear) to 100 (front). See below.
x, y, and z position within the model itself	A 3D echo. This position can be defined as a point in space, the coordinate of an atom, the average coordinate of a set of atoms, the position of a <a href="#">drawn point</a> , or the average position of a drawn object. See below.

text alignment	echoed text can be aligned LEFT, CENTER, or RIGHT, with multi-line text aligning line by line. With Jmol 11.6, <b>set echo center</b> also aligns multi-line text and images vertically. See below.
IMAGE	Starting with Jmol 11.6, one can also place JPEG, PNG, or GIF images at either a 2D screen position or a 3D molecular position. See below.
scaling	Using the command <a href="#">set fontScaling TRUE</a> 3D-positioned images and text along with atom labels can be made to scale when <a href="#">zoom</a> is applied. The current zoom setting is used to fix a "scaling factor" used for the font. However the <a href="#">font</a> command can be used to set a different reference scaling factor.
visibility	Prior to Jmol 11.6, echoed text could only be turned on. <b>set echo off</b> simply deletes the echo. Jmol 11.6 adds the capability to display or hide echos using either the <b>set echo HIDDEN/DISPLAYED</b> command or the <a href="#">hide</a> and <a href="#">display</a> commands with \$name, where name is the name given an echo in the <b>set echo</b> command (including the special echos TOP, MIDDLE, and BOTTOM using \$top, \$middle, and \$bottom, respectively).

Default settings for font, color, and background can be set by first issuing **set echo none**, so that no real echo is set, and then issuing the desired [font](#), [color](#), and [background](#) commands. In addition, any changes to these settings for any specific echo text become the new default for future text echos that are created for the same model.

set echo user-named [[horizontal-position](#)] {default: left}

Selects the horizontal text alignment (LEFT, CENTER, or RIGHT) for a user-positioned text echo. Any text already displayed in this position is immediately realigned. Starting with Jmol 11.6, the CENTER option also centers the text box (or image) vertically at the 2D or 3D point defined for this echo.

set echo user-named [x y]

Sets a custom echo position with the given name, where [0 0] is the bottom left corner. The next-issued [echo](#) command will write text to this position. Any text already displayed is immediately moved. If the text would flow out of the applet window, it is repositioned just inside the boundary.

set echo user-named [x y %]

Sets echo position based on a percentage of the applet window size. Note that **set echo myecho [0 100%]** is not quite the same as **set echo TOP LEFT**; **set echo myecho [100 0%]** is not quite the same as **set echo BOTTOM RIGHT**. The difference is that TOP LEFT and BOTTOM RIGHT automatically justify multiline text as well. Using percentages allows you to place text anywhere within the window and independently justify the text using **set echo myecho LEFT**, **set echo myecho RIGHT**, or **set echo myecho CENTER**.

set echo user-named {x y z}

Sets echo position in three dimensions, based on molecular coordinates (which may be fractional). Braces are required.

set echo user-named { [[atom-expression](#)] }

Sets echo position in three dimensions, based on the geometric center of the specified atoms. Braces are required. "OR" should be used rather than "," in the atom expression unless additional parentheses within the braces are also provided.

set echo user-named DEPTH %z

Sets the depth of the echoed text or image in percent of the model diameter (0 rear, 100 front).

set echo name DISPLAYED

Selectively displays an echo that previously has been hidden. The name can be any defined echo, including TOP, MIDDLE, or BOTTOM.

set echo user-named IMAGE "file name"

Reads the specified JPG, GIF, or PNG file and displays that image instead of text. All 2D and 3D options are available. Images can be scaled the same as text using the [font](#) command's scaling factor option.

set echo user-named MODEL (model number)

By default, when multiple models are present, all echo text is visible regardless of the model currently displayable using the [frame](#) command. The **set echo ... MODEL** command associates an echo with a specific model, allowing the text or image to appear and disappear when different models are displayed using the [frame](#) command.

set echo user-named SCRIPT "script"

Specifies a Jmol script that will be run when the echo is clicked.

set echo name HIDDEN

Selectively hides an echo without deleting it. The name can be any defined echo, including TOP, MIDDLE, or BOTTOM.

set echo ALL

Sets the current echo to be "all echos" for setting font, color, display, hidden, or text of all echos at once.

set echo DISPLAYED

Unhides the current echo -- or all echos if echo has been set NONE.

set echo HIDDEN

Hides the current echo -- or all echos if echo has been set NONE.

set echo NONE

Sets the current echo to be "none" -- The next **echo** command will be to the console/message callback only.

set echo OFF

Turns off (deletes) all echo texts. (Starting with Jmol 11.6, see also **set echo hidden**.)

where

**[horizontal-position]** is one of the following: -- LEFT, CENTER, RIGHT

**[atom-expression]** is any [expression](#) that evaluates to a set of atoms

See also:

[echo font](#) [hover label](#) [set \(highlights\)](#) [set \(labels\)](#)

[variables](#) [top](#) [search](#) [index](#)

## set modelKitMode

(v. 12.0 -- new [modelKitMode](#))

Jmol 12.0 includes a mode that allows building and modifying models, with minimization. Features include the ability to easily add, drag, and delete atoms, drag and minimize molecules and molecular fragments such as CH<sub>3</sub> groups, rotate bonds, minimize parts of models -- all mouse-driven. A special menu is employed that sits at the top left corner of the Jmol window. (Clicking the Jmol frank still brings up the standard menu.) After **set modelKitMode**, a [zap](#) command creates a CH<sub>4</sub> model, and the picking mode goes to **set picking assign\_C**.

[variables](#) [top](#) [search](#) [index](#)

## set picking

(v. 12.0 -- adds [set picking measure SEQUENCE](#))

The **set picking** command determines the response to clicking of atoms by the user. For those options for which they apply, the keywords **MEASURE** and **SELECT** are optional.

set picking ON

Setting this paramter OFF turns off the sending of picking information (atom identifier, atom number, and atomic position) to both to the status line and to the [PickCallback](#) function, if defined.

set picking CENTER

When the user clicks an atom, that atom is set to be the center of rotation. If windowCentered is true (default), then this atom is smoothly moved to the window center; if not windowCentered, then the atom stays in position, but now all rotation is around it and if perspectiveDepth is set (**set windowCentered off;set perspectiveDepth on**), the perspective around this atom shifts to indicate that it is now the focus of the perspective. If the same atom is clicked twice, it is zoomed in on by a factor of two. If no atom is clicked, then the model is zoomed out by a factor of two.

set picking CONNECT

Performs like measuring distances, except bonds are created. (Jmol 12.0)

set picking DELETEATOM

Clicking on an atom deletes it. (Jmol 12.0)

set picking DELETEBOND

Deletes the bond between pairs of clicked atoms (and with **set BONDPICKING TRUE**, deletes bonds as they are clicked.. (Jmol 12.0)

set picking DRAGATOM

Clicking and dragging an atom moves its position.

set picking DRAGMINIMIZE

Clicking and dragging an atom moves its position and then does a minimization of its associated molecule.

set picking DRAGMINIMIZEMOLECULE

Clicking and dragging an atom moves and minimizes its associated molecule. This setting can be used to simulate docking.

set picking DRAW

The vertices of all [draw](#) objects are highlighted. Holding the SHIFT key down while dragging moves the object to a new location ("shifts" the object); holding ALT down while dragging moves a single vertex to a new location ("alters" the shape of the object). The draw command required to reproduce the object can be seen immediately using the Java console. This new command can also be seen via message callback, or in the Jmol [console](#) using [show DRAW](#).

set picking IDENT  
 Same as **set picking ON**.

set picking INVERTSTEREO  
 Selecting an atom that is part of a ring after **set picking invertStereo** will reverse the two non-ring atoms -- actually rotating them 180 degrees, not doing a planar inversion, thus preserving whatever chirality might be attached to them. (See also [invertSelected STEREO](#)).

set picking LABEL  
 When the user clicks an atom, the label of that atom is toggled on or off.

set picking MEASURE  
 Same as **set picking MEASURE DISTANCE** but also displays a distance measurement on the molecule.

set picking MEASURE DISTANCE  
 Turns picking on and returns atom identities and distance between two atoms. Three messages are sent to the [MessageCallback](#) function, if defined: Atom #1 (after the first click) and then Atom #2 and Distance (after the second click).

Examples:

```
(pdb data in this case)
Atom #1:[VAL]8.CA #49
Atom #2:[GLU]23.OE2 #169
Distance [VAL]8.CA #49 - [GLU]23.OE2 #19 : 16.765396
```

set picking MEASURE ANGLE  
 Turns picking on and returns atom identities and angle involving three atoms. Four messages are sent to the [MessageCallback](#) function, if defined: Atom #1 (after the first click), Atom #2 (after the second click), and then Atom #3 and Angle (after the third click).

Examples:

```
(xyz data in this case)
Atom #1:C 2 #2
Atom #2:C 3 #3
Atom #3:C 4 #4
Angle C 2 #2 - C 3 #3 - C 4 #4 : 122.3754
```

set picking MEASURE TORSION  
 Turns picking on and returns atom identities and torsion angle (dihedral angle) involving four atoms. Five messages are sent to the [MessageCallback](#) function, if defined: Atom #1 (after the first click), Atom #2 (after the second click), Atom #3 (after the third click), and then Atom#4 and Torsion (after the fourth click).

Examples:

```
(cml data in this case)
Atom #1:a7 #7
Atom #2:a3 #3
Atom #3:a1 #1
Atom #4:a2 #2
Torsion a7 #7 - a3 #3 - a1 #1 - a2 #2 : -4.15209
```

set picking MEASURE SEQUENCE  
 Turns picking on and returns the 1-letter-code sequence of atoms in bioSMILES format, showing chain and residue range and biomolecular type (p for protein, d for dna, r for rna). For example: **//\* chain A protein 36 //I ~p-PG //\* 37 //I**.

set picking NAVIGATION  
 User mouse clicking sets the navigation center to the selected position. See [Navigation](#).

set picking SELECT ATOM  
 When the user clicks an atom, the selection of that atom is toggled on or off.

set picking SELECT CHAIN  
 When the user clicks an atom, the selection of the chain associated with that atom is toggled on or off.

set picking SELECT GROUP  
 When the user clicks an atom, the selection of the group associated with that atom is toggled on or off.

set picking SELECT ELEMENT  
 When the user clicks an atom, the selection is toggled on or off for all VISIBLE atoms with the same element as the selected atom.

set picking SELECT MOLECULE

When the user clicks an atom, the selection is toggled on or off for all atoms within the same covalently bonded set as the selected atom.

set picking SELECT POLYMER  
 When the user clicks an atom, the selection is toggled on or off for all atoms within the same polymer unit (PDB only).

set picking SELECT SITE  
 When the user clicks an atom, the selection is toggled on or off for all VISIBLE atoms with the same crystallographic site as the selected atom.

set picking SELECT STRUCTURE  
 When the user clicks an atom, the selection is toggled on or off for all atoms within the same structural unit (helix, sheet, turn -- PDB only).

set picking SPIN [[frames-per-second](#)]  
 Turns picking on and sets it so that when the user clicks two atoms in succession, the model starts to spin around the axis defined by those two atoms. A third click on any atom stops rotation. Optionally, a speed of rotation can be included.

set picking STRUTS  
 When the user clicks two atoms, a [strut](#) is created between them.

set picking SYMMETRY  
 When the user clicks two atoms of a crystallographic file, a representation of their symmetry relationship is drawn, and the operator(s) relating these positions is written to the console.

where  
[\[frames-per-second\]](#) is an (integer)

[variables](#) [↑top](#) [🔍search](#) [📖index](#)

## set pickingStyle

(v. 11.0 -- NEW)

With **set pickingStyle** you can change the behavior of Jmol in response to user mouse actions relating to selecting atoms or measuring distances, angles, and torsions.

SELECT	Sets the behavior of Jmol to different clicking styles. For the standard Jmol default selection behavior, use <b>set pickingStyle SELECT toggle</b> . The actions of these options depend upon the setting of <a href="#">set picking SELECT</a> . If picking has not been set to SELECT, then this command has no immediate effect. In the explanations given below, it is presumed that we have <b>set picking SELECT GROUP</b> . The <b>SELECT</b> keyword is recommended but not required.
MEASURE	<b>set pickingStyle MEASURE ON</b> in conjunction with <b>set picking MEASURE</b> displays the measurements on the structure as they are generated by user clicking. If picking has not been set to MEASURE, then this command has no immediate effect. <b>set pickingStyle MEASURE OFF</b> returns to the default Jmol behavior.

set pickingStyle SELECT toggle  
 left-click toggles that group selected/not selected (Chime style; Jmol default).

set pickingStyle SELECT selectOrToggle  
 left-click selects just that group  
 shift-left-click toggles the group selected/not selected (Rasmol style).

set pickingStyle SELECT extendedSelect  
 User mouse action is according to the following scheme, which is the style used by [PFAAT](#).

left-click	selects just that group, like rasmol
shift-left-click	toggles the group selected/not selected
alt-left-click	appends the group to the current selection
alt-shift-left-click	removes the group from the current selection
left-click off model	executes ( <b>select none</b> )

set pickingStyle SELECT DRAG  
 makes the LEFT button a click-and-drag button when associated also with **set PICKING select (molecule, group, chain, etc.)** and **set dragSelected**.

set pickingStyle SELECT NONE  
 Returns to the Jmol default **toggle** picking style.

set pickingStyle MEASURE ON

Clicking atoms measures and displays distance, angle, or torsions based on the setting of **set picking MEASURE**. By default the user sees this information displayed. Setting **set pickingStyle measure OFF** when **set picking MEASURE** is set to DISTANCE, ANGLE, or TORSION tells Jmol to stop indicating measurements on the model when the user clicks, even though the distance, angle, or torsion information is still sent to the message queue.

[variables](#) [top](#) [search](#) [index](#)

## set userColorScheme

Sets a custom color scheme. Color values can be names such as "red" or "blue" or hexadecimal numbers as [xRRGGBB] or {r g b} triples in the form of 3D points. The scheme can then be referred to as "user"; it's reverse as "resu".

set userColorScheme colorName colorName

See also:

[background color \(atom object\)](#) [color \(bond object\)](#) [color \(element\)](#) [color \(model object\)](#) [color \(other\)](#) [color measures show](#)

[variables](#) [top](#) [search](#) [index](#)

## show

(v. 11.6 -- adds show moveto, show rotation, show translation, show atom, show chain, show group, show selected)

show sends information about the model to the [MessageCallback](#) function and to the Java or Jmol [console](#). If using the application, using **Jmol -ionx filename.spt modelfile.xyz > output.txt**, you can put a **show** command in a script file (filename.spt), and have the output directed to output.txt. The command line options used in this example include -i (silent startup), -o (use standard output), -n (no display), -x (exit after running the specified script file). All of the parameters that can be [set](#) can be shown. They+I635 are not listed here.

show ATOM

Lists the atom numbers that are currently selected in the order of atom index; duplicates are not listed twice.

show BOUNDBOX

Delivers the coordinates of the center coordinate and a directional vector defining a box just perfectly enclosing the model. The vector is determined by taking the min and max values for the atom along each cartesian axis. The center is the geometric center of the model, not the default center of rotation (which is the mean atom position). The eight corners of the bounding box are found by adding the center point to the vector, with all possible combinations of +/- component vectors. The length of a side of the bounding box is determined by doubling the appropriate component of the vector. So, for example, the length of the bounding box along the x-axis is (2\*vectorX). Units are in Angstroms. Output is in the form bounding box: (centerX, centerY, centerZ) (vectorX, vectorY, vectorZ)

show CENTER

Delivers the coordinates of the center of the model. Units are in Angstroms. Output is in the form center: (centerX, centerY, centerZ)

show CHAIN

Lists the chains (A, B, C, etc.) of the atoms that are currently selected in the order of atom index; duplicates are not listed twice.

show COLORSCHEME "name"

Displays the list of colors comprising a color scheme, where "name" is, for example, "roygb", "rwb", or "user". Starting with Jmol 11.4, four additional built-in color schemes include "byElement\_Jmol", "byElement\_RasMol", "byResidue\_Shapely" (corresponding to **color shapely**), and "byResidue\_Amino" (corresponding to **color amino**).

show DATA "type"

Displays the most recently read [data](#) of the given type. See also [getProperty data](#).

show DRAW

Displays the command required to generate the current [draw](#) object. Particularly useful after using [set picking DRAW](#).

show FILE

Delivers the entire contents of the file for the current model.

show FILE filepath

Delivers the entire contents of the specified file on the server from which the applet was loaded. The filename must be relative to the current page (not necessarily the directory containing the applet) and must be enclosed in quotation marks.

show ISOSURFACE

Displays the [JVXL](#) file equivalent. (Not available for all object types.)

show FUNCTIONS

Lists all user-defined functions.

show GROUP

Lists the groups (ALA, VAL, etc.) of the atoms that are currently selected in the order of atom index; duplicates are not listed twice.

show HISTORY n

Shows n lines of command history. If no number is given, **show history** by itself shows the full set of recorded command lines (100 maximum by default, but this maximum can be changed using the [set history](#) command. History recording can be turned on and off using the [history](#) command.

show MEASUREMENTS

Displays a table of information relating to [measurements](#) that have been made.

show MENU

Displays the current Jmol menu in a format that can be loaded using [load MENU](#).

show MO

Displays the [JVXL](#) file equivalent for the current [molecular orbital](#). If no MO is currently shown, displays the JVXL equivalent for the entire set of molecular orbitals. (This can take some time to construct.)

show MODEL

Delivers properties associated with the currently loaded model. Output includes information about all of the models in the set. This command is still in development. The exact form and content of the output is subject to change (and suggestion).

show MOVETO

Same as **show orientation moveto**

show ORIENTATION [optional type]

Delivers the orientation of the model. Output consists of both a "moveTo" command and an alternative sequence of "reset; rotate z; rotate y; rotate z" commands that would result in the current orientation. Thus, this command allows reading and restoring specific user-specified orientations. Starting with Jmol 11.6, optional types include:

moveto	the moveto command leading to this orientation, with no comments
rotation	the current rotation, as a <a href="#">quaternion</a>
translation	the current translation in percent X and percent Y from center of the window

show PDBHEADER

Delivers the PDB file header.

show POINTGROUP

Displays a table summarizing the point group of a symmetrical or nearly symmetrical model. See [calculate pointgroup](#) for details of this calculation.

show RESIDUES

Reports a listing of currently selected residues:

```
[GLY] 1 :A [ARG] 2 :A
[ARG] 3 :A
[ILE] 4 :A
[GLN] 5 :A
[GLY] 6 :A
[GLN] 7 :A
[ARG] 8 :A
[ARG] 9 :A
[GLY] 10 :A
[ARG] 11 :A
[GLY] 12 :A
```

show ROTATION

Same as **show orientation rotation**

show SELECTED

Lists the default label for all selected atoms, in the order of atom index.

show SEQUENCE

Reports the sequence of the currently selected residues in the following format:

```
Model 1
Chain A:
[GLY] 1 [ARG] 2 [ARG] 3 [ILE] 4 [GLN] 5
[GLY] 6 [GLN] 7 [ARG] 8 [ARG] 9 [GLY] 10
[ARG] 11 [GLY] 12 [THR] 13 [SER] 14 [THR] 15
[PHE] 16 [ARG] 17 [ALA] 18 [PRO] 19 [SER] 20
[HIS] 21 [ARG] 22 [TYR] 23 [LYS] 24 [ALA] 25
```

```
[ASP] 26 [LEU] 27 [GLU] 28 [HIS] 29 [ARG] 30
[LYS] 31 [VAL] 32 [GLU] 33 [ASP] 34 [GLY] 35
[ASP] 36 [VAL] 37
```

.

show SET  
Delivers a list of all "set" commands that have been issued, with their values.

show SMILES  
Delivers a SMILES string for the selected atoms. Assumes full molecules; does not report for biomolecules -- for that, use **print {selected}.find("SMILES", true)**.

show SPACEGROUP "name"  
Displays information relating to crystallographic space groups. If a file with crystallographic symmetry is loaded, **show spacegroup** by itself displays information for that spacegroup. Quotes are required around the space group name. If the name itself includes double quotes, use two single quotes instead. For example: P 32 2', not P 32 2".

show STATE [optional name]  
Delivers a Jmol script that, when run, will regenerate the state of the Jmol applet or application. The following objects are not yet supported: dipole, isosurface, lcaoCartoon, mo. If a name is given, then the state **saved** with that name is used; if no name is given, the state described is the current state.

show SYMOP (integer)  
Describes the specified symmetry operation. For example, **show symop 3** might give "-y,x-y,z-1/3 3-fold screw axis[translation: 0 0 -1/3". If no integer is given, this command delivers the full list of symmetry operations for the current frame.

show SYMOP [atom-expression-or-coordinate] [atom-expression-or-coordinate]  
Describes the symmetry operation relating the two specified points. (If an atom set is given, only the average position of the coordinates is used.) For example, **show symop {molecule=1} {molecule=2}** might give "-y,x-y,z-1/3 3-fold screw axis[translation: 0 0 -1/3". If more than one symmetry operation relates the positions, then all are described.

show SYMMETRY  
Displays information relating to the crystallographic symmetry of the model, including space group name and symmetry operations. (Applicable file formats only.)

show TIMEOUT  
Lists all pending **timeouts**.

show TRACE  
Reports a trace of function and script calls leading to the current command.

show TRANSFORM  
Delivers the current 3x3 transformation matrix (rotation only).

show TRANSLATION  
Same as **show orientation translation**

show UNITCELL  
Displays information relating to the crystallographic unit cell of the model. (Applicable file formats only.)

show URL  
Opens the data file in a new browser window. (applet only)

show URL URL  
Opens the specified URL in a new browser window. (applet only)

show VARIABLES  
Reports a listing of all variables and their values.

show ZOOM  
Delivers the current zoom setting. Output is in the form of the zoom command: "zoom n" where "n" is an integer percent of "normal" zoom.

show SubjectID  
For **draw** objects, displays the command that can be used to generate that object. For **isosurface** objects, displays the **JVXL** file equivalent. The \$ is required. (Not available for all object types.)

where  
[atom-expression-or-coordinate] is any **expression** surrounded by parentheses or braces, or any {x y z} coordinate

See also:  
[background](#) [boundbox](#) [center](#) [centerAt](#) [color \(atom object\)](#) [color \(bond object\)](#) [color \(element\)](#) [color \(model object\)](#) [color \(other\)](#) [color measures](#) [draw](#) [getProperty](#) [set \(misc\)](#) [set userColorScheme](#) [write \(model\)](#) [write \(object\)](#)

## slab

Slab and Depth together control the percentage of the molecule to be displayed based on clipping planes. **slab on** turns slab/depth on. **slab 50** shows the back 50% of the molecule. **slab 25** show the back 25% of the molecule. **depth** does the opposite of **slab**, hiding atoms far from the user. The default settings to see all of the model, then, are **slab 100; depth 0**. Atoms appear solid; bonds appear hollow. Starting with Jmol 11.2, slabbing can also be applied "internally" -- that is, based on molecular coordinates, not screen coordinates. Internal slabbing involves defining a plane ax + by + cz + d = 0 as {a b c d}, using Miller indices {h k l}, or using standard notation such as "x=3" or "xy".

slab ON/OFF {default: ON}  
Turns slab/depth on or off. Either **slab ON** or **depth ON** can be used; in either case both slab and depth are turned on.

slab [[slab-percent](#)]  
Sets the position of the slab/depth plane from 0 (front) to 100 (rear) of the model

slab HKL {h k l} or NONE  
Sets the position of the slab/depth plane based on the specified Miller index plane. For slabbing, with positive hkl values, atoms far from {0 0 0} are removed. The value NONE removes the slab/depth plane

slab -HKL {h k l}  
Sets the position of the slab/depth plane based on the specified Miller index plane; reference point is opposite **slab hkl** or **depth hkl**.

slab PLANE plane\_expression or NONE  
Sets the slab/depth plane based on a plane using the general syntax for [plane expressions](#). For example: **slab PLANE {atomno=3} {atomno=2} {atomno=1}; slab on**. Note that the order of points defines the direction from the plane of atoms to be removed. If using your right hand to define the path from first point to second to third, your right thumb points to the atoms removed by slabbing. (Opposite this for **depth**.)

slab -PLANE plane\_expression  
Reverses the sense of the plane defined as above.

slab RESET  
Resets slab and depth to the standard slab 100, depth 0, clears any internal planes, and turns slab on.

slab SET  
Sets the current slab to be internal, so that rotation of the model preserves the current slab from a molecular perspective.

where  
[[slab-percent](#)] is an (integer, 0 to 100)

Examples: [in new window using 1crn.pdb](#)

slab 50; depth 0;slab on; # show the back half of the molecule  
slab 100;depth 50; slab on;# show the front half of the molecule  
slab 75; depth 25;slab on; # show middle 50% of the molecule  
slab 50;depth 50;slab on; # show a plane that is 1 pixel deep

### Chime Note:

The slab/depth effect is equivalent to the RasMol command 'set slabmode solid', however 'set slabmode [option]' is not supported.

See also:  
[depth](#)

## spacefill or cpk (v. 11.0 -- adds solvent-accessible surface "+" option)

Renders selected atoms as shaded spheres. A boolean value renders the spheres with the van der Waals radius. A decimal value specifies the sphere radius in Angstroms. An integer followed by "%" specifies sphere radius as a percentage of the van der Waals radius. A "+" sign followed by a number specifies to draw the surface that number of Angstroms beyond the van der Waals radius. See also [dots](#) and [isosurface sasurface](#). Note that the value of the current spacefill setting can be retrieved using the .radius value for the atom. (For example, **print (atomno=3).radius.**) [Note: Jmol 12.0 changes the default Van der Waals radius to "AUTO" to allow non-PDB files and PDB files with H atoms to load with a slightly different look than PDB files with no H atoms. This brings Jmol's default parameter set in line with OpenBabel 2.2 and affects the default rendering also of [halos](#) and [stars](#).]

spacefill ON/OFF {default: ON}  
Turns spacefill on/off

spacefill ONLY  
Turns spacefill rendering on and all other rendering off.

spacefill AUTO  
The default setting, renders atom sizes for non-PDB files or PDB files with H atoms using OpenBabel 2.2 values.

spacefill [decimal]  
Specifying a decimal number generates a sphere at a specific radius in Angstroms. Starting with Jmol 12.0, a negative number also implies **ONLY**.

spacefill [integer] %  
Specifying an integer percent generates a sphere at the specified percentage of the van der Waals radius. (Starting with Jmol 11.6, this refers to the currently set van der Waals radius set -- Jmol, Babel, Rasmol, or User.)

spacefill [integer] %Jmol  
Specifying an integer percent generates a sphere at the specified percentage of the van der Waals radius as defined by Jmol 10 constants (OpenBabel 1.0). See [vdw\\_comparison.xls](#).

spacefill [integer] %Babel  
Specifying an integer percent generates a sphere at the specified percentage of the van der Waals radius as defined by OpenBabel 2.2. See [vdw\\_comparison.xls](#)

spacefill [integer] %Babel21  
Specifying an integer percent generates a sphere at the specified percentage of the van der Waals radius as defined by OpenBabel 2.1. See [vdw\\_comparison.xls](#)

spacefill [integer] %Rasmol  
Specifying an integer percent generates a sphere at the specified percentage of the van der Waals radius as defined by Rasmol. See [vdw\\_comparison.xls](#)

spacefill [integer] %User  
Specifying an integer percent generates a sphere at the specified percentage of the van der Waals radius as defined by the user using the [data](#) command.

spacefill +(solvent probe radius)  
With an explicit plus sign, **spacefill** draws the surface the given number of angstroms beyond the van der Waals radius. This is the definition of a solvent-accessible surface. Note that **spacefill +0** is the same as **spacefill 100%**. More typical would be **spacefill +1.2**.

spacefill IONIC  
Generates a sphere for each atom according to an approximation of its ionic radius.

spacefill TEMPERATURE  
Generates a sphere for eah atom according to its crystallographic B-factor. If Uij data has been read from a CIF file, then this value is set to 100 \* (U11 \* U22 \* U33)^0.333

spacefill ADPMIN n%  
Generates a sphere at the radius corresponding to the minimum anisotropic displacement parameter for the selected atoms factored by the given percentage. See also [ellipsoid](#).

spacefill ADPMAX n%  
Generates a sphere at the radius corresponding to the maximum anisotropic displacement parameter for the selected atoms factored by the given percentage. See also [ellipsoid](#).

See also:  
[backbone](#) [background](#) [cartoon](#) [data](#) [dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(misc\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [strand](#) [trace](#) [vector](#) [wireframe](#)

[top](#) [search](#) [index](#)

## spin

Starts and stops the molecule spinning around the axis determined by [set spinX](#), [set spinY](#), and [set spinZ](#). This command was greatly expanded in version 11.0 to include all the possible functions of [rotate](#).

spin ON/OFF {default: ON}

Examples: [in new window using caffeine.xyz](#)

spin branch {N8} {C19} 50 # CH3 starts spinning around the N8-C19 bond

See also:  
[animation frame](#) [invertSelected model](#) [move](#) [moveto](#) [rotate](#) [rotateSelected](#) [set \(misc\)](#) [translate](#) [translateSelected](#) [zoom](#) [zoomto](#)

[top](#) [search](#) [index](#)

## ssbonds

Cysteine disulfide bonds can be turned on or off, colored, and given customized widths in Angstroms.

ssbonds ON/OFF {default: ON}  
ssbonds [\[width-angstroms\]](#)  
ssbonds [\[width-Rasmol\]](#)

where  
[\[width-angstroms\]](#) is a (decimal, <2.0)  
[\[width-Rasmol\]](#) is in 1/250ths of an Angstrom (deprecated) -- (integer, <500)

See also:  
[bondorder](#) [connect](#) [hbonds](#) [set \(bond styles\)](#) [set \(files and scripts\)](#) [wireframe](#)

[top](#) [search](#) [index](#)

## star or stars

The **star** command places a set of crosshairs of a given size in Angstroms on the selected atoms. The default size of the star is, like spacefill, the nominal van der Waals radius for the atom. The default color for the star is that of the atom it is associated with. Use [color star \[colorname\]](#) to then color selected stars the color of your choice. For options, see [spacefill](#).

Examples: [in new window using 1crn.pdb](#)

```
select *.S?;star ON
select 40;star 1.0;color star red
select *.color star NONE
select *.star OFF
```

[top](#) [search](#) [index](#)

## step

Within the Jmol application, after a [pause](#) command, **step** executes only the next command and then pauses again. To see what the next command is without executing it, enter ? as the command.

See also:  
[delay](#) [exit](#) [goto](#) [loop](#) [pause](#) [quit](#) [resume](#)

[top](#) [search](#) [index](#)

## stereo

Jmol supports two forms of stereo rendering for molecules. In the first form, the two images are placed side by side and rotated so as to appear from slightly different perspectives, creating the illusion of 3D when a practiced user trains one eye on one image and the other eye on the other image.

A second form of stereo rendering, **anaglyphic rendering**, nearly superimposes two identical models that are slightly rotated relative to each other. These models are each of a different color (red and one of blue, cyan, or green). The illusion of depth can then be created when the user wears an inexpensive pair of "3D glasses" that have differently colored lenses.



One should experiment with different background colors when using redcyan or redblue stereo rendering. For many users **background grey** looks better than **background white** or **background black**.

stereo [[stereo-viewing-angle](#)] {default: 5}

Turns side-by-side stereo viewing on. (Note that if this form of stereo viewing is desired, you will probably want to have the applet width twice the applet height.) The default rotation is -5 degrees. Sets the number of degrees of clockwise vertical-axis rotation of the RIGHT-hand image relative to the LEFT-hand image (which itself does not change rotation when stereo viewing is turned on and off). Negative values correspond to cross-eyed viewing, where the left eye is trained on the right image, and the right eye is trained on the left image. Positive values (clockwise rotation) correspond to "wall-eyed" viewing, where the right eye is trained on the right image and the left eye is trained on the left image. Note that **stereo 90** may be useful, as it shows two views of a model that rotate synchronously, a "front view" on the left and a "right side view" on the right.

stereo {default: ON}

Turns side-by-side stereo viewing on with a previously defined rotation or, if no rotation has been defined, a rotation of 5 degrees.

stereo OFF

Turns stereo viewing off.

stereo REDBLUE [[stereo-viewing-angle](#)] {default: 3}

Turns red/blue anaglyphic rendering on with a specific relative rotation, if desired. The default is 3 degrees.

stereo REDCYAN [[stereo-viewing-angle](#)] {default: 3}

Turns red/cyan anaglyphic rendering on with a specific relative rotation, if desired. The default is 3 degrees.

stereo REDGREEN [[stereo-viewing-angle](#)] {default: 3}

Turns red/green anaglyphic rendering on with a specific relative rotation, if desired. The default is 3 degrees.

stereo [[RGB-color](#)] [[RGB-color](#)] [[stereo-viewing-angle](#)] {default: 3}

Turns custom two-color anaglyphic rendering on with a specific relative rotation, if desired. The default is 3 degrees. The second color is optional. If it is left off, then the second color is the complement of the first. So, for example: **stereo red** is the same as **stereo red cyan**. Note that due to the odd way JavaScript designates green (as [x00FF00] rather than [x00FF00]), **stereo red green** is NOT the same as **stereo redgreen**. The equivalent of **stereo redgreen** is **stereo red lime**. When experimenting to match a given set of glasses, it is recommended that you use explicit hex codes for the colors:

**stereo [xFF0000] [x00FF00]**

The set of color names used in Jmol is the JavaScript set, given at <http://jmol.sourceforge.net/jscolors/#JavaScript.colors>.

where

**[stereo-viewing-angle]** is angle of rotation for stereo viewing -- (integer|decimal)

**[RGB-color]** is a name of a color or a red, green, blue color triple in decimal with commas, for example [255,0,255], or as a single hexadecimal number, for example [xFF00FF] (brackets included) -- (color name), [r, g, b], [RRRGGBB]

Examples: [in new window using 1crn.pdb](#)



```
zoom 50;background white;stereo ON
stereo 90
stereo 5
stereo -5
zoom 100;background grey;stereo REDCYAN
stereo REDBLUE 3
stereo REDGREEN;stereo OFF
```

## strand or strands

Strands offer a representation the protein backbone or nucleic acid helix using lines. Curvature control points are as described for [ribbon](#).

strand ON/OFF {default: ON}

strand ONLY

Turns strand rendering on and all other rendering off.

strand [[strand-radius](#)]

Normally, strands vary in width according to the amino acid atom positions. This command sets the radius of the set of strands to be a constant value (a decimal, in Angstroms). Starting with Jmol 12.0, a negative number also implies **ONLY**.

where

**[strand-radius]** is half of the overall width of the set of strands -- (decimal, <=4.0)

Examples:

See [structure.htm](#)



See also:

[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [trace](#) [vector](#) [wireframe](#)

## structure or \_structure

The **structure** command allows customized definition of the secondary structure of a protein model as HELIX, SHEET, TURN, or NONE.

structure HELIX[SHEET|TURN|NONE] (atomExpression)

The atom expression is optional and if missing defaults to the currently selected atoms. **structure HELIX {4-10:B}**, for example, sets residues 4-10 on chain B to be represented as a helix. Note that the file must be of PDB format (from a pdb, mmCIF, or suitable mol2 file) as well.

## struts or strut

(v. 12.0 - new)

In rapid prototyping of protein models, it is sometimes necessary to add short connectors between strands and helices to provide strength to the plastic model. Jmol 12.0 adds a new shape, STRUT, that creates these supports. These stick-like objects can be added using the [CONNECT ... STRUTS](#) or [SET PICKING STRUTS](#) commands, and they can be calculated automatically using [calculate STRUTS](#). STRUTS are not measures and they are not covalent bonds. **set strutDefaultRadius 0.3** sets the default radius for struts. (Their color by default is translucent white.)

struts ON/OFF {default: ON}

Turn struts on or off

struts ONLY

Turns struts on and all other rendering off.

struts [[radius-in-angstroms](#)]

Show struts with the given cylinder diameter in Angstroms

struts [[radius-Rasmol](#)]

Show struts with the given cylinder diameter in Rasmol units (deprecated).

where

**[radius-in-angstroms]** is a (decimal, <=3.0)

**[radius-Rasmol]** is in 1/250ths of an Angstrom (deprecated) -- (integer, 1 to 749)

See also:  
[bondorder](#) [connect](#) [hbonds](#) [set \(bond styles\)](#) [set \(files and scripts\)](#) [ssbonds](#) [wireframe](#) [undefined](#)

[top](#) [search](#) [index](#)

## subset

The **subset** command selects a subset of the atoms that will serve as a basis for all atom expressions until either another subset command is issued or until a new file is loaded. After **subset \*:C**, for example, **select \*** only selects atoms of chain C, **restrict none** only clears chain C, and **restrict \*:C** does nothing. By itself, the **subset** command is the same as **subset all**. Since all menu-driven commands work through the script processor, realize that any user selections will also be affected by the current subset. ([hover](#) is still active, though.) "Select All" from the menu, for example, will only select the current subset of atoms. In addition, atoms outside the current subset cannot be clicked on by the user for atom identification or for measurements. Thus, after operations requiring subset have been completed, consider issuing **subset all** unless you intend to shut off user clicking.

See also:  
[display](#) [hide](#) [restrict](#) [select](#)

[top](#) [search](#) [index](#)

## SWITCH

(v. 12.0 -- new)

**switch** and **case** allow a simplified syntax as an alternative to multiple **if** statements. The implementation in Jmol is more like the JavaScript equivalent than the Java version. Like JavaScript, CASE values can be expression that involve any variable type rather than just integer constants. Expressions are evaluated from the top down; the DEFAULT keyword may appear along with any other CASE or by itself. As for these other languages, an optional BREAK command is necessary after a case to prevent continuation of the script into the next case. For example:

```
switch(prompt("What sort of rendering would you like?","Spacefill|Wireframe|Ball&Stick",true)) {
  case 1:
    spacefill only
    break
  case 2:
    wireframe only
    break;
  case 3:
    wireframe only;wireframe reset;spacefill reset
    break;
}
```

Note:  
 The SWITCH command does not require @ { ... } around Jmol math expressions.

See also:  
[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [break](#) [case](#) [catch](#) [continue](#) [default](#) [echo](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [message](#) [reset](#) [return](#) [set](#) [try](#) [var](#) [while](#)

[top](#) [search](#) [index](#)

## sync

The **sync** command allows synchronization among applets. An applet can be a driver, a slave, or (like many supervisors) both (a slave driver, GET IT?). Three optional settings are available. With **set syncScript ON** scripted commands are transmitted; with **set syncMouse ON** mouse movements are transmitted. These two parameters are independent. If both are off (the default behavior) and sync is ON, then

the orientation of the target applet is maintained the same as the driving applet, but script commands are not automatically transmitted. Details can be seen at [examples-11/sync2.htm](#).

sync .>[\*]appletIdappletId[syncId] ON  
 Turns on synchronization for just this applet (.), all applets except this one (>), all applets (\*), a specific applet on this page (appletId), or a specific applet within a specific synchronization set (appletId[syncId]). All applets selected will drive orientations of all others.  
 sync .>[\*]appletIdappletId[syncId] SLAVE  
 Same as **sync ON**, but the selected applets will be set to receive commands only, not send them.  
 sync .>[\*]appletIdappletId[syncId] OFF  
 Turns off synchronization for the designated applets.  
 sync .>[\*]appletIdappletId[syncId] "command"  
 Sends the command (or scripted sequence of commands) to the designated applets. Quotation marks ARE required.

[top](#) [search](#) [index](#)

## timeout or timeouts

(v. 12.0 - new)

Sets a script to be executed after a given number of milliseconds. See also [show timeouts](#). [Jmol 12.0]

timeout OFF  
 Turns off all pending timeouts.  
 timeout "id" time(ms)  
 If the number of milliseconds given is negative, then the timeout will repeat after this many milliseconds until a new file is loaded or a ZAP command is given.  
 timeout "id" OFF  
 Turns off the named timeout if pending

[top](#) [search](#) [index](#)

## trace

A "trace" is a smooth hermite curve through the same control points used by [ribbons](#).

trace ON/OFF {default: ON}  
 trace ONLY  
 Turns trace rendering on and all other rendering off.  
 trace [\[trace-radius\]](#)

where  
**[trace-radius]** is the radius of the trace -- (decimal, <=4.0)

Examples:

See [structure.htm](#)



See also:  
[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [vector](#) [wireframe](#)

[top](#) [search](#) [index](#)

## translate

Moves the molecule along the specified window-based axis, X, Y, or Z.

translate X or Y (decimal)

Moves the center of rotation to the specified value. A value of 100 will move the molecule completely out of the window. The value represents the percentage of the display window, and 0 is the window center. A value of 50 will move the center of the molecule to the edge of the window. Positive values are to the right of center for X and below center for Y.

translate X or Y (decimal) %

Moves the center of rotation by the specified percent of the window width (X) or height (Y).

translate X or Y (decimal) NM or ANGSTROMS

Moves the center of rotation by the specified number of angstroms or nanometers at the midpoint depth of the model. . Positive values shift the model to the right for X and down for Y.

translate Z (decimal) %

Adjusts the zoom setting such that the specified percent change in field of view is effected. Positive values give the illusion of moving the model toward the user. Values that would magnify the model to the extent that less than one angstrom spans the entire window are ignored.

translate Z (decimal) NM or ANGSTROMS

Adjusts the zoom setting such that the specified number of angstroms or nanometers are removed from the field of view (positive value, magnification) or added to the field of view (negative value, giving the illusion of zooming out) at the midpoint depth of the model, as defined by the dimension from which zoom is referenced (based on **set zoomLarge**). Values that would magnify the model to the extent that less than one angstrom spans the entire window are ignored.

See also:

[animation](#) [frame](#) [invertSelected](#) [model](#) [move](#) [moveto](#) [rotateSelected](#) [set](#) (misc) [spin](#) [translateSelected](#) [zoom](#) [zoomto](#)

[↑top](#) [🔍search](#) [📖index](#)

## translateSelected

Translates a specified subset of atoms. All of the options for [translate](#) are available for **translateSelected**, which then operate only on the currently selected atoms.

translateSelected {x y z} {atomSet}

Moves the indicated atoms along the specified vector, in Angstroms. [Fractional coordinates](#) are allowed and are indicated with a "/" anywhere in any of the three coordinates. If the atom set is not included, the currently selected atom set is used.

Examples:

```
select molecule=1
translateSelected {0 0 1/1}
```

See also:

[animation](#) [frame](#) [invertSelected](#) [model](#) [move](#) [moveto](#) [rotateSelected](#) [set](#) (misc) [spin](#) [translate](#) [zoom](#) [zoomto](#)

[↑top](#) [🔍search](#) [📖index](#)

## try

(v. 12.0 - new)

Surrounding a set of Jmol script with try{...} and following it by catch(e){...} allows the catching of errors within larger blocks of script. The action is the same as in JavaScript. For example, in the following, the first model is never lost, and no error message is printed in the applet window. Any variables created with the VAR keyword are local to the try or catch block.

```
load quartz.cif
refresh
while(true) {
  try {
    load files "caffeine.xyz" "1d.pdb"
  } catch(e) {
    prompt @"oops -- " + e
    break;
  }
  prompt "File loaded successfully"
  break;
}
```

See also:

[break](#) [case](#) [catch](#) [continue](#) [default](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [return](#) [switch](#) [var](#) [while](#)

[↑top](#) [🔍search](#) [📖index](#)

## unbind

(v. 12.0)

Removes the tie between a mouse action and a Jmol action or user script.

unbind

By itself, unbind returns Jmol to its default mouse action binding.

unbind [\[mouse-action\]](#)

Removes all bindings to the specified mouse action.

unbind [\[jmol-action\]](#)

Removes all bindings to the specified jmol action

unbind "script"

Removes all bindings to the specified user script

unbind [\[mouse-action\]](#) [\[jmol-action\]](#)

Removes the specified mouse action for the specified Jmol action.

unbind [\[mouse-action\]](#) "script"

Removes the specified mouse action for the specified script.

where

**[mouse-action]** is any double-quoted combination of a control code (CTRL, ALT, or SHIFT) with a mouse button (LEFT, MIDDLE, RIGHT, or WHEEL) and a click type (SINGLE or DOUBLE)

**[jmol-action]** is one of -- [clickFrank](#), [depth](#), [dragDrawObject](#), [dragDrawPoint](#), [dragLabel](#), [dragSelected](#), [navTranslate](#), [pickAtom](#), [pickSOSurface](#), [pickLabel](#), [pickMeasure](#), [pickNavigate](#), [pickPoint](#), [popupMenu](#), [reset](#), [rotate](#), [rotateSelected](#), [rotateZ](#), [rotateZorZoom](#), [select](#), [selectAndNot](#), [selectNone](#), [selectOr](#), [selectToggle](#), [selectToggleOr](#), [setMeasure](#), [slab](#), [slabAndDepth](#), [slideZoom](#), [spinDrawObjectCCW](#), [spinDrawObjectCW](#), [swipe](#), [translate](#), or [wheelZoom](#)

Examples:

```
unbind "CTRL-SHIFT-LEFT";
unbind "clickFrank";unbind "popupMenu";
unbind "LEFT" "rotate"
```

See also:

[bind](#)

[↑top](#) [🔍search](#) [📖index](#)

## unitcell

(v. 11.0 -- new)

Turns on or off the unit cell for crystal structures, and determines its line style and line width (as a decimal number, in Angstroms).

unitcell ON/OFF{default: ON}

Turns the unit cell on or off

unitcell (decimal)

Sets the unit cell line diameter in Angstroms.

unitcell DOTTED

Sets the axes style to a thin dotted line.

unitcell {i j k}

Sets the origin of the unit cell to be the specified coordinate (with braces). No matter how the coordinate is written (with or without "/"), it is interpreted as a fractional coordinate. For example, **unitcell {0.5 0.5 0.5}** sets the origin of the unit cell to {1/2, 1/2, 1/2}. This change is for display purposes only -- the actual "{0 0 0}" point remains where it was initially.

unitcell TICKS X|Y|Z {major,minor,subminor} FORMAT [%0.2f, ...]  
Sets the parameters for ticks along the a, b, and c edges of the unit cell. An optional specific axis (X, Y, or Z) can be indicated. There are three levels of ticks - major, minor, and "subminor." Only the major ticks have labels. Which of these tick levels are displayed and the distance between ticks depends upon the parameter that takes the form of a point. This point may be in fractional form, {1/2 0 0}. The optional keyword FORMAT allows formatting of the labels for the major ticks. These are based on an array of strings given after the FORMAT keyword. If the array is shorter than the number of ticks, the formats in the array are repeated.

See also:  
[axes](#) [boundingbox](#) [measure](#)

[↑ top](#) [🔍 search](#) [📖 index](#)

## VAR

The **var** keyword is used with variable definitions to localize them to a specific script file or [function](#). Starting with Jmol 12.0, **var** variables within [for](#), [while](#), [try](#), and [catch](#) blocks or any code surrounded by { } except [if](#) and [switch](#) blocks.

Note:  
The VAR command does not require @ { ... } around Jmol math expressions.

See also:  
[break](#) [case](#) [catch](#) [continue](#) [default](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [return](#) [switch](#) [try](#) [while](#)

[↑ top](#) [🔍 search](#) [📖 index](#)

## vector or vectors

Draws vectors arising from vibrational mode data. Operates on whichever atoms have been selected. See also [color \(atom object\)](#)

vector ON/OFF {default: ON}  
Turns vibration vector display on and off  
vector [\[diameter-pixels\]](#)  
Sets the diameter of the vector  
vector [\[radius-in-angstroms\]](#)  
Sets the diameter of the vector  
vector SCALE [\[vector-scale\]](#)  
Adjusts the scale of the vector independently of the vibration motion.

where  
**[diameter-pixels]** is a scaling factor -- (integer, 1 to 19)  
**[radius-in-angstroms]** is a (decimal, <=3.0)  
**[vector-scale]** is a (decimal, -10.0 to 10.0)

See also:  
[backbone](#) [background](#) [cartoon](#) [dots](#) [ellipsoid](#) [geoSurface](#) [meshribbon](#) [ribbon](#) [rocket](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [strand](#) [trace](#) [wireframe](#)

[↑ top](#) [🔍 search](#) [📖 index](#)

## vibration

Provided information is present in the file (xyz format with columns 6-8 indicating dx, dy, and dz, or Gaussian harmonic frequency output), turns on and off display of vibration animation and allows setting of the time period for the vibration (in seconds) and the scale of the motion relative to the default (1).

vibration ON/OFF {default: ON}  
Turns vibration on with a 1-second time period or turns vibration off.  
vibration [\[time-period\]](#)  
Sets the time period for one full vibration in seconds and turns vibration on.  
vibration PERIOD [\[time-period\]](#)  
Sets the time period for one full vibration in seconds, but does not turn vibration on  
vibration SCALE [\[vibration-scale\]](#)  
Sets the scale of the vibration motion independently of the vector length.

where  
**[time-period]** is number of seconds per vibration -- (positive number)  
**[vibration-scale]** is a (decimal, -10.0 to 10.0)

See also:  
[load \[property\]](#)

[↑ top](#) [🔍 search](#) [📖 index](#)

## WHILE or

**while**, like [if](#) evaluates a logical expression, looping until the expression is no longer TRUE:

```
while (x > 0 && x < 10) {
  print x
  x = x - 2
}
```

The looping can be discontinued using either [break](#) or [continue](#). Any variables created with the VAR keyword are local to the **while** block.

See also:  
[\[Jmol command syntax\]](#) [\[Jmol math\]](#) [\[Jmol parameters\]](#) [\[atom expressions\]](#) [\[atom properties\]](#) [\[functions\]](#) [break](#) [case](#) [catch](#) [continue](#) [default](#) [echo](#) [else](#) [elseif](#) [for](#) [goto](#) [if](#) [message](#) [reset](#) [return](#) [set](#) [switch](#) [try](#) [var](#)

[↑ top](#) [🔍 search](#) [📖 index](#)

## wireframe

Wireframe refers to the bonds drawn between the atoms. A boolean value of ON draws the selected bonds as lines. Alternatively, a numeric value may be used to specify the radius of the bonds. A decimal value such as 0.2 or 0.4 specifies Angstroms. The **wireframe** command operates on bonds either BETWEEN ANY TWO atoms in a previously selected atom set (having previously [set bondmode AND](#)) or TO ANY atoms in a previously selected atom set (having previously [set bondmode OR](#)). Note that the selected atoms must already be connected (based on information in the file, Jmol's autobonding algorithm, or from use of the [connect](#) or [bondorder](#) command) in order to show any effect.

wireframe ON/OFF {default: ON}  
Turn wireframe on or off  
wireframe ONLY  
Turns wireframe rendering on and all other rendering off.  
wireframe [\[radius-in-angstroms\]](#)  
Show wireframe with the given cylinder diameter in Angstroms. Starting with Jmol 12.0, a negative number also implies **ONLY**.  
wireframe [\[radius-Rasmol\]](#)  
Show wireframe with the given cylinder diameter in Rasmol units (deprecated).

where  
[radius-in-angstroms] is a (decimal, <=3.0)  
[radius-Rasmol] is in 1/250ths of an Angstrom (deprecated) -- (integer, 1 to 749)

Examples:

wireframe 0.2  
wireframe off

See [bonds.htm](#)

Chime Note:

Rasmol and Chime will take **wireframe 0** as a one-pixel-width bond (equal to **wireframe** and hence to **wireframe on**), while Jmol will interpret it as **wireframe off**.

See also:  
[backbone](#) [background](#) [bondorder](#) [cartoon](#) [connect dots](#) [ellipsoid](#) [geoSurface](#) [hbonds](#) [meshribbon](#) [ribbon](#) [rocket set \(bond styles\)](#) [set \(files and scripts\)](#) [set \(highlights\)](#) [set \(lighting\)](#) [set \(navigation\)](#) [set \(perspective\)](#) [set \(visibility\)](#) [spacefill](#) [ssbonds](#) [strand](#) [trace](#) [vector](#)

[↑top](#) [🔍search](#) [📘index](#)

write  
(v. 12.0 adds LOCALPATH and REMOTEPATH options for write STATE)

(application or signed applet only). Writes information to a file or to the system clipboard. Options include: an image of the application screen, the script history, the current model state in the form of a script, the current molecular orbital in the form of a JVXL file, or the current isosurface in the form of a JVXL file. If the filename is simple (no spaces) and the type is clearly in the extension as, for example, **write test.spt**, then Jmol will deduce the type from the filename and create the proper file. Recognized extensions include ".spt", ".his", ".iso" and ".mo" (both JVXL file format), "mol", "pdb" and "xyz" (coordinates), "jpg", "jpeg", "jpg64", "ppm", and "png". For the clipboard, simply specify CLIPBOARD instead of a file name.

See also:  
[\[using the clipboard\]](#)

[↑top](#) [🔍search](#) [📘index](#)

write (export)

Jmol models can be exported to several formats that can be read by external rendering programs.

write IDTF "fileName"  
Exports the current scene as an [http://u3d.svn.sourceforge.net/viewvc/u3d/trunk/Docs/IntermediateFormat/IDTF%20Format%20Description.pdf~Intermediate File] that can be easily converted to a [http://en.wikipedia.org/wiki/Universal\_3D~Universal 3D] file for incorporation into 3D-PDF documents. The conversion to U3D is done external to Jmol using the (Windows-only) program [misc/ldtf.zip~]. (This is a copy from the Bin/Win32/Release directory found in the zip file downloadable from [http://sourceforge.net/projects/u3d/]; source code for that C++ program is available at this site as well.) Once Jmol has created the IDTF file using, for example, **write IDTF "myfile.idtf"**, simply execute from a Windows command window **IDTFConverter.exe -input myfile.idtf -output myfile.u3d**. The U3D file created will be highly compressed and FAR smaller than that created using other means (such as first creating a VRML file and then converting that to the U3D format). (Jmol 11.8.RC5)  
write MAYA "fileName"  
Exports the current scene as a Maya file. (Basic elements such as atoms and bonds only.)  
write POVRAY "fileName"  
Exports the current scene as POVRAY. The file "fileName" is written along with the additional file, "fileName.ini". (Typically "fileName" would include the ".pov" extension, so these would be of the form xxx.pov and xxx.pov.ini.) Note that the .pov file will embed the Jmol script, so the model can be loaded back into Jmol using [#.script~script "xxx.pov"].  
write VRML "fileName"

Exports the current scene as VRML.  
write X3D "fileName"  
Exports the current scene as X3D.

[↑top](#) [🔍search](#) [📘index](#)

write (image, frames)

Jmol can write a 2D image of the model and, for JPG and PNG format, can append its state so that a single file can be read either as an image or as a Jmol state. In addition, Jmol 11.8 can write sequences of frames as a set of JPG files that can then be incorporated by other programs into movie files.

write FRAMES atom\_expression width height "fileName.jpg"  
Allows making of Jmol movies, creating a set of JPG files filename0001.jpg, filename0002.jpg, etc. for all frames in the atom expression. These files can then be combined using an external program of user's choice to create a movie from a sequence of JPEG images. Width and height are optional, as is the atom expression.

Examples:

write frames {\*} "all.jpg"  
write frames {1.0} "allModelsInFirstFile.jpg"  
write frames {1.0,2.0} "firstTwoFiles.jpg"  
write frames {1.0} 200 200 "smallFile.jpg"

write IMAGE JPG n "fileName"  
Creates a "snapshot" image of the current display and saves it to disk as a JPG image. The optional integer n is a quality from 1 to 100 (default 75). If **set imageState** is TRUE (the default setting), then Jmol inserts into the JPG file its state, allowing a single image file to be used both for viewing as a 2D image and reading as a 3D Jmol state using the [#.script~] command or by dragging and dropping in the Jmol application.  
write IMAGE JPG64 n "fileName"  
Creates a "snapshot" image of the current display and saves it to disk as a JPG or base-64-encoded JPG image. The optional integer n is a quality from 1 to 100 (default 75).  
write IMAGE PNG n "fileName"  
Creates a "snapshot" image of the current display in PNG format and saves it to disk. The amount of compression, n, added in Jmol 11.4, is a number between 0 and 10 (default 2). If **set imageState** is TRUE (the default setting), then Jmol appends to the PNG file its state, allowing a single image file to be used both for viewing as a 2D image and reading as a 3D Jmol state using the [#.script~] command or by dragging and dropping in the Jmol application.  
write IMAGE PPM "fileName"  
Creates a "snapshot" image of the current display in **PPM** format and saves it to disk. The amount of compression, n is a number between 0 and 10 (default 2).

[↑top](#) [🔍search](#) [📘index](#)

write (info)  
(v. 11.0)

The Jmol state, history, menu, currently defined functions, and variables may all be written to files.

write FUNCTIONS "fileName"  
Writes all user-defined functions to a file. (Jmol 11.4)  
write HISTORY "fileName"  
Saves the script command [history](#) to a file.  
write JMOL "fileName"  
Same as ZIPALL (see below).  
write MENU "fileName"  
Saves the current Jmol popup menu as a file. See [load MENU](#).  
write STATE "fileName"  
Saves the current state to a script file.  
write STATE LOCALPATH "path" "fileName"  
Saves the current state to a script file, stripping all local file references to the indicated relative path. (Jmol 12.0)

```
write STATE REMOTEPATH "path" "fileName"
  Saves the current state to a script file, stripping all remote file references to the indicated relative path.
write VAR [variable name] "fileName"
  Saves the value of a variable to a file.
write ZIP "fileName"
  Saves the current state and all necessary LOCAL files to a ZIP file with the given name.
write ZIPALL "fileName"
  Saves the current state and all necessary files -- local or remote -- to a ZIP file with the given name.
```

[↑top](#) [🔍search](#) [iindex](#)

## write (model)

(v. 11.2)

There are several options for writing model data to a file.

```
write COORDS SPT|XYZ|MOL|PDB "fileName"
  Writes molecular data to a file determined by the extension (SPT, XYZ, MOL, or PDB). "coords" and the type may be
  omitted if the filename has a three-letter extension matching one of these types. In the case of XYZ, MOL, and PDB, only
  selected atoms are saved. In the case of SPT, a script file containing the coordinates of the model. This is useful in the case
  that one or more of the atoms have been moved using invertSelected, rotateSelected, or translateSelected. This format is
  also suitable for loading back into Jmol, but only in a simple XYZ-like format, using the load command. However, if the file is
  read using the script command, then atoms should appear in their new positions.
write FILE "fileName"
  Writes the most recent file loaded to the indicated filename. If the model was loaded using a manifest from a ZIP file, then
  the entire ZIP file is written.
write FRAMES atom_expression width height "fileName.jpg"
  Allows making of Jmol movies, creating a set of JPG files filename0001.jpg, filename0002.jpg, etc. for all frames in the atom
  expression. These files can then be combined using an external program of user's choice to create a movie from a sequence
  of JPEG images. Width and height are optional, as is the atom expression.
```

Examples:

```
write frames {*} "all.jpg"
write frames {1.0} "allModelsInFirstFile.jpg"
write frames {1.0,2.0} "firstTwoFiles.jpg"
write frames {1.0} 200 200 "smallFile.jpg"
```

```
write PROPERTIES ⚙️ "fileName"
  Writes a file that is of PDB format and contains the specified parameters in the X, Y, and Z fields, possibly scaled, with the
  scaling indicated in a REMARK record. See the plot PROPERTIES command for parameter options.
write QUATERNION ⚙️ "fileName"
  Writes a file that is of PDB format and contains the quaternion representation of the rotational frame of individual amino acid
  residues in a protein or base pairs in a nucleic acid. See the plot QUATERNION command for parameter options.
write RAMACHANDRAN ⚙️ "fileName"
  Creates a file in PDB format for the alpha carbons of a peptide, where the x, y, and z axis values are actually the PHI, PSI,
  and OMEGA values for each amino acid. An additional optional keyword DRAW writes a script file that would draw PHI and
  PSI angles on the structure. See the plot RAMACHANDRAN command for parameter options.
write MESH "fileName"
  Generates a relatively compact JVXL XML "vertex-only" mesh surface file from an isosurface. A typical command, after
  creation of an isosurface, would be write MESH t.jvxl. Note that standard JVXL files are considerably smaller, however.
write POINTGROUP "fileName"
  Writes a tabular summary of the point group for a symmetrical or nearly symmetrical molecule.
```

See also:  
[draw set \(misc\)](#) [show write \(object\)](#)

[↑top](#) [🔍search](#) [iindex](#)

## write (object)

Several Jmol objects can be saved independently.

```
write ISOSURFACE "fileName"
  Saves the current isosurface in the form of a JVXL file.
write MO "fileName"
  Saves the current molecular orbital in the form of a JVXL file.
write POINTGROUP DRAW "fileName"
  Creates a script file containing DRAW commands that will draw the planes and axes of symmetry for a molecule.
```

See also:  
[draw set \(misc\)](#) [show write \(model\)](#)

[↑top](#) [🔍search](#) [iindex](#)

## zap

Clears the currently loaded molecule. After **zap** or before any model is loaded, all of the following commands are active and may be used to create nonmolecular visualizations: [axes](#), [background](#), [boundbox](#), [center](#), [centerAt](#), [dipole](#), [draw](#), [echo](#), [font](#), [frank](#), [isosurface](#), [move](#), [moveto](#), [pmesh](#), [restore](#), [rotate](#), [save](#), [slab](#), [spin](#), [stereo](#), [translate](#), and related [set](#) commands.

\* indicates only aspects not requiring selection of atoms for these commands. Starting with Jmol 11.6, the ZAP command can be used to delete selected models.

```
zap
  Clears all loaded models.
zap (atom expression)
  Clears all models that include any of the selected atoms. Examples include: zap file=1, zap model=1.1, zap atomno=1, zap not visible
```

See also:  
[define initialize load refresh reset restore save set \(files and scripts\)](#)

[↑top](#) [🔍search](#) [iindex](#)

## zoom

(v. 11.0 -- new capabilities)

Allows enlarging or shrinking of the displayed model. A percentage value specifies the zoom relative to 100, the default value, which in Jmol is calculated so that all atoms are completely visible on the screen through all rotations using the default vanderWaals rendering percentage. The command "zoom OFF" disables mouse-based zooming and zooms to 100. The command "zoom ON" re-enables zooming at the current zoom setting. If the zoom has been turned off, setting the zoom using, for example, "zoom 50," though it sets the "current zoom setting," has no effect until the next "zoom ON" command is given. If an atom is given, then zoom also sets this atom as the rotation center, and if windowCentered is true, that point is moved the center of the screen.

```
zoom ON/OFF {default: ON}
zoom IN
  Zooms IN by a factor of 2 over the course of 1 second.
zoom OUT
  Zooms OUT by a factor of 2 over the course of 1 second.
zoom [percent-zoom]
zoom (atom expression) or {x y z} [percent-zoom]
  Sets a new zoom setting and also designate the specified atom expression or coordinate as the rotation center.
zoom (atom expression) or {x y z} + or - delta
  Adds or subtracts an absolute amount from the current zoom setting.
zoom (atom expression) or {x y z} * or / factor
```

Multiplies or divides the current zoom setting by the indicated factor.

where  
[percent-zoom] is an (integer, 5 to 200000)

See also:  
[animation frame](#) [invertSelected model](#) [move moveto rotateSelected set \(misc\)](#) [spin translate translateSelected zoomto](#)

[↑top](#) [🔍search](#) [📖index](#)

## zoomto

(v. 11.0 -- new)

Carries out a smooth transition to the specified zoom setting. Indicating a new rotation center is optional.

zoomto

By itself, **zoomTo** smoothly zooms IN by a factor of 2 over the course of 1 second.

zoomto IN

Smoothly zooms IN by a factor of 2 over the course of 1 second.

zoomto OUT

Smoothly zooms OUT by a factor of 2 over the course of 1 second.

zoomto [\[time-in-seconds\]](#) (atom expression) or {x y z}

Smoothly moves the specified atom or coordinate to the center of the window if windowCentered or designates it as the center of rotation if not windowCentered. xTrans and yTrans parameters specify percent translation within the window along X and Y, respectively (Jmol 12.0). All parameters are options. The default time is 1 second; indicating no center position results in simple, smooth zooming.

zoomto [\[time-in-seconds\]](#) (atom expression) or {x y z} [\[percent-zoom\]](#) xTrans yTrans

Smoothly transitions to the indicated zoom setting over the course of the specified number of seconds. Optional xTrans and yTrans parameters specify percent translation within the window along X and Y, respectively (Jmol 12.0)

zoomto [\[time-in-seconds\]](#) (atom expression) or {x y z} + or - delta xTrans yTrans

Adds or subtracts an absolute amount from the current zoom setting over the course of the specified number of seconds. Optional xTrans and yTrans parameters specify percent translation within the window along X and Y, respectively (Jmol 12.0)

zoomto [\[time-in-seconds\]](#) (atom expression) or {x y z} \* or / factor xTrans yTrans

Multiplies or divides the current zoom setting by the indicated factor over the course of the specified number of seconds. Optional xTrans and yTrans parameters specify percent translation within the window along X and Y, respectively (Jmol 12.0)

zoomto [\[time-in-seconds\]](#) (atom expression) or {x y z} 0 xTrans yTrans

Zooms to the setting that fills the screen with the designated atoms. (Requires **set perspectiveModel 11** for proper operation.) Can include modifiers +n, -n, \*n, /n after the 0. Optional xTrans and yTrans parameters specify percent translation within the window along X and Y, respectively (Jmol 12.0)

where  
[time-in-seconds] is in (seconds)  
[percent-zoom] is an (integer, 5 to 200000)

See also:  
[animation frame](#) [invertSelected model](#) [move moveto navigate rotateSelected set \(misc\)](#) [set \(navigation\)](#) [spin translate translateSelected zoom](#)

[↑top](#) [🔍search](#) [📖index](#)

## Index

[Jmol SMARTS/SMILES](#)

[Jmol command syntax](#)

[Jmol math](#)

[\[Jmol parameters\]](#)

[\[atom expressions\]](#)

[\[atom properties\]](#)

[\[comment \(#\)\]](#)

[\[export\]](#)

[\[fractional coordinates\]](#)

[\[functions\]](#)

[\[plane expressions\]](#)

[\[read/write ZIP files\]](#)

[\[status reporting\]](#)

[\[using the clipboard\]](#)

[animation or anim](#)  
[animation ON/OFF {default: ON}](#)  
[animation direction -l](#)  
[animation direction +l](#)  
[animation fps \[frames-per-second\]](#)  
[animation frame](#)  
[animation mode LOOP](#)  
[animation mode LOOP \[time-delay1\] \[time-delay2\]](#)  
[animation mode ONCE](#)  
[animation mode PALINDROME](#)  
[animation mode PALINDROME \[time-delay1\] \[time-delay2\]](#)

[axes](#)  
[axes ON/OFF {default: ON}](#)  
[axes \(decimal\)](#)  
[axes CENTER {x y z}](#)  
[axes DOTTED](#)  
[axes \(integer\)](#)  
[axes LABELS "x-label" "y-label" "z-label"](#)  
[axes LABELS "x-label" "y-label" "z-label" "-x-label" "-y-label" "-z-label"](#)  
[axes MOLECULAR](#)  
[axes POSITION \[x y\] or \[x y %\]](#)  
[axes SCALE \(decimal\)](#)  
[axes TICKS XYZ {major,minor,subminor} FORMAT \[%0.2f, ...\] SCALE {scaleX, scaleY, scaleZ} | x.xx](#)  
[axes UNITCELL](#)  
[axes WINDOW](#)

[backbone](#)  
[backbone ON/OFF {default: ON}](#)  
[backbone ONLY](#)  
[backbone \[backbone-radius\]](#)

[background](#)  
[background \[RGB-color\]](#)  
[background ECHO \[color-none-CPK\]](#)  
[background IMAGE "filename"](#)  
[background HOVER \[color-none-CPK\]](#)  
[background LABELS \[color-none-CPK\]](#)

[bind](#)  
[bind \[mouse-action\] \[jmol-action\]](#)  
[bind \[mouse-action\] "script"](#)

[bondorder](#)  
[bondorder 0.5, 1, 1.5, 2, 2.5, 3, 4, -1, -1.5, -2.5](#)  
[bondorder \[connection-options\]](#)

[boundbox or boundingBox](#)

[boundbox \[atom-expression\] {default: \\*} \[line-width-or-type\] {default: ON}](#)  
[boundbox \[atom-expression-or-coordinate\] \[xyz-coordinate\] \[line-width-or-type\] {default: unchanged}](#)  
[boundbox CORNERS \[atom-expression-or-coordinate\] \[atom-expression-or-coordinate\] \[line-width-or-type\] {default: unchanged}](#)  
[boundbox TICKS XYZ {major,minor,subminor} FORMAT \[%0.2f, ...\] SCALE {scaleX, scaleY, scaleZ} | x.xx](#)  
[boundbox SCALE x.xx](#)  
[boundbox \\$isosurfaceID](#)

**[break](#)**

**[calculate](#)**

[calculate AROMATIC](#)  
[calculate HBONDS \[atom-expression\] \[atom-expression\]](#)  
[calculate HYDROGENS \[atom-expression\] {default: \\*}](#)  
[calculate POINTGROUP](#)  
[calculate STRAIGHTNESS](#)  
[calculate STRUCTURE](#)  
[calculate STRUTS](#)  
[calculate SURFACEDISTANCE FROM \[atom-expression\]](#)  
[calculate SURFACEDISTANCE WITHIN \[atom-expression\]](#)

**[cartoon or cartoons](#)**

[cartoon ON/OFF {default: ON}](#)  
[cartoon ONLY](#)  
[cartoon \[cartoon-radius\]](#)

**[CASE](#)**

**[CATCH](#)**

[cd](#)  
[cd](#)  
[cd ""](#)  
[cd "directoryName"](#)  
[cd ?](#)  
[cd =](#)

**[center or centre](#)**

[center \[atom-expression\]](#)  
[center \[xyz-coordinate\]](#)  
[center \[drawn-object\]](#)  
[center](#)

**[centerAt](#)**

[centerAt ABSOLUTE x y z {default: 0.0 0.0 0.0}](#)  
[centerAt AVERAGE x y z {default: 0.0 0.0 0.0}](#)  
[centerAt BOUNDBOX x y z {default: 0.0 0.0 0.0}](#)

**[color or colour](#)**

[color \[color-scheme\]](#)  
[color \[color-scheme\] TRANSLUCENT](#)

**[color \(atom object\)](#)**

[color \[atom-associated-object\] \[color-scheme\]](#)

**[color \(bond object\)](#)**

[color BONDS \[color-none-CPK\]](#)  
[color SSBONDS \[color-none-CPK\]](#)  
[color HBONDS \[color-none-CPK\]](#)  
[color HBONDS ENERGY](#)  
[color HBONDS TYPE](#)

**[color \(element\)](#)**

[color \[element-name\] \[RGB-color\]](#)

**[color \(model object\)](#)**

[color \[model-object\] \[RGB-color\]](#)

**[color \(named object\)](#)**

[color \[drawn-object\] \[RGB-color\]](#)

**[color \(other\)](#)**

[color HIGHLIGHT \[RGB-color\]](#)  
[color SELECTIONHALOS \[color-none-CPK\]](#)

**[color \(scheme\)](#)**

[color "colorSchemeName" RANGE \[min\] \[max\]](#)

**[color measures](#)**

[color measures \[RGB-color\]](#)

**[compare](#)**

[compare {model1} {model2} SUBSET {atomSet} ATOMS \[paired atom list\]](#)  
[compare {model1} {model2} ORIENTATIONS \[paired atom list\]](#)  
[compare {model1} {model2} ORIENTATIONS \[paired quaternion array list\]](#)  
[compare {model1} {model2} SMARTS or SMILES "smartsString"](#)

**[configuration or conformation or config](#)**

[configuration \[configuration number\]](#)

**[connect](#)**

**[console](#)**

**[continue](#)**

**[data](#)**

[data "label"](#)  
[data "label @x"](#)  
[data "data2d\\_xxx"](#)  
[data "property\\_xxx.propertyAtomField.propertyDataField"](#)  
[data "property\\_xxx.propertyAtomField.propertyAtomColumnCount.propertyDataField.propertyDataColumnCount"](#)  
[data CLEAR](#)  
[data element\\_vdw 6 1.7; 7 1.8 END element\\_vdw](#)

**[default](#)**

**[define or @@](#)**

[define \[variable-name\] \[atom-expression\]](#)

**[delay](#)**

[delay \[time-delay\]](#)  
[delay on](#)

**[delete](#)**

[delete](#)

**[depth](#)**

**[dipole or dipoles](#)**

**[display](#)**

[display \[atom-expression\]](#)

**[dots](#)**

[dots ON/OFF {default: ON}](#)  
[dots ONLY](#)  
[dots VANDERWAALS](#)  
[dots IONIC](#)  
[dots m%](#)  
[dots \(decimal\)](#)  
[dots +\(decimal\)](#)  
[dots ADPMIN n%](#)  
[dots ADPMAX n%](#)

**[draw](#)**

[draw BOUNDBOX](#)  
[draw DELETE](#)  
[draw FRAME {atom expression} {quaternion}](#)  
[draw HELIX AXIS](#)  
[draw INTERSECTION boundBox \(plane expression\)](#)

[draw INTERSECTION UnitCell \(plane expression\)](#)  
[draw LIST](#)  
[draw POINTGROUP \[parameters\]](#)  
[draw POLYGON \[polygon definition\]](#)  
[draw QUATERNION \[parameters\]](#)  
[draw RAMACHANDRAN](#)  
[draw SYMOP {atom expression} {atom expression}](#)  
[draw SYMOP \(integer\) {atom expression} {atom expression}](#)  
[draw SYMOP \[matrix\]](#)  
[draw UNITCELL](#)  
[draw SYMOP \[n or "x,y,z"\] {atom expression}](#)

[echo](#)  
[echo \(string\)](#)

**[ellipsoid or ellipsoids](#)**  
[ellipsoid ON/OFF {default: ON}](#)  
[ellipsoid nr%](#)  
[ellipsoid ID \[object id\] ON](#)  
[ellipsoid ID \[object id\] OFF](#)  
[ellipsoid ID \[object id\] AXES {ax av az} {bx by bz} {cx cy cz}](#)  
[ellipsoid ID \[object id\] CENTER {x y z}](#)  
[ellipsoid ID \[object id\] CENTER { atom expression }](#)  
[ellipsoid ID \[object id\] CENTER Subject](#)  
[ellipsoid ID \[object id\] COLOR \[color parameters\]](#)  
[ellipsoid ID \[object id\] DELETE](#)  
[ellipsoid ID \[object id\] SCALE \(decimal\)](#)

[else](#)

[elseif](#)

[exit](#)

[fix](#)  
[fix \[atom-expression\]](#)

**[font](#)**  
[font \[object-with-text\] \[font-size\] \[font-face\] {default: SansSerif} \[font-style\] {default: Plain} \[scaling factor\]](#)

**[FOR](#)**

**[frame or frames](#)**  
[frame \(integer >= 1\)](#)  
[frame \(decimal\)](#)  
[frame \(decimal\) - \(decimal\)](#)  
[frame 0](#)  
[frame 0.0](#)  
[frame ALIGN { atom expression }](#)  
[frame ALL](#)  
[frame LAST](#)  
[frame NEXT](#)  
[frame PAUSE](#)  
[frame PLAY \(starting frame\)](#)  
[frame PLAYREV \(starting frame\)](#)  
[frame PREVIOUS](#)  
[frame RANGE \(starting frame\) \(ending frame\)](#)  
[frame RESUME](#)  
[frame REWIND](#)  
[frame TITLE "title"](#)

**[frank](#)**  
[frank ON/OFF](#)

**[geoSurface](#)**  
[geoSurface ON/OFF {default: ON}](#)  
[geoSurface ONLY](#)  
[geoSurface VANDERWAALS](#)  
[geoSurface IONIC](#)  
[geoSurface \(integer\)](#)  
[geoSurface \(decimal\)](#)

[geoSurface +\(decimal\)](#)

**[getProperty](#)**  
[getProperty animationInfo](#)  
[getProperty appletInfo](#)  
[getProperty atomInfo \(atom expression\)](#)  
[getProperty atomList \(atom expression\)](#)  
[getProperty auxiliaryInfo](#)  
[getProperty bondInfo \(atom expression\)](#)  
[getProperty boundBoxInfo](#)  
[getProperty centerInfo](#)  
[getProperty chainInfo \(atom expression\)](#)  
[getProperty dataInfo type](#)  
[getProperty extractModel \(atom expression\)](#)  
[getProperty fileContents](#)  
[getProperty fileContents filepath](#)  
[getProperty fileHeader](#)  
[getProperty fileName](#)  
[getProperty image](#)  
[getProperty jmolStatus statusNameList](#)  
[getProperty jmolViewer](#)  
[getProperty measurementInfo](#)  
[getProperty messageQueue](#)  
[getProperty menu](#)  
[getProperty minimizationInfo](#)  
[getProperty modelInfo](#)  
[getProperty moleculeInfo \(atom expression\)](#)  
[getProperty orientationInfo](#)  
[getProperty polymerInfo \(atom expression\)](#)  
[getProperty shapeInfo](#)  
[getProperty stateInfo \(atom expression\)](#)  
[getProperty transformInfo](#)

**[goto](#)**

**[halos](#)**  
[halos ONLY](#)  
[halos ON/OFF {default: ON}](#)  
[halos reset](#)

**[hbonds](#)**  
[hbonds ON/OFF {default: ON}](#)  
[hbonds \[width-in-angstroms\]](#)  
[hbonds CALCULATE](#)

**[help](#)**  
[help query](#)

**[hide](#)**  
[hide \[atom-expression\]](#)

**[history](#)**  
[history ON/OFF {default: ON}](#)

**[hover](#)**

**[IF](#)**

**[initialize](#)**

**[invertSelected](#)**  
[invertSelected POINT point\\_definition](#)  
[invertSelected PLANE plane express](#)  
[invertSelected HKL {h k l}](#)  
[invertSelected STEREO {center} {atomsToInvert}](#)

**[isosurface](#)**  
[isosurface AREA](#)  
[isosurface VOLUME](#)  
[isosurface DELETE](#)  
[isosurface LATTICE {a b c}](#)

[isosurface LIST](#)

**[label or labels](#)**

[label ON/OFF {default: ON}](#)  
[label TOGGLE {atom expression}](#)

**[lcaoCartoon](#)**

[lcaoCartoon ON/OFF {default: ON}](#)  
[lcaoCartoon CREATE {type}](#)  
[lcaoCartoon CREATE {type} MOLECULAR](#)  
[lcaoCartoon COLOR \[RGB-color\]](#)  
[lcaoCartoon COLOR \[RGB-color\] \[RGB-color\]](#)  
[lcaoCartoon DELETE](#)  
[lcaoCartoon LIST](#)  
[lcaoCartoon SCALE {decimal}](#)  
[lcaoCartoon SELECT {atom expression}](#)  
[lcaoCartoon SELECT {type}](#)  
[lcaoCartoon TRANSLUCENT or OPAQUE](#)

**[load](#)**

[load](#)  
[load "filename" {integer}](#)  
[load "filetype::filename"](#)  
[load @variableName](#)  
[load "@variableName"](#)  
[load =XXXX](#)  
[load SXXXX](#)  
[load SMILES "smilesString"](#)  
[load keyword "filename"](#)  
[load "filename" FILTER "filter specification"](#)  
[load "remoteFileName" AS "localFileName"](#)

**[load APPEND](#)**

**[load DATA](#)**

**[load FILES](#)**

[load FILES "filename1" "filename2" #](#)

**[load MENU](#)**

[load MENU "menufile"](#)

**[load MODELS](#)**

[load MODELS {first last stride} "filename"](#)  
[load MODELS {\(i j k:l m...\)} "filename"](#)

**[load TRAJECTORY](#)**

[load TRAJECTORY "filename"](#)  
[load TRAJECTORY {first last stride} or {\(i j k:l m...\)} "filename"](#)  
[load TRAJECTORY "filename" FILTER "filter specification" COORD {first last stride} or {\(i j k:l m...\)} mdcrd::crdfile](#)

**[load \[property\]](#)**

**[LOG](#)**

**[loop](#)**

[loop \[time-delay\]](#)  
[loop on](#)

**[mapProperty](#)**

[mapProperty {atomExpression1} .property1 {atomExpression2} .property2 .propertyKey](#)  
[mapProperty SELECTED {atomExpression} .propertyKey](#)

**[measure or measures or monitor or monitors](#)**

[measure ON/OFF {default: ON}](#)  
[measure "n:labelFormat"](#)  
[measure {two to four atom expressions, each in parentheses} "labelFormat"](#)  
[measure {integer} {integer} "labelFormat"](#)  
[measure {integer} {integer} {integer} "labelFormat"](#)  
[measure {integer} {integer} {integer} {integer} "labelFormat"](#)

[measure TICKS X|Y|Z {major,minor,subminor} FORMAT \[%0.2f ...\] SCALE {scaleX, scaleY, scaleZ} \[ x.xx FIRST x.xx {point1} {point2}](#)  
[measure ALL {two to four atom expressions} "labelFormat"](#)  
[measure ALLCONNECTED {two to four atom expressions} "labelFormat"](#)  
[measure DELETE](#)  
[measure DELETE {integer}](#)  
[measure DELETE {two to four atom expressions}](#)  
[measure RANGE {decimal} {decimal} ALL|ALLCONNECTED|DELETE {two to four atom expressions, each in parentheses}](#)

**[meshribbon or meshribbons](#)**

[meshribbon ON/OFF {default: ON}](#)  
[meshribbon ONLY](#)  
[meshribbon \[mesh-ribbon-radius\]](#)

**[message](#)**

[message {string}](#)

**[minimize or minimization](#)**

[minimize](#)  
[minimize ADDHYDROGENS](#)  
[minimize CANCEL](#)  
[minimize CLEAR](#)  
[minimize CONSTRAINT CLEAR](#)  
[minimize CONSTRAINT {two to four atom expressions} {decimal}](#)  
[minimize CRITERION](#)  
[minimize ENERGY](#)  
[minimize SELECT {atom-expression}](#)  
[minimize STEPS {integer}](#)  
[minimize STOP](#)  
[minimize FIX {atom-expression}](#)

**[mo](#)**

[mo ON/OFF {default: ON}](#)  
[mo {integer}](#)  
[mo COLOR \[RGB-color\]](#)  
[mo COLOR \[RGB-color\] \[RGB-color\]](#)  
[mo CUTOFF {decimal}](#)  
[mo DELETE](#)  
[mo HOMO \[+/-n\]](#)  
[mo LUMO \[+/-n\]](#)  
[mo MODEL n or x.y](#)  
[mo NEXT](#)  
[mo NOPLANE](#)  
[mo PLANE plane\\_expression](#)  
[mo PREVIOUS \[RGB-color\]](#)  
[mo RESOLUTION {decimal}](#)  
[mo TITLEFORMAT "format"](#)

**[model or models](#)**

**[move](#)**

[move \[x-rotation\] \[y-rotation\] \[z-rotation\] \[zoom-factor\] \[x-translation\] \[y-translation\] \[z-translation\] \[slab-cutoff\] \[seconds-total\] \[move-frames-per-second\] {default: 30} \[maximum-acceleration\] {default: 5}](#)

**[moveto](#)**

[moveto timeSeconds FRONT|BACK|LEFT|RIGHT|TOP|BOTTOM](#)  
[moveto timeSeconds {x y z} .degrees zoomPercent transX transY {x y z} .rotationRadius navigationCenter .navTransX .navTransY .navDepth](#)  
[moveto timeSeconds {x y z} .degrees 0 transX transY {atom expression} 0 zoomAdjustment navigationCenter .navTransX .navTransY .navDepth](#)  
[moveto timeSeconds {x y z} .degrees {atom expression} 0 zoomAdjustment navigationCenter .navTransX .navTransY .navDepth](#)  
[moveto STOP](#)

**[navigate or navigation](#)**

[navigate timeSeconds CENTER {x y z}](#)  
[navigate timeSeconds CENTER { atom expression }](#)  
[navigate timeSeconds CENTER Subject](#)  
[navigate timeSeconds DEPTH percent](#)  
[navigate timeSeconds PATH Subject](#)  
[navigate timeSeconds PATH {any combination of coordinates, atom expressions, and objects}](#)  
[navigate timeSeconds QUATERNION { quaternion }](#)  
[navigate timeSeconds QUATERNION MOLECULAR { quaternion }](#)  
[navigate timeSeconds QUATERNION { atom expression }](#)  
[navigate timeSeconds ROTATE X degrees](#)

[navigate timeSeconds ROTATE Y degrees](#)  
[navigate timeSeconds ROTATE Z degrees](#)  
[navigate timeSeconds TRACE \(atom expression\)](#)  
[navigate timeSeconds TRANSLATE x.xx y.yy](#)  
[navigate timeSeconds TRANSLATE X x.xx Y y.yy](#)  
[navigate timeSeconds TRANSLATE {x y z}](#)  
[navigate timeSeconds TRANSLATE \(atom expression\)](#)  
[navigate timeSeconds TRANSLATE \\$object](#)

[parallel/process](#)

[pause or wait](#)  
[pause message](#)

**plot**  
[plot PROPERTIES property1 property2](#)  
[plot PROPERTIES property1 property2 property3](#)  
[plot QUATERNION w, x, y, or z](#)  
[plot QUATERNION a,r DIFFERENCE](#)  
[plot QUATERNION a,r DIFFERENCE2](#)  
[plot RAMACHANDRAN](#)  
[plot RAMACHANDRAN r](#)

**pmesh**  
[pmesh ID \[object ID\]](#)  
[pmesh ID \[object ID\] ON/OFF {default: ON}](#)  
[pmesh ID \[object ID\] DELETE](#)  
[pmesh ID \[object ID\] "filename"](#)  
[pmesh ID \[object ID\] DOTS or NODOTS {default: NODOTS} "xyz.pmesh.gz" {default: current}](#)  
[pmesh ID \[object ID\] FILL or NOFILL {default: FILL} "xyz.pmesh.gz" {default: current}](#)  
[pmesh LIST](#)  
[pmesh ID \[object ID\] MESH or NOMESH {default: NOMESH} "xyz.pmesh.gz" {default: current}](#)

[polyhedra](#)

**PRINT**

**PROMPT**  
[prompt](#)  
[prompt "message"](#)

[quaternion or quaternions](#)

[quit](#)

[ramachandran or rama](#)

[refresh](#)

**reset**  
[reset](#)  
[reset AROMATIC](#)  
[reset FUNCTIONS](#)  
[reset variableName](#)  
[reset ALL](#)

**restore**  
[restore BONDS saveName](#)  
[restore ORIENTATION saveName timeSeconds](#)  
[restore SELECTION saveName](#)  
[restore STATE saveName](#)

**restrict**  
[restrict {default: ALL}](#)  
[restrict \[atom-expression\]](#)  
[restrict BONDS \[atom-expression\]](#)

**resume**  
[resume](#)

**RETURN**  
[return returnValue](#)

**ribbon or ribbons**  
[ribbon ON/OFF {default: ON}](#)  
[ribbon ONLY](#)  
[ribbon \[ribbon-radius\]](#)

**rocket or rockets**  
[rocket ON/OFF {default: ON}](#)  
[rocket ONLY](#)  
[rocket \[rocket-radius\]](#)

[rotate](#)

[rotateSelected](#)

**save**  
[save BONDS saveName](#)  
[save ORIENTATION saveName](#)  
[save SELECTION saveName](#)  
[save STATE saveName](#)

**script or source**  
[script \[file-name\]](#)  
[script LOCALPATH "path" \[file-name\]](#)  
[script REMOTEPATH "path" \[file-name\]](#)  
[script \[file-name\] CHECK](#)  
[script \[file-name\] COMMAND n](#)  
[script \[file-name\] COMMANDS n - m](#)  
[script \[file-name\] LINE n](#)  
[script \[file-name\] LINES n - m](#)  
[script APPLELET appleName @\({Jmol math expression}\)](#)  
[script INLINE @\({Jmol math expression}\)](#)  
[script javascript.functionCall\(\)](#)

**select**  
[select {default: ALL}](#)  
[select \[atom-expression\]](#)  
[select \[atom-expression\] \(property expression\)](#)

**selectionHalos**  
[selectionHalos ON/OFF {default: ON}](#)

**set**  
[set](#)  
[set xxx?](#)

**set (antialiasing)**  
[set antialiasDisplay OFF](#)  
[set antialiasTranslucent ON](#)  
[set antialiasImages ON](#)

**set (bond styles)**  
[set bondMode AND](#)  
[set bondMode OR](#)  
[set bondModeOr FALSE](#)  
[set bondRadiusMilliAngstroms \(integer\)](#)  
[set bondTolerance \(decimal\)](#)  
[set dipoleScale \(-10.0 to 10.0\)](#)  
[set hbondsRasmol TRUE](#)  
[set hbondsSolid FALSE](#)  
[set hbondsBackbone FALSE](#)  
[set minBondDistance \(decimal\)](#)  
[set showMultipleBonds ON](#)  
[set ssbonds BACKBONE or SIDECHAIN](#)  
[set ssBondsBackbone FALSE](#)

**set (callback)**  
[set AnimFrameCallback "function name"](#)

[set EchoCallback "function name"](#)  
[set EvalCallback](#)  
[set HoverCallback "function name"](#)  
[set LoadStructCallback "function name"](#)  
[set MeasureCallback "function name"](#)  
[set MessageCallback "function name"](#)  
[set MinimizationCallback "function name"](#)  
[set PickCallback "function name"](#)  
[set ResizeCallback "function name"](#)  
[set ScriptCallback "function name"](#)  
[set SyncCallback "function name"](#)

**[set \(debugging\)](#)**

[set debug OFF](#)  
[set debugScript OFF](#)  
[set delayMaximumMs 0](#)  
[set fontCaching TRUE](#)  
[set historyLevel \(integer\)](#)  
[set logLevel \(0 - 5\)](#)  
[set scriptReportingLevel \(integer\)](#)  
[set showScript OFF](#)  
[set showScript milliseconds](#)

**[set \(files and scripts\)](#)**

[set allowEmbeddedScripts](#)  
[set appendNew TRUE](#)  
[set appletProxy "URL"](#)  
[set applySymmetryToBonds OFF](#)  
[set atomTypes "..."](#)  
[set autobond ON](#)  
[set autoLoadOrientation FALSE](#)  
[set currentLocalPath "path"](#)  
[set dataSeparator "separator text"](#)  
[set defaultDirectory "directory path"](#)  
[set defaultLattice { i j k }](#)  
[set defaultLoadScript "script"](#)  
[set edsUrlCutoff "url"](#)  
[set edsUrlFormat "url"](#)  
[set forceAutoBond OFF](#)  
[set history nLines](#)  
[set loadFormat "URL"](#)  
[set scriptQueue ON](#)  
[set smallMoleculeMaxAtoms 40000](#)

**[set \(highlights\)](#)**

[set display SELECTED/NORMAL](#)  
[set frank](#)

**[set \(labels\)](#)**

[set fontScaling OFF](#)  
[set fontSize \[font-size\] {default: 8}](#)  
[set labelAlignment LEFT, RIGHT, or CENTER](#)  
[set labelAtom ON/OFF {default: ON} { atom expression }](#)  
[set labelFront ON/OFF {default: ON} { atom expression }](#)  
[set labelGroup ON/OFF {default: ON} { atom expression }](#)  
[set labelOffset \[x-offset\] \[y-offset\] { atom expression }](#)  
[set labelPointer OFF { atom expression }](#)  
[set labelPointer BACKGROUND { atom expression }](#)  
[set labelToggle { atom expression }](#)

**[set \(language\)](#)**

[set language "two-letter code"](#)  
[set languageTranslation ON](#)

**[set \(lighting\)](#)**

[set ambientPercent \(integer 0 to 100\)](#)  
[set diffusePercent \(integer 0 to 100\)](#)  
[set phongExponent \(integer 0 to 1000\)](#)  
[set specular OFF](#)  
[set specularExponent \(integer 1 to 10\)](#)  
[set specularPercent \(integer 0 to 100\)](#)

[set specularPower \(integer 0 to 100\)](#)  
[set zShadePower \(integer\)](#)

**[set \(measure\)](#)**

[set defaultDistanceLabel "format"](#)  
[set defaultAngleLabel "format"](#)  
[set defaultTorsionLabel "format"](#)  
[set dynamicMeasurements ON](#)  
[set measurements \[width-in-angstroms\]](#)  
[set measurements \[linewidth-pixels\]](#)  
[set justifyMeasurements FALSE](#)  
[set measurements DOTTED](#)  
[set measurementLabels ON](#)  
[set measurementUnits \[distance-unit\]](#)  
[set showMeasurements TRUE](#)

**[set \(misc\)](#)**

[set allowGestures FALSE](#)  
[set allowKeystrokes FALSE](#)  
[set allowModelKit TRUE](#)  
[set allowMultiTouch TRUE](#)  
[set allowRotateSelected FALSE](#)  
[set animationFps \(integer\)](#)  
[set autoFPS FALSE](#)  
[set axesColor "color\\_name"](#)  
[set axis1Color "color\\_name"](#)  
[set axis2Color "color\\_name"](#)  
[set axis3Color "color\\_name"](#)  
[set atomPicking TRUE](#)  
[set backgroundModel \(integer >= 1\) or "file model"](#)  
[set bondPicking FALSE](#)  
[set chainCaseSensitive FALSE](#)  
[set colorRasmol FALSE](#)  
[set defaultColorScheme JMOL or RASMOL](#)  
[set defaultDrawArrowScale \(decimal\)](#)  
[set defaults JMOL or RASMOL](#)  
[set defaultVDW JMOL or BABEL or RASMOL or USER](#)  
[set dotDensity -3 to 3](#)  
[set dotScale \(integer\)](#)  
[set dotsSelectedOnly FALSE](#)  
[set dotSurface ON](#)  
[set dragSelected OFF](#)  
[set drawHover OFF](#)  
[set drawPicking OFF](#)  
[set exportDrivers "driver\\_list"](#)  
[set formalCharge \(integer\)](#)  
[set fractionalRelative FALSE](#)  
[set helixStep \(integer\)](#)  
[set helpPath "URL"](#)  
[set hoverDelay \(decimal\)](#)  
[set hoverLabel \(string\)](#)  
[set imageState ON](#)  
[set isKiosk OFF](#)  
[set isosurfacePropertySmoothing ON](#)  
[set loadAtomDataTolerance \(decimal\)](#)  
[set measureAllModels OFF](#)  
[set messageStyleChime FALSE](#)  
[set minimizationCriterion \(decimal\)](#)  
[set minimizationRefresh TRUE](#)  
[set minimizationSilent FALSE](#)  
[set minimizationSteps \(integer\)](#)  
[set mouseDragFactor \(decimal\)](#)  
[set mouseWheelFactor \(decimal\)](#)  
[set multiprocessor FALSE](#)  
[set pdbGetHeader FALSE](#)  
[set pdbSequential FALSE](#)  
[set percentVdw Atom \(integer\)](#)  
[set pickingSpinRate \(integer\)](#)  
[set pointGroupDistanceTolerance \(decimal\)](#)  
[set pointGroupLinearTolerance \(decimal\)](#)  
[set pickLabel \(string\)](#)  
[set preserveState TRUE](#)

[set propertyAtomNumberColumnCount \(integer\)](#)  
[set propertyAtomNumberField \(integer\)](#)  
[set propertyColorScheme "colorSchemeName"](#)  
[set propertyDataColumnCount \(integer\)](#)  
[set propertyDataField \(integer\)](#)  
[set quaternionFrame A,B,C,N,P,Q,R,C,RP,X](#)  
[set rangeSelected](#)  
[set repaintWaitMs 1000](#)  
[set saveProteinStructureState TRUE](#)  
[set selectAllModels TRUE](#)  
[set selectHetero ON](#)  
[set selectHydrogen ON](#)  
[set showKeyStrokes TRUE](#)  
[set smartAromatic ON](#)  
[set spinFps \[frames-per-second\]](#)  
[set spinX \[degrees-per-second\]](#)  
[set spinY \[degrees-per-second\]](#)  
[set spinZ \[degrees-per-second\]](#)  
[set stateVersion \(integer\)](#)  
[set statusReporting ON](#)  
[set stereoDegrees \(decimal\)](#)  
[set strutDefaultRadius 0.3](#)  
[set strutLengthMaximum 7](#)  
[set strutsMultiple OFF](#)  
[set strutSpacing 6](#)  
[set syncMouse OFF](#)  
[set syncScript OFF](#)  
[set useMinimizationThread ON](#)  
[set useNumberLocalization ON](#)  
[set vectorScale \(decimal\)](#)  
[set vibrationPeriod \(decimal\)](#)  
[set vibrationScale \(decimal\)](#)  
[set waitForMoveto ON](#)  
[set wireframeRotation OFF](#)

**[set \(navigation\)](#)**

[set hideNavigationPoint FALSE](#)  
[set navFPS \(integer\)](#)  
[set navigateSurface FALSE](#)  
[set navigationDepth \(percent\)](#)  
[set navigationMode FALSE](#)  
[set navigationPeriodic FALSE](#)  
[set navigationSpeed \(integer\)](#)  
[set navigationSlab \(percent\)](#)  
[set navX \(decimal\)](#)  
[set navY \(decimal\)](#)  
[set navZ \(decimal\)](#)  
[set showNavigationPointAlways FALSE](#)  
[set visualRange \(angstroms\)](#)

**[set \(perspective\)](#)**

[set cameraDepth \(positive number\)](#)  
[set perspectiveDepth ON](#)  
[set perspectiveModel 11](#)  
[set scaleAngstromsPerInch \[viewing-distance\]](#)  
[set rotationRadius \(Angstroms\)](#)  
[set windowCentered ON](#)  
[set zoomEnabled ON](#)  
[set zoomLarge ON](#)  
[set zShade OFF](#)

**[set \(structure\)](#)**

[set cartoonBaseEdges FALSE](#)  
[set cartoonRockets OFF](#)  
[set hermiteLevel \(integer, -8 to 8\)](#)  
[set highResolution OFF](#)  
[set ribbonAspectRatio \(integer\)](#)  
[set ribbonBorder OFF](#)  
[set rocketBarrels OFF](#)  
[set sheetSmoothing \(0 to 1\)](#)  
[set strandCount \[strand-count\]](#)

[set strandCountForMeshRibbon \[strand-count\]](#)  
[set strandCountForStrands \[strand-count\]](#)  
[set traceAlpha TRUE](#)

**[set \(visibility\)](#)**

[set axes \[line-width-or-type\]](#)  
[set axesMode 0, 1, or 2](#)  
[set axesMolecular OFF](#)  
[set axesScale \(decimal\)](#)  
[set axesUnitcell OFF](#)  
[set axesWindow ON](#)  
[set backgroundColor \[RGB-color\]](#)  
[set boundingbox \[line-width-or-type\]](#)  
[set boundingboxColor "color\\_name"](#)  
[set defaultTranslucent \(decimal\)](#)  
[set disablePopupMenu FALSE](#)  
[set displayCellParameters TRUE](#)  
[set greyScaleRendering OFF](#)  
[set hideNameInPopUp FALSE](#)  
[set hideNotSelected FALSE](#)  
[set refreshing TRUE](#)  
[set showAxes FALSE](#)  
[set showBoundingBox FALSE](#)  
[set showFrank TRUE](#)  
[set showHiddenSelectionHalos FALSE](#)  
[set showHydrogens TRUE](#)  
[set showSelections FALSE](#)  
[set showUnitcell FALSE](#)  
[set slabByAtom FALSE](#)  
[set slabByMolecule FALSE](#)  
[set slabEnabled FALSE](#)  
[set solventProbe OFF](#)  
[set solventProbeRadius \[probe-radius-in-angstroms\] {default: 1.2}](#)  
[set unitcell](#)  
[set unitCellColor "color\\_name"](#)

**[set echo](#)**

[set echo user-named \[horizontal-position\] {default: left}](#)  
[set echo user-named \[x y\]](#)  
[set echo user-named \[x y %\]](#)  
[set echo user-named \[x y z\]](#)  
[set echo user-named \[atom-expression\] }](#)  
[set echo user-named DEPTH.%z](#)  
[set echo name DISPLAYED](#)  
[set echo user-named IMAGE "file name"](#)  
[set echo user-named MODEL \(model number\)](#)  
[set echo user-named SCRIPT "script"](#)  
[set echo name HIDDEN](#)  
[set echo ALL](#)  
[set echo DISPLAYED](#)  
[set echo HIDDEN](#)  
[set echo NONE](#)  
[set echo OFF](#)

**[set modelKitMode](#)**

**[set picking](#)**

[set picking ON](#)  
[set picking CENTER](#)  
[set picking CONNECT](#)  
[set picking DELETEATOM](#)  
[set picking DELETEBOND](#)  
[set picking DRAGATOM](#)  
[set picking DRAGMINIMIZE](#)  
[set picking DRAGMINIMIZEMOLECULE](#)  
[set picking DRAW](#)  
[set picking IDENT](#)  
[set picking INVERTSTEREO](#)  
[set picking LABEL](#)  
[set picking MEASURE](#)  
[set picking MEASURE.DISTANCE](#)

[set picking MEASURE ANGLE](#)  
[set picking MEASURE TORSION](#)  
[set picking MEASURE SEQUENCE](#)  
[set picking NAVIGATION](#)  
[set picking SELECT.ATOM](#)  
[set picking SELECT.CHAIN](#)  
[set picking SELECT.GROUP](#)  
[set picking SELECT.ELEMENT](#)  
[set picking SELECT.MOLECULE](#)  
[set picking SELECT.POLYMER](#)  
[set picking SELECT.SITE](#)  
[set picking SELECT.STRUCTURE](#)  
[set picking SPIN \[frames-per-second\]](#)  
[set picking STRUTS](#)  
[set picking SYMMETRY](#)

**[set pickingStyle](#)**  
[set pickingStyle SELECT toggle](#)  
[set pickingStyle SELECT selectOrToggle](#)  
[set pickingStyle SELECT extendedSelect](#)  
[set pickingStyle SELECT DRAG](#)  
[set pickingStyle SELECT NONE](#)  
[set pickingStyle MEASURE.ON](#)

**[set userColorScheme](#)**  
[set userColorScheme: colorName colorName #](#)

**[show](#)**  
[show.ATOM](#)  
[show.BOUNDBOX](#)  
[show.CENTER](#)  
[show.CHAIN](#)  
[show.COLORSCHEME "name"](#)  
[show.DATA "type"](#)  
[show.DRAW](#)  
[show.FILE](#)  
[show.FILE filepath](#)  
[show.ISOSURFACE](#)  
[show.FUNCTIONS](#)  
[show.GROUP](#)  
[show.HISTORY n](#)  
[show.MEASUREMENTS](#)  
[show.MENU](#)  
[show.MO](#)  
[show.MODEL](#)  
[show.MOVETO](#)  
[show.ORIENTATION \[optional type\]](#)  
[show.PDBHEADER](#)  
[show.POINTGROUP](#)  
[show.RESIDUES](#)  
[show.ROTATION](#)  
[show.SELECTED](#)  
[show.SEQUENCE](#)  
[show.SET](#)  
[show.SMILES](#)  
[show.SPACEGROUP "name"](#)  
[show.STATE \[optional name\]](#)  
[show.SYMOP \(integer\)](#)  
[show.SYMOP \[atom-expression-or-coordinate\] \[atom-expression-or-coordinate\]](#)  
[show.SYMMETRY](#)  
[show.TIMEOUT](#)  
[show.TRACE](#)  
[show.TRANSFORM](#)  
[show.TRANSLATION](#)  
[show.UNITCELL](#)  
[show.URL](#)  
[show.URL URL](#)  
[show.VARIABLES](#)  
[show.ZOOM](#)  
[show SubjectID](#)

**[slab](#)**  
[slab ON/OFF {default: ON}](#)  
[slab \[slab-percent\]](#)  
[slab HKL {h k l} or NONE](#)  
[slab -HKL {h k l}](#)  
[slab -PLANE plane\\_expression or NONE](#)  
[slab -PLANE plane\\_expression](#)  
[slab RESET](#)  
[slab SET](#)

**[spacefill or cpk](#)**  
[spacefill ON/OFF {default: ON}](#)  
[spacefill ONLY](#)  
[spacefill AUTO](#)  
[spacefill \[decimal\]](#)  
[spacefill \[integer\] %](#)  
[spacefill \[integer\] %Jmol](#)  
[spacefill \[integer\] %Babel](#)  
[spacefill \[integer\] %Babel21](#)  
[spacefill \[integer\] %Rasmol](#)  
[spacefill \[integer\] %User](#)  
[spacefill +\(solvent probe radius\)](#)  
[spacefill IONIC](#)  
[spacefill TEMPERATURE](#)  
[spacefill ADPMIN n%](#)  
[spacefill ADPMAX n%](#)

**[spin](#)**  
[spin ON/OFF {default: ON}](#)

**[ssbonds](#)**  
[ssbonds ON/OFF {default: ON}](#)  
[ssbonds \[width-angstroms\]](#)  
[ssbonds \[width-Rasmol\]](#)

**[star or stars](#)**

**[step](#)**

**[stereo](#)**  
[stereo \[stereo-viewing-angle\] {default: 5}](#)  
[stereo {default: ON}](#)  
[stereo OFF](#)  
[stereo REDBLUE \[stereo-viewing-angle\] {default: 3}](#)  
[stereo REDCYAN \[stereo-viewing-angle\] {default: 3}](#)  
[stereo REDGREEN \[stereo-viewing-angle\] {default: 3}](#)  
[stereo \[RGB-color\] \[RGB-color\] \[stereo-viewing-angle\] {default: 3}](#)

**[strand or strands](#)**  
[strand ON/OFF {default: ON}](#)  
[strand ONLY](#)  
[strand \[strand-radius\]](#)

**[structure or structure](#)**  
[structure HELIXSHEETTURNNONE \(atomExpression\)](#)

**[struts or strut](#)**  
[struts ON/OFF {default: ON}](#)  
[struts ONLY](#)  
[struts \[radius-in-angstroms\]](#)  
[struts \[radius-Rasmol\]](#)

**[subset](#)**

**[SWITCH](#)**

**[sync](#)**  
[sync &PER;|>\["appletId|appletId|syncId\] ON](#)  
[sync &PER;|>\["appletId|appletId|syncId\] SLAVE](#)  
[sync &PER;|>\["appletId|appletId|syncId\] OFF](#)  
[sync &PER;|>\["appletId|appletId|syncId\] "command"](#)

**timeout or timeouts**

timeout OFF  
timeout "id" time(ms)  
timeout "id" OFF

**trace**

trace ON/OFF {default: ON}  
trace ONLY  
trace [trace-radius]

**translate**

translate X or Y (decimal)  
translate X or Y (decimal) %  
translate X or Y (decimal) NM or ANGSTROMS  
translate Z (decimal) %  
translate Z (decimal) NM or ANGSTROMS

**translateSelected**

translateSelected {x y z} {atomSet}

**try**

**unbind**

unbind  
unbind [mouse-action]  
unbind [jmol-action]  
unbind "script"  
unbind [mouse-action] [jmol-action]  
unbind [mouse-action] "script"

**unitcell**

unitcell ON/OFF {default: ON}  
unitcell (decimal)  
unitcell DOTTED  
unitcell {i j k}  
unitcell TICKS X|Y|Z {major,minor,subminor} FORMAT [%0.2f,...]

**VAR**

**vector or vectors**

vector ON/OFF {default: ON}  
vector [diameter-pixels]  
vector [radius-in-angstroms]  
vector SCALE [vector-scale]

**vibration**

vibration ON/OFF {default: ON}  
vibration [time-period]  
vibration PERIOD [time-period]  
vibration SCALE [vibration-scale]

**WHILE or**

**wireframe**

wireframe ON/OFF {default: ON}  
wireframe ONLY  
wireframe [radius-in-angstroms]  
wireframe [radius-Rasmol]

**write**

**write (export)**

write IDTE "fileName"  
write MAYA "fileName"  
write POVRAY "fileName"  
write VRML "fileName"  
write X3D "fileName"

**write (image, frames)**

write FRAMES atom\_expression width height "fileName.jpg"  
write IMAGE JPG n "fileName"  
write IMAGE JPG64 n "fileName"  
write IMAGE PNG n "fileName"  
write IMAGE PPM "fileName"

**write (info)**

write FUNCTIONS "fileName"  
write HISTORY "fileName"  
write JMOL "fileName"  
write MENU "fileName"  
write STATE "fileName"  
write STATE LOCALPATH "path" "fileName"  
write STATE REMOTEPATH "path" "fileName"  
write VAR [variable name] "fileName"  
write ZIP "fileName"  
write ZIPALL "fileName"

**write (model)**

write COORDS SPT|XYZ|MOLPDB "fileName"  
write FILE "fileName"  
write FRAMES atom\_expression width height "fileName.jpg"  
write PROPERTIES  $\frac{h}{k}$  "fileName"  
write QUATERNION  $\frac{h}{k}$  "fileName"  
write RAMACHANDRAN  $\frac{h}{k}$  "fileName"  
write MESH "fileName"  
write POINTGROUP "fileName"

**write (object)**

write ISOSURFACE "fileName"  
write MO "fileName"  
write POINTGROUP DRAW "fileName"

**zap**

zap  
zap (atom expression)  
zap

**zoom**

zoom ON/OFF {default: ON}  
zoom IN  
zoom OUT  
zoom [percent-zoom]  
zoom (atom expression) or {x y z} [percent-zoom]  
zoom (atom expression) or {x y z} + or - delta  
zoom (atom expression) or {x y z} \* or / factor

**zoomto**

zoomto  
zoomto IN  
zoomto OUT  
zoomto [time-in-seconds] (atom expression) or {x y z}  
zoomto [time-in-seconds] (atom expression) or {x y z} [percent-zoom] xTrans yTrans  
zoomto [time-in-seconds] (atom expression) or {x y z} + or - delta xTrans yTrans  
zoomto [time-in-seconds] (atom expression) or {x y z} \* or / factor xTrans yTrans  
zoomto [time-in-seconds] (atom expression) or {x y z} 0 xTrans yTrans

last updated: Jul 27, 2010

[html](#)  
[xml docbook](#)

## Jmol 12.0 new features – 7/27/2010

### 176. TRY/CATCH

Jmol 12.0.RC27 adds the try/catch option similar to that of JavaScript and Java, allowing you to catch script errors, including file load issues, BEFORE they trash your model.

- [try{load "garbage"}catch\(e\){prompt e}spin on](#) # notice that we get the full error message and that the script continues on to start the model spinning

### 175. prompt() function

Jmol 12.0.RC27 adds the PROMPT function, which allows user-typed input or user response to option buttons.

- [if \(prompt\("Do you want spacefill","Yes|No",true\)=="Yes"\){spacefill only}else{wireframe 0.15:spacefill 23%}](#)
- [x = prompt\("Enter a script command",x\);if \(x != "null"\){script inline @x}](#)
- [x = prompt\("What sort of rendering do you want?","Spacefill","Wireframe","Ball&Stick","cancel"\).true;script inline @ {\(x == 1 ? "spacefill only" : x == 2 ? "wireframe only" : x == 3 ? "wireframe only:wireframe reset:spacefill reset" : ""\)}](#)

### 174. associative arrays

Jmol 12.0.RC27 completes the full-fledged scripting capability of Jmol by adding associative arrays -- arrays that allow you to retrieve information using a string-based key rather than a number. See the documentation for details.

- [load 1crn.pdb;a={"area1":{"helix"},"area2":{"sheet"}}](#)
- [display @ {a\["area1"\]}](#)
- [display @ {a\[prompt\("What do you want to see?","area1|area2",true\)\]}](#)

### 173. SWITCH/CASE

Jmol 12.0.RC27 adds one final piece of JavaScript-like scripting -- SWITCH/CASE. See the documentation for details.

### 172. PROMPT command

Jmol 12.0.RC26 adds the PROMPT command to allow pausing a script until the user presses OK. If no parameter is given, PROMPT returns a script stack trace showing where it is. This may be useful for debugging script files.

- [load caffeine.xyz;refresh:prompt "caffeine is loaded"](#)

### 171. FEATURE CHANGE: halos ON

Jmol 12.0.RC26 fixes a long-standing problem with halos and selection halos. With "halos ON" (the default size) they are supposed to track the spacefill size, appearing slightly larger. In previous versions their size was simply set to 20% of the default van der Waals radius. You can still set them to that if you want, with **halos 20%**.

### 170. wireframe/spacefill RESET

Jmol 12.0.RC26 adds the RESET option, which returns wireframe or spacefill to the Jmol default settings. Thus, a simple "Ball&Stick" rendering is simply **wireframe only;wireframe reset;spacefill reset**. (The "wireframe only" removes any other rendering.)

- [load caffeine.xyz;wireframe only](#)
- [wireframe;delay 1;wireframe reset](#)
- [spacefill;delay 1;spacefill reset](#)
- [dots only;delay 1;wireframe only;wireframe reset;spacefill reset](#)

### 169. 3D Jmol in PDF files

... well, ALMOST! Starting with Jmol 12.0.RC26, you can easily write the files necessary to add a Jmol-like 3D model into PDF files. To do this you are going to need a few pieces of software besides Jmol. The idea is Jmol --write--> IDTF + TEX --idtfConverter--> U3D + TEX --pdflatex--> PDF. (I think that equation is balanced!) It is a bit of a process, but it is not too bad. Once you get it working, the process takes a few seconds only. Those who are TeX savvy are at an advantage here, but if you have Adobe Acrobat PRO you should not need to know any TeX at all to do this. Sample files are in the [pdf](#) subdirectory. [test1.idtf.pdf](#) is an example.

### 168. full 3D-SEARCH SMILES/SMARTS and bioSMILES/bioSMARTS implementation

Jmol 12.0 implements a full version of [SMILES](#) and [SMARTS](#) matching, augmenting that with extensive capabilities for 3D conformation matching (distance, angle, and torsion ranges, as well as conformational RMSD comparison measurements of selected atoms) and additional extensions to search biomolecular substructure such as sequence and base-pairing. All aspects of Daylight SMARTS matching are implemented -- see the [3D-SEARCH specifications](#) for details.

### 167. load DATA

Jmol 12.0.RC25 merges the DATA "model..." and DATA "append..." methods with the LOAD command, thus extending all of the options to the LOAD command to the DATA command.

- [load data "model molfile"|line1|line2|line3| 5 4 0 0 0|0 0 0 0 0.00000000 C|0 0 1 0.225207 0.00000000 H|0.8855288 -0.511260 0.00000000 H|-0.885528 -0.511260 0.00000000 H|1 0 0 0.00000000 C| 5 1 1| 4 1 1| 2 1 1| 1 3 1|end "model molfile" FILTER "2D"](#) # Here we are adding 2D-to-3D minimization -- the vertical bars are only necessary here because of the JavaScript being used. Generally one would use new-line characters
- [load ""](#) # same file loaded again, this time with no 2D conversion

### 166. set dotScale

Jmol 12.0.RC24 adds the capability to adjust the size of the dots.

- [load caffeine.xyz;dots on](#)
- [set dotScale 1](#)
- [set dotScale 2](#)
- [set dotScale 3](#)
- [set dotScale 4](#)
- [load 1crn.pdb;dots on](#)
- [set dotScale 1](#)
- [set dotScale 2](#)
- [set dotScale 3](#)
- [set dotScale 4](#)

### 165. show SMILES

Jmol 12.0.RC25 adds show SMILES; same as **print {selected}.find("SMILES")**

- [load cholesterol.mol;show SMILES](#)

### 164. MAPPROPERTY command

Jmol 12.0.RC23 adds a powerful new way to transfer properties from one atom set to another. The operation involves identifying two sets of atoms and properties and also a common "key" property such as atomno or resno. If no key is given, atomno is assumed. A shortcut allows quick transfer of atom selection.

- [load 1crn.pdb;plot ramachandran](#)
- [mapProperty {1.1}.temperature {2.1}.property t RESNO; color {2.1} property t "bwr"](#)
- [selectionHalos on; select 1.1 and within\(1, group, pro\);mapProperty SELECTED {2.1}](#)
- [select {1.1};calculate straightness;mapProperty {1.1}.straightness {2.1}.property s; color {2.1} property S](#)

### 163. {\*}.find("SEQUENCE")

Jmol 12.0.RC23 allows quick display of sequence information in Jmol bioSMILES notation, either with or without cross-links.

- [load 1crn.pdb](#)
- [show sequence](#) # standard way
- [print {\\*}.find\("SEQUENCE"\)](#) # new way
- [print {\\*}.find\("SEQUENCE", true\)](#) # with cross-links

### 162. new QuaternionFrame settings

Jmol 12.0.RC23 adds several new quaternion frame settings. See the [Jmol 12.0 documentation](#) for details.

### 161. PLOT command

Jmol 12.0.RC23 combines the quaternion and ramachandran commands into a new command, **plot**, and adds to that the capability to graph any two or three atom properties. Optional parameters for the PROPERTIES option include MIN and MAX. All plots are scaled to fit a 200 x 200 x 200 angstrom box using a unit cell to convert actual (now "fractional") units to Jmol.xyz units. This allows access still to the actual data coordinates via fractional coordinate notation (such as {2.0 3.0 5.0/1}).

- [load 1crn.pdb](#)
- [plot quaternion difference](#)
- [plot ramachandran](#)
- [zap !1.1:select \\*CA:plot PROPERTIES phi psi resno](#)
- [zap !1.1:select \\*CA:plot PROPERTIES phi psi resno MIN {-180 -180.0} MAX {180 180.30}](#)
- [zap !1.1:select \\*CA:plot PROPERTIES phi psi structure; select visible; color structure; spin on, delay 2; spin off](#)
- [zap !1.1:select \\*:plot PROPERTIES atomno temperature](#)
- [select 1.1:plot PROPERTIES structure temperature atomno; select visible; color structure](#)
- [load 1d66.pdb](#)
- [select \\*P:plot PROPERTIES eta theta resno; select 2.1.wireframe only # a "DNA worm"](#)

### 160. eta/theta for nucleic acids

Jmol 12.0.RC23 adds nucleic acid conformation properties eta and theta as per Carlos M. Duarte, Leven M. Wadley, and Anna Marie Pyle, RNA structure comparison, motif search and discovery using a reduced representation of RNA conformational space, Nucleic Acids Research, 2003, Vol. 31, No. 16 4755-4761. eta is the C4'[i-1]-P[ij]-C4'[ij]-P[i+1] dihedral angle; theta is the P[ij]-C4'[ij]-P[i+1]-C4'[i+1] dihedral angle.

### 159. axes center {x y z}

Jmol 12.0.RC22 adds the capability to move the axes origin to a new center.

- [load caffeine.xyz; axes on; axes molecular](#)
- [axes center {atomno=1}](#)

### 158. set picking measure SEQUENCE

Jmol 12.0.RC22 adds the ability to pick two atoms of a biomolecule and have the sequence containing them displayed.

- [load 1crn.pdb; set picking measure sequence](#) # now start picking atoms

### 157. set cartoonBaseEdges

Jmol 12.0.RC21 introduces a new way to render nucleic acid cartoons. **set cartoonBaseEdges TRUE** tells Jmol to display nucleic acid bases as triangles that highlight the sugar edge (red), Watson-Crick edge (green), and Hoogsteen edge (blue). See Nasalean L, Strombaugh J, Zirbel CL, and Leontis NB in [Non-Protein Coding RNAs](#), Nils G. Walter, Sarah A. Woodson, Robert T. Batey, Eds., Chapter 1, p 6.

- [load 1d66.pdb; display nucleic; reset; center {35.5123 34.30653 27.29455}; rotate z 140.14; rotate y 11.62; rotate z -138.18; zoom 380.63; set rotationRadius 51.55; cartoons only](#)
- [set cartoonBaseEdges true](#)
- [set cartoonBaseEdges false](#)

### 156. isosurface ... MAP MEP functionType

Jmol 12.0.RC19 adds preliminary molecular lipophilic potential mapping by allowing MEP (molecular electrostatic potential) to be mapped using functions other than the standard Coulomb potential (1/d). The data for the mapping comes by default from the partialCharge atom property, but can easily be drawn from any property or a variable. See the [Jmol 12.0 documentation](#) for details.

### 155. measure search("...")

Jmol 12.0.RC19 adds a simple way to create measurements based on SMARTS searching.

- [load caffeine.xyz](#)
- [measure delete; select \\*; measure search\("Ocn"\)](#) # measure the angles associated with the CO bonds
- [measure delete; select \\*; measure search\("cn"\)](#) # measure all aromatic CN bonds

### 154. substructure() function deprecated

Jmol 12.0.RC19 deprecates the substructure() function, which used a SMILES string to search for substructure. Really, SMILES strings were never intended to be for searching; that is what SMARTS is for. The find() and search() functions are recommended.

- [load caffeine.xyz; selectionhalos on; select none](#)
- [select search\("ccc"\)](#)
- [select search\("c\)cO"\)](#)
- [print {\\*}.find\("\[r5\]"\)](#)
- [print {\\*}.find\("SMILES", "Cn1c\(\[O\]\)\[c\]2n\(C\)\[cH\]\[n\]\[c\]2n\(C\)\[c\]\(\[O\]\)1"\)](#) # a SMILES string for caffeine -- using "SMILES" checks for a whole-structure match

### 153. atropisomer SMARTS matching

Jmol 12.0.RC16 adds a new bond symbol to SMILES/SMARTS C^C and C^^C or C!^C (SMARTS) -- atropisomer (dihedral angle) check

### 152. x.find("SMILES"/"SMARTS", "MF")

Jmol 12.0.RC16 lets you quickly find the molecular formula of the SMILES or SMARTS string associated with a set of atoms or SMILES string. The two options allow for a full all-hydrogen formula or one without hydrogen atoms.

### 151. load SCCC(C)C -- smiles string loading

Jmol 12.0.RC15 will read SMILES strings into 3D using the [smi23d server](#) at Indiana University. Only catch is that since this is a third-party server, you will have to use the [signed applet](#) to do it. Note that this service uses PCMODEL v9.1 for its conversion.

- [load SMILES "C1C\(C\)C1C@H\(F\)C1C\(=O\)C"](#)
- [load \\$CCI1CCCC1\(C\)C](#)
- [load "\\$\[H\]\[C\]@\(\[12CCCC1\]C@\(\[H\]\(C\)\[C@H\]\(CC\)\[C@\]3\(O\)CCCC\[C@\]\(\[23\]\[H\]\)" # from JME](#)
- [load SMILES "F/C=C/C=C/C" # correct](#)
- [load SMILES "F/C=C/C=C/C" # should be different](#)
- [load SMILES "CCCC" as "butane.mol" # save file on your local drive](#)

### 150. set HIGHLIGHT

Jmol 12.0.RC15 adds a new sort of selectionHalo. The HIGHLIGHT is a just a small ring around an atom. The default color is red, but you can use **color highlight** to set it to a different color. It is not an "atom property", so individual atoms cannot have their own colored highlight.

- [load caffeine.xyz](#)
- [set highlight { O }](#)
- [color highlight yellow](#)
- [set highlight off](#)

#### 149. set ModelKitMode and set allowModelKit

Jmol 12.0.RC15 represents a first attempt at creating a model kit option for Jmol. I cannot seem to get the menu to pop up using "set modelKitMode" via Javascript, but you can call it up yourself from the menu. It is under "computation". Using **set allowModelKit FALSE** you can disallow modelKitMode.

- [set modelKitMode:zap](#) # the magenta bar in the top left corner is the menu

#### 148. FIX command

The **FIX** command takes an atom expression argument like the **display** or **select** commands. It fixes the positions of atoms and ensures that no atoms of this set will be moved or dragged anywhere accidentally.

#### 147. set picking invertStereo

Starting with Jmol 12.0.RC15, selecting an atom that is part of a ring after **set picking invertStereo** will reverse the two non-ring atoms -- actually rotating them 180 degrees, not doing a planar inversion, thus preserving whatever chirality might be attached to them.

#### 146. set picking dragMinimizeMolecule -- responsive docking

With Jmol 12.0.RC15 you can move a small molecule around and watch it react to its environment. Grab the caffeine molecule and move it toward the protein. Holding SHIFT down allows rotation. CTRL-Z undoes an action; CTRL-Y does a redo.

- [load files "caffeine.xyz" "1crn.pdb";frame \\*:zoomto {1.1} 0;set picking dragMinimizeMolecule](#)
- [load files "1crn.pdb" "water-AM1.sparchive";frame \\*:isosurface select {1.1} sasurface 0 frontonly translucent 0.3;fix protein;select {2.1}; spacefill;set picking dragMinimizeMolecule # now drag the water around the surface](#)

#### 145. set picking dragMinimize

Wouldn't it be fun to have a real "sculpture" mode, where you can play with a molecule like an artist does with clay, molding it to the shape you want -- within the limitations of the medium? (Don't try this with a protein.) Well, it's very possible this is a bad idea, but Jmol 12.0.RC13 adds this capability.

- [load mecy.jme;set picking dragMinimize;set minimizationSteps 500;set echo top left;echo drag an atom...set echo bottom left;echo @{\\_minimizationEnergy\\_} just grab atoms and move them. Or click on an atom to further minimize. I know...questionable pedagogy...Actually, there is a sort of lesson here in how minimization works...](#)

#### 144. set picking dragAtom

Jmol 12.0.RC13 adds **set picking dragAtom**, allowing dragging of atoms to new locations. Attached hydrogens atoms are dragged along with the selected atom.

- [load caffeine.xyz;set picking dragAtom; # go ahead...](#)

#### 143. drag-and-drop to signed applet and from browsers

Jmol 12.0.RC13 adds drag-and-drop capability for the signed applet and adds MULTIPLE-FILE drag-and-drop for both signed applet and application. Just fire up a page such as [drop.htm](#) that uses the signed applet and then drag from a directory listing into the applet as many files as you like. They will be loaded into different frames, and **frame \*;reset** will be issued so that they are all displayed. (PNG and JPG images

created by Jmol can be loaded this way, but only one at a time, since they reset the state when loaded.) For standard model files, if **set defaultLoadScript** is defined, then that script will be executed. (For example, a simple one might be **set defaultLoadScript 'select protein or nucleic;cartoons only'**.) Electron density MAP files may also be dropped in. Or you can simply clip file name(s) in a file directory, then CTRL-V into Jmol loads the file(s).

#### 142. COMPARE with SMILES or SMARTS allows conformational testing and alignment

Jmol 12.0.RC12 adds the capability to align two structures based on SMILES or SMARTS atom matching. The basic idea is to use a SMILES (whole molecule) or SMARTS (substructure) description to find the atoms in one structure that correlate one-for-one with atoms in the second structure, then find the rotation and translation that best aligns them. If no actual atom moving is desired, you can get the standard deviation alone using the **compare()** function with the "STDDEV" option. A return of "NaN" indicates that the desired SMILES/SMARTS match could not be made in one or the other structure.

- ["2-CiBu.spt"](#) # this script loads two versions of (R)-2-chlorobutane and aligns them
- [load inline "5.4 C 5.88 -4.59 C 7.09 -3.89 C 8.30 -4.59 C 4.73 -3.79 Cl 7.09 -2.49 1 2 1 2 3 1 1 4 1 2 5 - 2";load append "2-CiBu.mol";frame \\*:select \\*:wireframe only;label %\[atomno\];moveto /\\* time\\_axisAngle \\*/ 1.0 { 334 -913 -234 46.04} /\\* zoom\\_translation \\*/ 57.58 0.13 0.13 /\\* center\\_rotationRadius \\*/ {3.205987 -2.11722 -0.18840191} 4.361869 /\\* navigation\\_center\\_translation\\_depth \\*/ {0.0 0.0 0.0} - 14.783268 -18.059193 0.0;](#)
- [print compare {2.1} {1.1} "SMILES" "CC\(C\)CC" "stddev"](#); # just getting the standard deviation
- [print compare {2.1} {1.1} "SMILES" "C\[CH@\]\(C\)CC" "stddev"](#); # not this enantiomer
- [print compare {2.1} {1.1} "SMILES" "C\[CH@\]\(C\)CC" "stddev"](#); # there you go
- [compare {2.1} {1.1} SMILES "CC\(C\)CC" rotate translate](#)
- [select 2.1;color labels yellow;set labeloffset -5 10;color yellow:zoomTo \[visible\] 0](#)
- [script "2-CiBu.spt" lines 1-2;reset;frame \\*:compare {2.1} {1.1} SMARTS "\[CH3\]\[CH@\]\(C\)\[CH2\]" rotate translate # alignment based just on the stereocenter](#)

#### 141. JmolSmilesApplet.jar

Jmol 12.0.RC11 adds a new light-weight applet (only 43K) that checks SMILES strings. This is particularly useful for comparison of drawn stereochemistry, which is not possible using JME alone. See [JmolSmiles.htm](#), [JmolSmilesTest.htm](#), and [JmolSmilesApplet.jar](#).

#### 140. SMILES stereochemistry matching

Jmol 12.0.RC11 allows unprecedented matching of non-canonical SMILES strings independent of any 3D structure. This includes both atom and bond stereochemistry, including cis/trans, allene, tetrahedral, square planar, trigonal bipyramidal, and octahedral stereochemistry. Starting with a SMILES string from some source, you can test for an equivalent structure WITHOUT any need for "canonicalization" of the SMILES (that is, turning it into some standard form). While canonicalization is important for database searching just in terms of speed, with this addition to Jmol, **canonicalization is no longer necessary for pattern matching between two SMILES strings or pattern searching of a SMILES string using a SMARTS pattern**. This feature is completely independent of any actual Jmol model (although that, too, can be tested against a SMILES or SMARTS string). Thus, for example, you can have a user create a 2D structure in JME and use Jmol to test its equivalence to some reference SMILES or SMARTS string you are expecting.

- [print "CCCC\[C@\]\(F\)\(Cl\)" find\("smiles","O\(CCC\)\[C@\]\(F\)\(Cl\)I"](#)
- [print "O\[C@\]\(F\)\(Cl\)" find\("smiles","\[C@\]\(O\)\(F\)\(Cl\)I"](#)
- [print "O\[C@\]\(F\)\(Cl\)" find\("smiles","\[C@@\]\(O\)\(Cl\)\(F\)I"](#)
- [print "OC\(Cl\)=\[C@\]=C\(C\)F" find\("smiles","OC\(Cl\)=\[C@AL1\]=C\(C\)F"](#)
- [print "OC\(Cl\)=\[C@\]=C\(C\)F" find\("smiles","OC\(Cl\)=\[C@AL2\]=C\(C\)F"](#)
- [print "F\[Po@SP1\]\(Cl\)\(Br\)" find\("smiles","F\[Po@SP1\]\(Cl\)\(Br\)I"](#)
- [print "F\[Po@SP1\]\(Cl\)\(Br\)" find\("smiles","F\[Po@SP2\]\(Br\)\(Cl\)I"](#)
- [print "F\[Po@SP1\]\(Cl\)\(Br\)" find\("smiles","F\[Po@SP3\]\(Cl\)\(DBr\)"](#)
- [print "S\[As@@\]\(F\)\(Cl\)\(Br\)C=O" find\("smiles","S\[As@@\]\(F\)\(Cl\)\(Br\)C=O"](#)
- [print "S\[As@@\]\(F\)\(Cl\)\(Br\)C=O" find\("smiles","O=C\[As@\]\(F\)\(Cl\)\(Br\)S"](#)
- [print "S\[Co@@\]\(F\)\(Cl\)\(Br\)C=O" find\("smiles","S\[Co@@\]\(F\)\(Cl\)\(Br\)C=O"](#)
- [print "S\[Co@@\]\(F\)\(Cl\)\(Br\)C=O" find\("smiles","O=C\[Co@\]\(F\)\(Cl\)\(Br\)IS"](#)

- [print "F/C=C/F".find\("smiles","F/C=C/F"\)](#)
- [print "F/C=C/F".find\("smiles","F\C=C\F"\)](#)
- [print "C/\(F\)\(Cl\)=C/F".find\("smiles","C/\(Cl\)\(F\)=C\F"\)](#)
- [print "C/\(F\)\(Cl\)=C/F".find\("smiles","C\\F\)=C/\(Cl\)F"\)](#)

139. `{*}.find("smartsString",asArray)`

Jmol 12.0.RC11 adds the SMARTS searching to the `find()` function. If the optional second parameter is TRUE, the return is an array of atom sets.

- [load caffeine.xyz](#)
- [print {\\*}.find\("cn"\)](#)
- [print {\\*}.find\("cn", true\)](#)

138. `load @x` where x is an array of file names

Jmol 12.0.RC11 adds the capability to load a set of files that are defined in an array variable.

137. `select search()` -- 3D-SMARTS

Jmol 12.0.RC11 adds a completely new way to select atoms based on the [SMARTS](#) chemical informatics search language. **3D-SEARCH** is an almost exact implementation of SMARTS, differing primarily in how 3D-SEARCH defines "aromatic" and also adding a "search and select" capability as opposed to just "search." The search language allows very precise selection of atoms based on connectivity. Examples include "all alpha carbons" `{C}C=O`, "all meta-related groups" `{A&!H}aaa{A&!H}`, "all trans double bonds" `*{C=C}/*`, and "all biphenyl connecting atoms" `a:a`. All stereochemical capability of SMARTS described on that page is implemented, including double-bond, allenic, tetrahedral, square planar, trigonal pyramidal, and octahedral stereochemistry. "Aromatic" is defined simply as any atom in a flat ring that has all sp<sup>2</sup> hybridization. This is a compromise definition meant to be inclusive of some compounds that by Hueckel analysis would not be considered aromatic. In addition, setting `{ }` around an atom or group of atoms specifies to select only those atoms, and an optional second parameter allows selecting atoms within a specific subset of atoms. Variables may be assigned within the SMARTS string (which unlike the specification at the Daylight site does allow white space and comments). Just a few examples are shown below.

- [load ketone.jme](#)
- [selectionhalos on;select search\("{C}C=O"\)](#) # just the alpha carbons
- [selectionhalos on;select search\("{C&!H0}C=O"\)](#) # just the alpha carbons with H atoms
- [selectionhalos on;select search\("{A&!H}aaa{A&!H}"\)](#) # just the meta groups
- [load caffeine.xyz;selectionhalos on;select none](#)
- [select search\("a"\)](#) # all "aromatic"
- [select search\("\[nD2\]"\)](#) # aromatic N with only two bonds
- [select search\("\[Od1\]"\)](#) # oxygen with only one non-hydrogen connection
- [select search\("\[cv3\]"\)](#) # trivalent aromatic carbon
- [select search\("\[{cv3}\]O"\)](#) # trivalent aromatic carbon attached to oxygen
- [select search\("\[cr5\]"\)](#) # aromatic carbon in a five-membered ring
- [select search\("\[r6\]"\)](#) # six-membered ring
- [select search\("\[r9\]"\)](#) # nine-membered ring
- [select search\("\[R2\]"\)](#) # in two rings (ring-fusion atoms)
- [select search\("\[!r&!H\]"\)](#) # non-ring non-hydrogen atoms
- [select search\("\[!r&!H\]>\\*\[R2\]"\)](#) # non-ring non-hydrogen atoms two atoms from a ring-fusion
- [select search\("\[S\(\[!rja\]\\*\\*\[R2\]\)&!S\(\\*\\*\[R2\]\)\]"\)](#) # non-ring atoms attached to an aromatic ring and three atoms from a ring-fusion but not two atoms from a ring fusion atom
- [select search\("c",search\("\[r6\]"\)\)](#) # only aromatic carbons in 6-membered rings
- [select search\("\\$Aro1="\[a&!\\$aC\]"\)](#) (i.e. aromatic not attached to aliphatic carbon);

[\\$CarbonylO="O";\[!\\$CarbonylO\];\[!\\$Aro1\]](#)

136. Spartan/Cygraph FILTER "noOrient"

Jmol 12.0.RC10 adds the noOrient filter for loading Spartan and Cygraph (CAGe) files. Generally these files are loaded with a default orientation that was the orientation when the file was saved.

135. `isosurface "=nnnn"`

Jmol 12.0.RC10 adds the capability to automatically load electron density maps from the Uppsala Electron Density Server into the application and signed applet. Supporting this are the global variables `edsUrlCutoff` and `edsUrlFormat`, which set the method of getting cutoff and file from electron density server.

134. `isosurface lattice {a b c}`

Jmol 12.0.RC9 adds the capability to duplicate isosurface areas based on the unit cell lattice. This is a rendering option, so it can be applied any time after an isosurface is created. It is best done with packed unit cells.

- [load quartz.cif {1 1 1};zoom 75;center {3/2 3/2 3/2}](#)
- [isosurface slab unitcell vdw](#)
- [isosurface lattice {3 3 3}](#)

133. `wireframe ONLY` and `wireframe -x.y`

Jmol 12.0.RC7 adds two new option for wireframe, stars, halos, spacefill, cartoons, etc. "ONLY" removes all other atom rendering; "-x.y" does the same, but allows setting of the radius.

- [load 1crn.pdb](#)
- [spacefill only](#)
- [cartoons only](#)
- [wireframe -0.2](#)

132. PARALLEL MULTIPROCESSING

Jmol 12.0.RC6 introduces parallel processing for Jmol. Jmol 12 will be able to use multiple processors on a multiple-CPU machine. Basically what you can do is to tell Jmol which statements in a script you want to run in parallel, and it will do that. The way this is done is to create a function using the keyword **PARALLEL** in place of the keyword **function**. Within that block of code, any group of commands surrounded by **PROCESS{ }** will be collected and run in parallel just before Jmol returns from the function. Any commands NOT within these sets will be run BEFORE any **PROCESS** commands. For example:

```
parallel twoIsosurfaces(model1, model2) {
  var x = 1
  process {
    isosurface s1 model @model1 molecular; color isosurface red
  }
  process {
    isosurface s2 model @model2 molecular; color isosurface green
  }
  x = 2
}
```

`load files "1crn.pdb" "1blu.pdb"`  
`twoIsosurfaces("1.1", "2.1")`  
`frame *`

In this case, the variable x will be 2 BEFORE the isosurfaces are created. See also [multi-mo.txt](#), [multi-surface.txt](#), and [multiProcessTest.txt](#)[multiProcessTest.txt](#). You can selectively turn on and off the use of multiprocessors using `set multiProcessor`. If this setting cannot be set true, then it means you do not have a multiprocessor machine. Not all processes will work; currently the only implemented parallel processes are for isosurfaces and molecular orbitals. The parallel capability of Jmol should be considered experimental at this time.

- [set multiProcessor false;script multi-mo.txt](#)
- [set multiProcessor true;script multi-mo.txt](#)
- [set multiProcessor false;script multi-surface.txt # be patient!](#)
- [set multiProcessor true;script multi-surface.txt # somewhat faster on my machine](#)

131. `set minimizationSilent` and `minimize SILENT`

Jmol 12.0.RC5 adds the option to make minimization totally silent. The default value is FALSE. In association with **set useMinimizationThread FALSE**, this option is used automatically to effect the loading of 2D JME strings into Jmol as 3D objects directly. Note that in some cases, more than the default **set minimizationSteps 100** may be necessary for a satisfying result.

- [load test.jme](#)
- [load test.jme FILTER "NOMINIMIZATION";wireframe only](#) # the unminimized structure - note that Jmol has interpreted some bond angles as representing stereochemistry.
- [load stereo.jme](#) # delay is due to processing the minimization

### 130. load INLINE "JME string" and load "@x"

Jmol 12.0.RC5 allows inline load of data that have no new-line characters (JME strings) using load INLINE, and any model using load "@x", where x is a Jmol variable containing the model. Note that JME models are automatically turned into 3D.

- [load inline "5 4 C 5.88 -4.59 C 7.09 -3.89 C 8.30 -4.59 C 4.73 -3.79 Cl 7.09 -2.49 1 2 1 2 3 1 1 4 1 2 5 - 2";](#)
- [x = load\("ch3cl.mol"\);load "@x"](#)

### 129. select CYSTINE

Jmol 12.0.RC5 adds select CYSTINE synonymous with within(group, cys.sg and connected(cys.sg))

- [load 1crn.pdb;select none;selectionhalos on](#)
- [select CYSTINE](#)

### 128. 2D MOL reader to 3D

Jmol 12.0.RC5 adds the FILTER "2D" option to read MOL files that have all zero z coordinates and automatically minimize the structure into 3D, as with JME, below.

- [load flat.mol;rotate X 80](#) # still 2D
- [load flat.mol FILTER "2D";rotate X 80](#)

### 127. JME reader reads stereochemistry and automatically turns 2D to 3D with addition of H atoms

The JME format is a two-dimensional file format that is very easy to create using, for example, [JmeToJmol.htm](#) but not easily adapted to 3D (until now!). Jmol 12.0.RC5 adds the automatic addition of H atoms and transformation of simple JME structures (including stereochemistry) to 3D. The 2D-to-3D conversion can be prevented using the FILTER "noMinimization" option. The transfer is not perfect, and it may take some testing to get this right.

- [load stereo.jme](#)
- [load stereo.jme FILTER "NOMINIMIZATION";](#)

### 126. select within(SEQUENCE,"1-letter-code sequence")

Jmol 12.0.RC5 allows selecting groups that are in groups that are within a specific sequence such as "GCAUGGC".

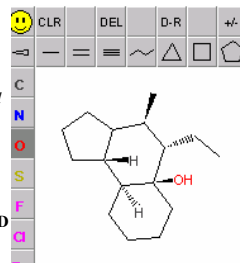
- [load 1d66.pdb](#)
- [display within\(SEQUENCE,"GAC"\)](#)

### 125. within(nResidues, GROUP, {atoms})

Jmol 12.0.RC5 will allow selection of all groups within a given number of residues of a specified set of atoms.

- [load 1d66.pdb](#)
- [display within\(3.GROUP,within\(SEQUENCE,"GAC"\)\)](#)

### 124. show BASEPAIRS



Jmol 12.0.RC5 will display a table of base pairs.

- [load 1d66.pdb](#)
- [show basepairs](#)

### 123. select within(nResidues, GROUP, atoms)

Jmol 12.0.RC5 adds the capability to select atoms within a certain number of residues along a chain of a specified group of atoms. This feature works for nucleic acids, proteins, and carbohydrates.

- [load 1crn.pdb;cartoons only](#)
- [display within\(HELIX\)](#) # just the helices
- [display within\(3.GROUP,within\(HELIX\)\)](#) # within 3 groups of a helix
- [load 2qnh.pdb.gz;zoom 300;wireframe -0.4;display rna;select \\*P and within\(BASEPAIR,"AC"\);label %n%;set picking center](#)
- [display within\(BASEPAIR,"AC"\)](#) # all AC pairs
- [color {within\(BASEPAIR,"AC"\)} white;display within\(5.GROUP,within\(BASEPAIR,"AC"\)\)](#) # within 5 residues of the AC pairs

### 122. select within(BASEPAIR)

Jmol 12.0.RC5 adds the capability to select atoms based on RNA or DNA base pairing. Thank you, Neena Grover!

- [load 2qnh.pdb.gz;zoom 300;wireframe -0.4;display rna;select \\*P;label %n%;select rna;set rangeSelected;](#)
- [color property polymer;set picking center;slab on;depth 40;slab 60](#)
- [display within\(BASEPAIR,"GC"\)](#) # just the GC pairs
- [display within\(BASEPAIR,"AU"\)](#) # just the AU pairs
- [color cpk;calculate hbonds {displayed} {displayed};hbonds 0.3;color hbonds yellow;set hbonds sidechain](#)
- [display rna and !within\(BASEPAIR\)](#) # bases that are not paired
- [display within\(BASEPAIR\) and !within\(BASEPAIR,"GCAU"\)](#) # noncanonical base pairs

### 121. negative sizes implies ONLY

Jmol 12.0.RC4 reads negative decimals for cartoons, wireframe, etc. as "ONLY" at the corresponding positive size in angstroms

- [load 1crn.pdb](#)
- [cartoons -0.5](#)
- [trace -0.3](#)
- [wireframe -0.2](#)

### 120. better RNA hydrogen bond calculation

Jmol 12.0.RC4 extends and improves the canonical (purine-pyrimidine) hydrogen bond calculation

- [load 2qnh.pdb.gz;zoom 300;backbone -0.5;display rna;select rna;set rangeSelected;color property polymer](#)
- [calculate hbonds {rna} {rna};hbonds 0.2;color hbonds yellow;set hbonds backbone](#)

### 119. select RANGESELECTED extended

Jmol 12.0.RC4 extends the RANGESELECTED parameter to all properties, not just temperature.

- [load 2qnh.pdb.gz;cartoons only](#)
- [select rna;set rangeSelected;color property polymer](#)

### 118. generalized hydrogen bond calculation

Jmol 12.0.RC3 will calculate hydrogen bonds of the standard sort -- connecting H atoms with other atoms. See the [documentation](#) for details. In short, **calculate HBONDS** will behave as previously with standard no-hydrogen PDB files -- creating "pseudo-hydrogen" bonds between O and N atoms. However, with PDB

files having hydrogen atoms and other files with hydrogen atoms, the **calculate HBONDS** command will generate standard hydrogen bonds from hydrogen atoms on O or N to any other atoms. If it is desired to have RasMol-like behavior even in the case of PDB files with no H atoms, one can simply **select not hydrogens**. The two settable parameters **hbondsAngleMinimum** (default value 90) and **hbondsDistanceMaximum** (default 3.25 for pseudo-hydrogen bonds; 2.5 for standard hydrogen bonds) can be adjusted to suit.

- [load maleic.cif 5;zoom 200](#)
- [calculate hbonds](#)
- [load ice.pdb;rotate z -72.64; rotate y 72.56; rotate z 88.29; translate x 0.13; translate y 0.13;](#)
- [calculate hbonds](#)
- [load Water!Liquid.xodydata](#)
- [calculate hbonds # note the missing H-bonds](#)
- [set hbondsAngleMinimum 75;select \\*.calculate hbonds;set hbondsAngleMinimum 90 # better in this case](#)
- [load 1cdr.pdb 1; rotate z -0.78; rotate y 54.15; rotate z 1.69; zoom 400.0;display sheet;wireframe only](#)
- [hbonds delete; select displayed;calculate hbonds;hbonds 0.2 # new](#)
- [hbonds delete; set hbondsRasmol TRUE; select displayed and not hydrogen; calculate hbonds;color hbonds yellow;hbonds 0.2 # RasMol version](#)
- [hbonds delete; set hbondsRasmol FALSE;select displayed and not hydrogen; calculate hbonds;hbonds 0.2 # Jmol version -- not just backbone](#)
- [hbonds delete; set hbondsDistanceMaximum 3.8;set hbondsRasmol FALSE;select displayed and not hydrogen; calculate hbonds;hbonds 0.2 # Jmol version -- not just backbone](#)

#### 117. invertSelected STEREO option

Jmol 12.0.RC2 adds the STEREO option to the **invertSelected** command. This option rotates branches connected to a center by 180 degrees around a line connecting that center and the geometric center of specified connected atoms not to invert. Simply put, this carries out the standard organic chemist's "stereochemical inversion" at that point -- which does not invert stereochemistry along the branches, but instead switches two connected branches by simple rotation only.

- [load ala\\_5\\_180\\_0.pdb; select {atomno=12};halo 1.0;color halo yellow; reset:center {1.3850001 1.8339999 -0.65849996}; rotate z -6.42; rotate y 47.52; rotate z 80.26; translate x 0.13; translate y 0.13; connect { O1 C1 } double modify;set dragged;set minimizationSteps 25;minimize addhydrogens;select H40;color yellow;](#)
- [invertSelected STEREO {atomno=12} {atomno=15 or atomno=38}](#)

#### 116. new ROTATE TRANSLATE {x y z} option

Jmol 11.9.37 extends the ROTATE command to include rotation/translation -- that is, helical or screw motion.

#### 115. new ROTATE HELIX option

Jmol 11.9.37 adds the HELIX option to the rotate command. Given a 4x4 matrix or a combination of translation and rotation, Jmol will determine the helical (screw-motion) path instead of the direct-line path of motion. Notice that the SELECTED parameter is unnecessary when in this context because the group being moved (model 2.1) is listed explicitly after the definition of the motion.

- [load 3cc2\\_900-1100.pdb;color orange;cartoons only;select 936-941 or 1025-1034;color red;load append 3cc2\\_Kt7.pdb;select 2.1;color yellow;cartoons only;frame \\*.reset:center {42.475437 88.69623 89.13157}; rotate z 76.29; rotate y 129.77; rotate z 39.64; zoom 268.05; set rotationRadius 114.69;save coord sc;kt7coord = {2.1}.xyz.all; m = compare\({2.1 and spine and 77-82}, {1.1 and spine and 936-941}\);](#)
- [restore coord sc; rotate {2.1 and spine and 77-82} @m {2.1} -3 # ...straight in](#)
- [restore coord sc; rotate HELIX {2.1 and spine and 77-82} @m {2.1} -3 # ...along a helical path](#)
- [rotate COMPARE {2.1} @kt7coord -3 # ...and back to the original position](#)

#### 114. new ROTATE COMPARE option

- [load 3cc2\\_900-1100.pdb;color orange;cartoons only;select 936-941 or 1025-1034;color red;load append 3cc2\\_Kt7.pdb;select 2.1;color yellow;cartoons only;frame \\*.reset:center {42.475437 88.69623 89.13157};](#)

- [rotate z 76.29; rotate y 129.77; rotate z 39.64; zoom 268.05; set rotationRadius 114.69;save coord sc;kt7atoms = {2.1 and spine and 77-82};kt38atoms = {1.1 and spine and 936-941};kt7coords = {2.1}.xyz.all; kt7ac = kt7atoms.xyz.all;m = compare\(kt7atoms, kt38atoms\);](#)
- [restore coord sc; compare {2.1} {1.1} @kt7atoms @kt38atoms rotate translate -2](#)
- [restore coord sc; compare {2.1} {1.1} @kt7atoms @kt38atoms rotate translate](#)
- [rotate COMPARE {2.1} @kt7coords -2](#)
- [restore coord sc; rotate compare @kt7atoms @kt38atoms {2.1}](#)
- [select {2.1}; restore coord sc; rotate selected compare @kt7atoms @kt38atoms -2](#)

#### 113. new ROTATE (matrix variable)

Jmol 11.9.37 adds the capability to define rotations in the ROTATE command as 3x3 or 4x4 matrices. In the case of a 4x4 matrix, the indicated translation will also be conducted along with the rotation to provide a screw motion; in the case of a 3x3 matrix, one can use the new TRANSLATE option for the ROTATE command to also apply a translation separately.

#### 112. new COMPARE function

Jmol 11.9.37 adds the **compare** function. The function takes either two sets of atoms or two arrays of atom positions (or any other sort of positions) and returns a 4x4 matrix **m** indicating the rotation (**m%1**) and translation (**m%2**) required to BEST transform set 1 into set 2. Unlike the COMPARE command, no filtering is done -- every atom or position in set1 is compared with every position in set2. An optional third parameter "stddev" allows return of just the RMSD for this comparison.

- [load files "3CC2\\_HmKt7.pdb" "3CC2\\_HmKt38.pdb";frame \\*.cartoons only;color {1.1} red;color {2.1} yellow](#)
- [m = compare\({2.1 and spine and 936-941}, {1.1 and spine and 77-82}\); show m](#)
- [print compare\({2.1 and spine and 936-941}, {1.1 and spine and 77-82}, "stddev"\)](#)
- [print m%1](#)
- [print m%2](#)

#### 111. new 4x4 matrix functions

Jmol 11.9.37 adds %1 and %2 modulus for 4x4 matrices: m%1 returns the 3x3 rotation matrix, and m%2 returns the translation vector. Note that quaternion(m%1) then returns a quaternion equivalent of the rotation matrix, which has its own % operations.

#### 110. COMPARE command animation

With Jmol 11.9.37, the COMPARE command adds the option to indicate the number of seconds to use to animate the comparison. This is primarily just for WOW effect. But associated with new ROTATE SELECTED options, can be reversed.

- [load 3cc2\\_900-1100.pdb;color orange;cartoons only;select 936-941 or 1025-1034;color red;load append 3cc2\\_Kt7.pdb;select 2.1;color yellow;cartoons only;frame \\*.zoomto 2 {\\*} 0 # the two together in their native orientations.](#)
- [compare {2.1} {1.1} SUBSET {spine} ATOMS {77-82} {936-941}, {92-93 or 97-100} {1025-1026 or 1031-1034} ROTATE TRANSLATE 3](#)

#### 109. ROTATE options

Jmol 11.9.37 introduces a number of new options for the ROTATE command. First, **rotateSelected** is deprecated. The SELECTED option for the **rotate** command takes care of this need. More significantly,

#### 108. rotateSelected deprecated

Jmol 11.9.37 deprecates the **rotateSelected** command. Just add the **SELECTED** option to the **rotate** command.

#### 107. translateSelected deprecated

Jmol 11.9.37 deprecates **translateSelected**. Simply add the **SELECTED** option to the **translate** command.

#### 106. Natural Bond Orbitals

**NBO** PLOT output may be visualized by Jmol. Starting with version 11.9.36, Jmol can LOAD any of the files xx.31 - xx.41 as well as the ASCII GenNBO output file xx.48 (sometimes referred to as the "xx.nbo" file). These files are generated by adding PLOT to any NBO parameter list (\$NBO \$END), which may appear in Gaussian, GAMESS, Jaguar, Psi, QChem, or GenNBO input decks. The xx.48 file may be used as input for visualization specifically if AONBO=P is added to the input parameters. (This visualization is equivalent to that of xx.37 files -- NBO orbitals in AO basis.) Any of the files xx.31 - xx.41 may be used in the LOAD command (or drag-dropped into the Jmol application) provided the necessary xx.31 and xx.46 files are also present in the same directory. Note that JPG and PNG files create by Jmol using the WRITE command and stored in the directory of the .nn files contain a full state description along with the image and can be loaded or dragged into Jmol the same as the original .nn file, with the added benefit of recreating the exact 3D model state as shown in their image.

- [load t.37](#) # also reads t.31 (atomic orbital basis set) and t.46 (orbital labels)
- [mo homo](#)
- [mo homo - 1](#)
- [mo homo - 2](#)

#### 105. translucent color schemes

Jmol 11.9.35 adds a translucent option for color schemes. The idea is that a color scheme can go from "transparent" to "opaque" while it goes from one color to another.

- [load 1cm.pdb:cartoons only;color straightness](#)
- [color "BW" TRANSLUCENT;color property straightness](#)
- [load 3hyd.pdb:reset:center {16.626621 2.7783334 1.7281427}; rotate z 100.36; rotate y 112.6; rotate z -72.02; zoom 200; translate x 0.13; translate y 0.13;boundbox scale 1.2 {tyr};isosurface boundbox SIGMA 1 color DENSITY colorscheme "BW" TRANSLUCENT "3hyd\\_map.ccp4.gz"](#)

#### 104. property SELECTED

Jmol 11.9.35 adds SELECTED to the list of properties of atoms that can be checked, averaged, listed, and set. The value of the property is 1.0 if an atom is selected and 0.0 if not. As for all properties, the value for a set of atoms is the average, which in this case is the fraction of atoms selected. The example used here uses {""}.selected to get the fraction of all atoms selected and displays that in an echo at the top left.

- [load caffeine.xyz:selectionhalos on:select none; select carbon;set picking select;set echo top left; echo @{""" + \(100\\*{""}.selected\)%1 + "% of the atoms are selected"} # click on atoms to change the selection.](#)

#### 103. new COMPARE command

Jmol 11.9.35 introduces the capability to compare two models and to reorient one model relative to another based on a given atom-atom coordinate pairing or quaternion-based group-group orientation pairing. References to the atom-atom correlation algorithm can be found in the literature [1] and [2]. Quaternion-based orientation pairing is an unpublished technique specific to Jmol at this point. It minimizes the standard deviation of the correlated quaternion frames for groups in the two models using spherical averaging. (Results of this option depend upon the setting of **set quaternionFrame**.)

By default the command does not move any atoms and just reports RMSD. The independent options ROTATE and TRANSLATE allow the option to do just rotation, do just center-of-mass translation, or do both. The basic command involves specifying at the very least two models. If no other parameters are included, the models are matched atom-for-atom based on "SPINE" atoms (or, with orientational comparisons, group by group). A subset of atoms for pairing other than SPINE, such as {\*.CA} can be specified using the SUBSET option. Finally, any number of atom sets to be correlated atoms can be given. The result of atom-atom pairing comparison is essentially the same as the PyMol **pair\_fit** command, though easier to implement and using an exact form of the structure-structure correlation rather than an iterative process.

Given here is a comparison that can be found at the Lilley group [Kink-Turn site](#) comparing the [Kt-38 site](#) to the [Kt-7 site](#).

- [load 3cc2\\_Kt7.pdb;color yellow:cartoons only # the kink-turn we will compare](#)
- [load 3cc2\\_900-1100.pdb;color orange:cartoons only:select 936-941 or 1025-1034;color red # A section of the 23S ribosome subunit containing a similar kink-turn, highlighted in red](#)
- [load 3cc2\\_900-1100.pdb;color orange:cartoons only:select 936-941 or 1025-1034;color red;load append 3cc2\\_Kt7.pdb:select 2.1;color yellow:cartoons only:frame \\*:zoomto 2 {\\*} 0 # the two together in their native orientations.](#)
- [moveto /\\* time,axisAngle \\*/ 1.0 { 386 700 -600 176.39} /\\* zoom, translation \\*/ 500.0 0.26 0.26 /\\* center, rotationRadius \\*/ {25.776789 71.75962 74.88684} 90.03123 /\\* navigation center, translation, depth \\*/ {0.0 0.0 0.0} 82.41045 -549.58514 0.0;depth 40;slab 60; slab on;](#)
- [compare {2.1} {1.1} SUBSET {spine} ATOMS {77-82} {936-941}, {92-93 or 97-100} {1025-1026 or 1031-1034} ROTATE TRANSLATE # best correlation of atoms](#)
- [set quaternionFrame "C"; compare {2.1} {1.1} SUBSET {spine} ORIENTATIONS {77-82} {936-941}, {92-93 or 97-100} {1025-1026 or 1031-1034} ROTATE TRANSLATE # best correlation of base orientations](#)

#### 102. SPINE predefined set

Jmol 11.9.34 adds SPINE to the list of predefined sets that can be used in Jmol. Spine includes just the atoms that form the continuous backbone set: \*.CA, \*.N, \*.C for proteins; \*.P, \*.O3\*, \*.O5\*, \*.C3\*, \*.C4\*, \*.C5 for nucleic acids. Phosphate oxygens OP1 and OP2 are not included but can be added using **spine** or **\*.OP1** or **\*.OP2**.

#### 101. set quaternionFrame "A", "C", and "P" for nucleic acids

quaternion frames are predefined orientations used by Jmol to compare relative or absolute orientations in space. Frame "A" was introduced in Jmol 11.8.1 and relates one alpha carbon to the next along a protein chain. In Jmol 11.9.34, this is extended to the phosphorus atoms along a nucleic acid chain. Specifically, this frame allows calculation and display of straightness and identification of secondary structure in models that only contain alpha carbons and phosphorus backbone atoms. Frame "C" is a frame that indicates the orientation of the sidechain of a protein in relation to the backbone. In Jmol 11.9.34 this is extended to purines and pyrimidine bases. Frame "P" for proteins indicates the absolute orientations of the peptide planes. Jmol 11.9.34 extends this to nucleic acids, indicating the absolute orientations of the phosphorus backbone tetrahedra.

#### 100. quaternion arrays, differences, means, and standard deviations

Quaternions can be used in Jmol to characterize the orientation of groups. In particular, Jmol can assign quaternion frames to amino acid and nucleic acid residues automatically. This suggests the possibility of manipulating arrays of quaternions and investigating their properties. Jmol 11.9.35 adds the capability to create arrays of quaternions, find the mean, and determine their standard deviation. In addition, Jmol 11.9.35 adds the capability to calculate the quaternion differences of an array of quaternions. Quaternion differences are interesting in the area of protein-protein alignment and in quantifying the regularity of structure in helices, strands, and other secondary structure motifs.

- [load 1cm.pdb:cartoons only;select within\(helix\):x = quaternion\({selected}\)](#)
- [print x # an array of quaternions](#)
- [print quaternion\(x\) # mean quaternion](#)
- [print x.average # mean quaternion](#)
- [print x.stddev # standard deviation](#)
- [dq = quaternion\({8-18}, {9-19}\);print "helicalAxis/angle = " + \(dq.average\)%6 + "nhelicalError = " + dq.stddev](#)
- [draw scale 10.0 VECTOR {8-18.CA} @/{\(dq.average\)%6-1}](#)

#### 99. webExport enhancements

Jmol 11.9.33 adds widgets in WebExport pages (background colorpicker, spin on/off, stereo mode)

#### 98. x\*\*y

Jmol 11.9.33 adds exponentiation to its math operators. 2\*\*3 = 8; -2\*\*2 = (-2)\*\*2

## 97. .x .y .z

Jmol 11.9.33 adds .x, .y, and .z as equivalents to .atomX, .atomY, and .atomZ.

## 96. translateSelected x/y/z

Jmol 11.9.32 adds the capability to translate a model programmatically a given number of pixels, nanometers, or Angstroms in the x, y, or z direction.

- [load caffeine.xyz](#)
- [translateSelected x 10](#)
- [translateSelected y 10](#) # note that on the screen a positive y direction is downward.
- [translateSelected x 2 Angstroms](#)
- [translateSelected Z 10](#)

## 95. connect XX% YY% ...

Jmol 11.9.32 adds the option to connect atoms based on ranges of percentage of bonding/ionic radii instead of fixed values.

## 94. lcaoCartoon CPK

Jmol 11.9.32 adds a new option to lcaoCartoon, CPK, which creates a sphere at the current spacefill radius. The reason this could be useful is that such spheres, though associated with an atom, can be slabbed and capped like an isosurface. This allows for a useful "unit cell only" rendering of spacefill models, for example.

- [load nacl.cif {1 1 1}; { Na }.formalCharge = 1; { Cl }.formalCharge = -1](#)
- [spacefill ionic](#)
- [lcaocartoon scale 1.0 CAP unitecell "cpk";spacefill off](#) # we turn the spacefill off -- it was just to provide the reference sizes

## 93. isosurface SLAB/CAP

Jmol 11.9.32 allows the clean slabbing of an isosurface based on a plane definition such as x=3 or {1 0 0 - 3} or the currently defined unitcell or boundingbox. A negative sign prior to a plane definition indicates "the opposite-facing plane". The distinction between SLAB and CAP is that SLAB leaves the isosurface open at the boundaries, whereas CAP closes it off.

- [load 1cm.pdb](#)
- [isosurface sasurface translucent](#)
- [isosurface SLAB x=3 sasurface fullylit](#)
- [isosurface SLAB - x=3 sasurface fullylit](#)
- [isosurface CAP xy sasurface](#)
- [boundingbox {0 0 0} {10 10 10};isosurface SLAB boundingbox sasurface](#)

## 92. specialized file data atom properties

Jmol 11.9.33 allows readers to create atom-based property data that is unique to their specific reader. For example, the Crystal09 reader can read atomic spin and magnetic moment properties.

- [load full\\_PBE40.out;reset:center {1.2758056 2.209848 -1.3375282E-4}; rotate z -114.4; rotate y 82.07; rotate z 95.48; translate x 0.13; translate y 0.13;](#)
- [select \\*, font labels 35](#)
- [select property\\_spin = 1;label "↑" # up arrow](#)
- [select property\\_spin = -1;label "↓" # down arrow](#)
- [color {\\*} property\\_spin](#)
- [display property\\_spin <= 0](#)
- [display property\\_spin >= 0](#)

## 91. 1D and 2D unit cells

Jmol 11.9.32 generalizes unit cells to 1D (polymer) and 2D (slab) periodicity. To date, the only reader that supports this sort of symmetry is the [Crystal09](#) solid state computation reader. When reading these files, unit cell designations {1 2 3} are adjusted to only involve the appropriate dimensionality.

- [load mgo\\_slab.out {3 3 1}](#) # note that the Z direction is Angstroms
- [load polymer.out {3 1 1}](#) # note that the Y and Z directions are Angstroms

## 90. CRYSTAL reader

Jmol 11.9.32 adds a reader for [Crystal09](#) solid state computation output files.

- [load mgo\\_slab.out](#)
- [load polymer.out](#)

## 89. isosurface DSN6/O reader

Jmol 11.9.31 adds a reader for DSN6/O map files as served by the [Uppsala Electron Density Server](#). These maps are alternatives to CCP4/MRC format.

## 88. frame TITLE enhancements

Jmol 11.9.31 allows setting the frame title similar to the way echos are set, using variables, and to set all frame titles at once. The default title is the model name.

- [load cyclohexane\\_movie.xyz](#)
- [frame \\*:frame title:frame 1](#)
- [animation on](#)

## 87. load "somefilename" 0

Jmol 11.9.30 adds the capability to load specifically only the LAST model in a file containing multiple models.

## 86. isosurface SIGMA

Jmol 11.9.30 adds the SIGMA option for automatically setting the isosurface cutoff to multiples of root mean square deviation (RMSD) as found specifically in CCP4/MRC electron density data files. Specifically, "sigma 2.0" delivers an electron density plot that is tighter around the nuclei. If, for example, a cutoff of 0.35 is associated with "sigma=1", then **sigma 2.0** sets the isosurface cutoff at 0.75.

## 85. set picking DELETEATOM

Jmol 11.9.28 adds the capability to simply click an atom to delete it.

## 84. set isKiosk

Jmol 11.9.28. This flag, when set TRUE, forces focus at all times and presumes no underlying applets. This allows multi-touch to activate immediately and not wait for a second click on the applet, using the first to establish focus.

## 83. \_multiTouchServer, \_multiTouchClient flags

Jmol 11.9.28 sets \_multiTouchServer and \_multiTouchClient to indicate whether it has been successful in connecting to a multi-touch device (and is thus working as a server) or to a server (and is thus working as a client).

## 82. FRANK OFF

Jmol 11.9.28 allows the local signed applet to turn off the frank. Especially useful for KIOSK applications in science exhibitions such as the recent installation of "Touch a Molecule" at the Epcot Theme Park.

#### 81. set mouseDragFactor and mouseWheelFactor

Introduced in Jmol 11.9.24, Jmol 12.0 will change the default sensitivity of the mouse wheel to work less aggressively. To use the older Jmol 11.8 sensitivity, use **set mouseWheelFactor 1.15**; the default for Jmol 12.0 is 1.02. Similarly, Jmol 12.0 will have a less aggressive drag sensitivity for the mouse, allowing the mouse to work more appropriately, especially in a touch-screen environment, (where the "mouse" is a finger). The default value of 1.0 for **set mousedragFactor** allows a 180-degrees rotation when the pointer drags across the full window width.

- [set mouseDragFactor 2.0](#) # more like 11.8 -- drag the model and observe
- [set mouseDragFactor 1.0](#) # Jmol 12.0 - note how the model tracks the mouse better
- [set mouseWheelFactor 1.15](#) # Jmol 11.8 - zooming super sensitive
- [set mouseWheelFactor 1.02](#) # Jmol 12.0 - zooming not so sensitive

#### 80. implicit SCRIPT command

Jmol 11.9.24 makes any command given of the type **xxxx.xxx** into an implicit **SCRIPT xxxx.xxx** command. This is just for convenience when working with the console, primarily.

#### 79. ionicRadius

Jmol 11.9.24 adds the ionicRadius property, and allows setting it.

- [load quartz.cif {1 1 1}](#).
- [{ O}.ionicRadius=1.2; { Si}.ionicRadius=0.5; spacefill ionic](#)

#### 78. new LOG command

Jmol 11.9.24 adds a new command specifically for the signed applet and the application. The LOG command works the same as **print** but records the information in a log file. If the printed data starts with the characters NOW, then those are replaced by the date and time. For example: **log "NOW" + getProperty("modelInfo")**. The file to log to must first be designated using **set logFile "someName"**. This name will be prepended with "JmolLog\_" and must not contain any directory path. The file will always be created in the Jar file directory. Note that logging is not ever possible with the web-based version, even with the signed applet, but signed applet or application running locally can log to a file. In addition to explicit use of the LOG command, two settings, **set logCommands** and **set logGestures** allow automatic tracking of commands and gestures (swipe, pinch, zoom, spin) to the designated log file.

#### 77. set waitForMoveTo FALSE

In Jmol versions prior to 11.9.24, all moveTo operations executed to completion prior to continuing a script. With **set waitForMoveTo FALSE**, one can allow moveTo operations to work asynchronously, meaning they have their own thread, and the script will continue immediately. To stop an asynchronous moveTo operation, use **moveTo STOP**. In addition, the mouse action DOUBLE-CLICK-LEFT is bound to "\_stopMotion".

#### 76. set allowMultiTouch

Jmol 11.9.24 allows turning off multi-touch gestures (two-finger spread for zoom, two-finger drag for translation).

#### 75. set pickingStyle DRAG

Jmol 11.9.24 adds the DRAG picking style option. **set pickingstyle DRAG** makes the LEFT button a click-and-drag button when associated also with **set PICKING select** (molecule, group, chain, etc.) and **set dragSelected**.

#### 74. print {\*}.polymer

Jmol 11.9.24. The sequential number of the polymer, starting with 1 for each model in the set.

#### 73. select POLYMER=n and select within(POLYMER, {someAtoms})

Jmol 11.9.24. Same as within GROUP, CHAIN, etc., but for polymers. The number n starts with 1 for each new model.

#### 72. set picking select STRUCTURE/POLYMER

Jmol 11.9.24 adds STRUCTURE (i.e. helix, strand or turn), and POLYMER to picking select option. POLYMER is distinct from CHAIN in that some chains contain more than one connected sets of groups, while a "polymer" is defined as a connected set of groups.

- [load 1crn.pdb;selectionHalos on;select none](#)
- [set picking select STRUCTURE](#) # now click on an atom atom
- [set picking select POLYMER](#) # now click on an atom atom

#### 71. new TIMEOUT command and show TIMEOUTS

Jmol 11.9.24 adds the option to set a script to execute at a future time, either as an isolated event or repetitively. The command takes the form **TIMEOUT name mSecDelay "script"**, where a name is given for future reference along with a delay in milliseconds and a script to run. If the millisecond delay is given as a negative number, then the event repeats every -(mSecDelay) milliseconds. The minimum practical value for the delay is about 100 ms. If the delay is 0 or the script is the empty string "", then any currently waiting timeout with that name is canceled. In addition, **timeout name OFF** cancels a timeout, and **timeout OFF** cancels all timeouts. All timeouts are also canceled when a new structure is loaded. The **show timeouts** command displays a list of all currently active timeouts.

- [timeout mytimeout 2000 "background white"](#)
- [timeout mytimeout -1000 "set echo top left;echo @ {now\(\)}"](#)
- [timeout mytimeout off](#)
- [timeout mytimeout -1000 "rotate x 30"](#)
- [show timeouts](#)
- [timeout OFF](#)

#### 70. set preserveState FALSE

This option, introduced in Jmol 11.9.23, turns off many memory-consuming features of Jmol that are necessary for preserving the state. It can be used in situations where memory is at a premium and there is no desire to write or save the current Jmol state.

#### 69. write MESH

Jmol 11.9.23 will generate a relatively compact JVXL XML "vertex-only" mesh surface file from an isosurface. A typical command, after creation of an isosurface, would be **write MESH t.jvxd**. Note that standard JVXL files are considerably smaller, however.

- [load caffeine.xyz](#)
- [isosurface molecular](#)
- [write MESH](#)

#### 68. simple PDB trajectories

Jmol 11.9.23 allows reading of PDB files that contain no MODEL line and are instead simply concatenated versions of the same atoms as trajectories using the TRAJECTORY keyword. Each model must start with atom number 1 for this to work.

- [load TRAJECTORY hb.pdb.gz;cartoons only;select not protein;wireframe 0.3](#)
- [animation on](#)

#### 67. set command autocompletion

Jmol 11.9.22 adds a new feature for the console. If in the middle of typing a set command one starts pressing the TAB key, Jmol cycles through all possible SET options starting with whatever characters have been typed so far.

#### 66. new STRUTS shape

In rapid prototyping of protein models, it is sometimes necessary to add short connectors between strands and helices to provide strength to the plastic model. Jmol 11.9.22 adds a new shape, STRUT, that creates these supports. Created especially for the folks at the [Center for Biomolecular Modeling](#) at the Milwaukee School of Engineering, these stick-like objects can be added using the **CONNECT... STRUTS** or **SET PICKING STRUTS** commands, and they can be calculated automatically using **calculate STRUTS**, which utilizes an algorithm contributed by [George Phillips](#) at the University of Wisconsin. STRUTS are not measures and they are not covalent bonds. When calculating struts, three parameters are used (defaults given): **set strutSpacing 6** sets the minimum spacing between struts, **set strutLengthMaximum 7.0** sets the maximum length that is allowed for a strut, and **set strutsMultiple FALSE** when set TRUE allows multiple struts on a given atom. In addition, **set strutDefaultRadius 0.3** sets the default radius for struts. (Their color by default is translucent white.)

- [load 1crn.pdb;wireframe only; wireframe 0.1](#)
- [calculate struts](#)
- [connect {atomindex=100} {atomindex=73} STRUT](#)
- [set picking STRUTS #now select two atoms](#)

#### 65. isosurface INLINE

Jmol 11.9.21 adds an INLINE option to the isosurface command, complementing the inline option for the PMESH command. Data would generally already be present in a variable. It is advisable to reset the variable after use to improve performance, however note that the state will only be preserved if the value of the variable is left unchanged.

- [load ch3cl.mol](#)
- [x = load\("ch3cl.jvxl"\)](#)
- [isosurface INLINE @x](#)
- [reset x](#)

#### 64. set picking CONNECT/DELETEBOND

Jmol 11.9.21 adds two new picking options. CONNECT performs like measuring distances, except bonds are created; DELETEBOND does the opposite -- deleting the bonds between selected atoms (and with **set BONDPICKING TRUE**, deletes bonds as they are clicked).

- [load caffeine.xyz](#)
- [set picking CONNECT # now click on atoms](#)
- [set picking DELETEBOND # now click on atoms](#)
- [set bondPicking true # now click on bonds](#)

#### 63. isosurface COLOR DENSITY

Jmol 11.9.21 adds a new option for isosurface allowing rendering of the actual grid of numbers (volume rendering) of the data rather than an actual isosurface. With CUTOFF 0.0, this setting delivers the entire set of data points.

- [load 3hyd.pdb;reset:center {16.626621 2.7783334 1.7281427}; rotate z 100.36; rotate y 112.6; rotate z -72.02; zoom 231.78; translate x 0.13; translate y 0.13;](#)
- [boundingbox scale 1.2 {tyr};isosurface boundingbox cutoff 1.6 "3hyd\\_map.ccp4.gz" mesh nofill](#)
- [boundingbox scale 1.2 {tyr};isosurface boundingbox cutoff 0.0 color DENSITY "3hyd\\_map.ccp4.gz"](#)
- [boundingbox scale 1.2 {tyr};isosurface boundingbox cutoff 1.6 color DENSITY "3hyd\\_map.ccp4.gz"](#)
- [set drawhover # now hover over a data point](#)

#### 62. color schemes BW and WB

Jmol 11.9.21 adds Black/White and White/Black (grey-scale) color schemes

- [load caffeine.xyz](#)

- [color property atomno](#)
- [color "BW"](#)
- [color property atomno](#)
- [color "WB"](#)
- [color property atomno](#)

#### 61. load "filename" AS "localFileName"

Jmol 11.9.20 (signed applet and application) allows reading and immediately saving a file to a local drive.

#### 60. extended XYZ file format

Jmol 11.9.20 adds a ninth column to XYZ files that allows for the atom number (as in PDB files) to be specified. Primarily this is for internal use.

#### 59. set slabByMolecule and slabByAtom

Jmol 11.9.19 adds the **slabByMolecule** and **slabByAtom** options. The first removes entire molecules when they are partially clipped by a slabbing plane; the second removes whole atoms and bonds only.

- [load Water!Liquid.xodydata;slab 60;slab on;zoomto 1.400](#)
- [set slabByMolecule !slabByMolecule](#)
- [set slabByMolecule false;set slabByAtom !slabByAtom](#)

#### 58. draw POLYGON

Jmol 11.9.19 allows the ability to draw polygons based on a set of vertices and a set of faces. This capability allows drawing any number of flat triangular (not quadrilateral) faces with or without edges around each face. The description is somewhat like that for PMESH files and involves (a) giving the number of vertices, (b) listing those vertices, (c) giving the number of faces, and (d) listing the faces with a special syntax. Each face is described as an array indicating the three (0-based) vertex indices followed by a number from 0 to 7 indicating which edges to show a border on when the **mesh** option is given: 0 (no edge), 1(v0-v1 edge), 2(v1-v2 edge), 4(v2-v3 edge), and then combinations of these to give combinations of edges.

- [load caffeine.xyz](#)
- [draw POLYGON 3 {0 0 0} {1 1 1} {1 2 1} 1 \[0 1 2 6\] # 6 here means "edges only between v1-v2 and v2-v3"](#)
- [draw POLYGON 3 {0 0 0} {1 1 1} {1 2 1} 1 \[0 1 2 6\] mesh nofill](#)
- [draw POLYGON 4 {0 0 0} {1 1 1} {1 2 1} {0 5 0} 2 \[0 1 2 6\] \[0 3 2 6\] mesh nofill](#)

#### 57. calculate/delete HYDROGENS and minimize ADDHYDROGENS

Jmol 11.9.19 adds the ability to add and delete hydrogens, and to add hydrogens and minimize the structure in one operation.

- [load thyroxinenoh.mol;rotate x 30;rotate y 30](#)
- [calculate HYDROGENS](#)
- [delete HYDROGENS](#)
- [minimize ADDHYDROGENS](#)

#### 56. isosurface WITHIN x.x {points}

Jmol 11.9.19 adds a WITHIN option to produce the surface within a given distance of ANY atom in the set, not just the center. (isosurface WITHIN was introduced in Jmol 11.1.21 but not documented).

- [load caffeine.xyz;rotate x 30;rotate y 30](#)
- [isosurface WITHIN 2.0 {O11} sasurface 0](#)

#### 55. boundingbox SCALE x.x option

Jmol 11.9.19 adds a SCALE option for boundingbox, allowing scaling with a variety of definition options.

- [load caffeine.xyz:boundbox on:rotate x 30:rotate y 30](#)
- [boundbox scale 0.5](#)
- [boundbox scale 1.1 { O }](#)
- [boundbox scale 1.1 CORNERS {0 0 0} {3 3 3}](#)
- [boundbox scale 1.1 {-3 3 3} {3 3 3}](#)
- [isosurface sa1 select\(O11\) sasurface 0:boundbox scale 1.1 \\$sa1](#)

#### 54. boundbox Sisosurface1

Jmol 11.9.19 allows the boundbox to be display around an isosurface.

#### 53. XPLOr electron density reader

Jmol 11.9.19 adds an [XPLOr](#) electron density reader.

#### 52. set zshadePower

Jmol 11.9.18 allows for an exponential sort of fog shading. The parameter is log<sub>2</sub>(p) where the opacity function  $f = [(zDepth - z) / (zDepth - zSlab)]^p$ . This provides a more dramatic effect of depth.

- [set background white:load WaterLiquid.xodydata:reset:center {6.5661736 6.6120844 6.639286}; rotate z -125.08; rotate y 55.12; rotate z 120.68; zoom 400.0; translate x 0.13; translate y 0.13;spin on](#)
- [set zshade on](#)
- [set zshadePower 2](#)
- [set zshadePower 1 # default](#)

#### 51. VASP vasprun.xml

Jmol 11.9.18 adds a [Vienna Ab Initio Simulation Package](#) reader for vasprun.xml files.

- [load vasprun.xml](#)

#### 50. MEASURE function

Jmol 11.9.18 adds the MEASURE() function, allowing programmatic return of measurement information. The syntax is: measure({exp1},{exp2},{exp3},{exp4}), min, max, format, units, "CONNECTED"). That is, two to four expressions, minimum and maximum range in Angstroms, a format expression of the sort "%a1 %a2 %5.3VALUE %UNITS", a specification of desired units, including "pm", "nm", "au", or "angstroms", and the keyword "CONNECTED" if the atoms must be connected.

- [load caffeine.xyz](#)
- [print measure\({carbon},{oxygen}, 1.2, 1.5, "%a1 %a2 %3.0VALUE %UNITS", "pm", "CONNECTED"\)](#)

#### 49. select configuration=1

Jmol 11.9.18 adds CONFIGURATION to select options. When a model has two alternative locations for some atoms, one can now simply select (or display, hide, color, etc.) the set of atoms that have no alternative location and the set of the nth alternative location using **select configuration=n**.

- [load 1VIF.cif:reset:center {44.914 36.531498 39.387}; rotate z -65.12; rotate y 83.6; rotate z 97.82; translate x 0.13; translate y 0.13;restrict protein:cartoons only;select not protein; wireframe 0.3 # two inhibitor alignments](#)
- [display configuration=1](#)
- [display configuration=2](#)

#### 48. hkl(a,b,c)

Jmol 11.9.17 adds the function hkl, which generates the plane associated with a given set of Miller plane indices.

- [load quartz.cif {1 1 1}](#)
- [print hkl\(1,1,1\)](#)

#### 47. draw/show SYMOP

Jmol 11.9.17 introduces the ability to describe and illustrate symmetry operations. In addition, you can pick any two atoms or groups or any two positions in space and see the symmetry relations between those groups.

- [load maleic.cif 3:moveto /\\* time\\_axisAngle \\*/ 1.0 { 539 -803 -254 57.63} /\\* zoom\\_translation \\*/ 100.0 0.13 0.13 /\\* center\\_rotationRadius \\*/ {0.62577057 3.7338495 4.0119743} 9.027727 /\\* navigation center\\_translation\\_depth \\*/ {0.0 0.0 0.0} 13.809929 7.7656565 0.0;](#)
- [show symop](#)
- [show symop 2](#)
- [draw delete:draw symop 2](#)
- [select \\*.wireframe only;color translucent:draw symop {molecule=1} {molecule=2}; select molecule=1 or molecule=2;Halos 0.1;show symop {molecule=1} {molecule=2};](#)
- [select \\*.wireframe only;color translucent:draw symop {molecule=1} {molecule=3}; select molecule=1 or molecule=3;Halos 0.1;show symop {molecule=1} {molecule=3};](#)
- [select \\*.wireframe only;color translucent:draw symop {molecule=1} {molecule=4}; select molecule=1 or molecule=4;Halos 0.1;show symop {molecule=1} {molecule=4};](#)

#### 46. draw planes intersecting planes with unit cells and bounding boxes

Jmol 11.9.17 expands the "intersection" idea to allow the drawing of planes within boundaries.

- [load quartz.cif packed:moveto /\\* time\\_axisAngle \\*/ 1.0 { -996 -73 -60 78.27} /\\* zoom\\_translation \\*/ 100.0 0.0 0.0 /\\* center\\_rotationRadius \\*/ {1.2289999 2.1286902 2.7027001} 5.8060317 /\\* navigation center\\_translation\\_depth \\*/ {0.0 0.0 0.0} -12.054248 29.988888 0.0;](#)
- [draw intersection unitcell hkl {0 2 0};](#)
- [draw intersection unitcell plane x=3](#)
- [draw intersection unitcell plane x=@{{1/2 0 0}.x};](#)

#### 45. draw BOUNDBOX; draw UNITCELL

Jmol 11.9.17 adds the capability to draw a bound box (which can be defined) and the unit cell and to do so with scaling.

- [load 1crn.pdb;](#)
- [draw boundbox mesh nofill](#)
- [select structureID=H2;color group:display selected:boundbox {selected};draw box1 boundbox color white mesh nofill:zoomTo {selected} 0;](#)
- [select structureID=H1;color group:display selected:boundbox {selected};draw box2 boundbox color yellow mesh nofill:zoomTo {selected} 0;](#)
- [load caffeine.xyz:draw boundbox backlit:draw b2 boundbox translucent 0.7;rotate x 30;spin on # possibly interesting... or perhaps just too weird.](#)
- [spin off](#)

#### 44. new Van der Waals default 23%AUTO

Jmol 11.9.17 changes the default Van der Waals radius to "AUTO" to allow non-PDB files and PDB files with H atoms to load with a slightly different look than PDB files with no H atoms. This brings Jmol's default parameter set in line with OpenBabel 2.2.] Along with this change there is a simple syntax for specifying SPACEFILL, HALO, and STAR size.

- [load caffeine.xyz](#)
- [spacefill JMOL # the old look -- bloated carbons for caffeine. Jmol 11.8 default -- proposed 12.0 default for 1CRN](#)
- [spacefill AUTO # the new look -- proposed 12.0 default for all models](#)
- [spacefill BABEL # same as AUTO for caffeine -- new for Jmol; proposed 12.0 default for caffeine and 1CDS](#)
- [spacefill 20%JMOL # the OLD default ball and stick look](#)
- [spacefill 23%AUTO # the NEW default ball and stick look](#)
- [load 1crn.pdb # now try the above links -- this time JMOL and AUTO will be the same](#)
- [load 1cdr.pdb # this PDB file with H atoms uses the more appropriate BABEL set of parameters automatically](#)

#### 43. "mountain" plots from plane mappings

Jmol 11.9.17 adds the SCALE3D option to isosurface. This generates a 3D plot of the desired scale from a mapped plane. It can be introduced either with the original definition of the isosurface or later.

- [load C6H6.smol;mo homo;reset:center {0.0 0.0 0.0}; rotate z -129.9; rotate y 63.91; rotate z 95.81; translate x 0.13; translate y 0.13;](#)
- [isosurface plane z=0.5 MO homo](#)
- [isosurface scale3d 5 offset {0 0 2}](#)
- [isosurface scale3d 2](#)
- [isosurface mesh nofill](#)

#### 42. write PMESH

Jmol 11.9.14 adds the PMESH option to the write command, creating XJVXL files.

- [zap;pmesh "pmesh.bin"](#)
- [write PMESH](#)

#### 41. getProperty SHAPEINFO

Jmol 11.9.14 expands the getProperty command to include information about an isosurface extent

- [load caffeine.xyz;isosurface molecular translucent](#)
- [getProperty "shapeInfo.isosurface\[1\].xyzMin"](#)
- [getProperty "shapeInfo.isosurface\[1\].xyzMax"](#)
- [isosurface delete;pmesh "pmesh.bin"](#)
- [getProperty "shapeInfo.pmesh\[1\].xyzMin"](#)
- [getProperty "shapeInfo.pmesh\[1\].xyzMax"](#)

#### 40. DGRID reader

Jmol 11.9.14 adds a [DGRID](#) reader. These files are generalized representations of output from a variety of quantum mechanical calculation packages, including especially [ADE](#).

- [load H2O.adf.dgrid.frame last;MO lumo](#)

#### 39. isosurface and pmesh bounding box information

Jmol 11.9.14 allows access to the bounding box that contains a graph.

- [zap;isosurface ID "s1" FILE "func.jvxl"](#)
- [print getProperty\("shapeInfo.isosurface\[1\].xyzMin"\)](#)
- [print getProperty\("shapeInfo.isosurface\[1\].xyzMax"\)](#)
- [zap;pmesh "pmesh.bin"](#)
- [print getProperty\("shapeInfo.pmesh\[1\].xyzMin"\)](#)
- [print getProperty\("shapeInfo.pmesh\[1\].xyzMax"\)](#)

#### 38. surface scaling and repositioning

Jmol 11.9.13/14 adds the ANISOTROPY option for all PMESH and ISOSURFACE objects. Just add a point or array after the ANISOTROPY keyword to indicate the scaling in the X, Y, and Z directions. An optional CENTER can also be defined rather than the standard {0 0 0}. The OFFSET option allows an after-the-fact repositioning capability.

- [zap;Pmesh "pmesh.bin";axes on;axes molecular](#)
- [Pmesh anisotropy {0.1 1 1} "pmesh.bin"](#)
- [Pmesh anisotropy {1 2 1} "pmesh.bin"](#)
- [Pmesh anisotropy {1 2 1} center {-1 -1 -1} "pmesh.bin"](#)
- [Pmesh anisotropy {1 2 1} offset {2 2 2} "pmesh.bin"](#)
- [pmesh offset {1 1 1}](#)
- [pmesh offset {1 -1 1}](#)

- [load C6H6.smol;moveto /\\* time, axisAngle \\*/ 1.0 {-999 34 -8 65.28} /\\* zoom, translation \\*/ 21.85 0.13 0.13 /\\* center, rotationRadius \\*/ {0.0 0.0 0.0} 3.657167 /\\* navigation center, translation, depth \\*/ {0.0 0.0 0.0} 0.0 0.0 0.0;](#)
- [isosurface "func.jvxl" color translucent](#)
- [isosurface anisotropy {0.5 0.5 0.5} offset {-5 -5 -5} "func.jvxl" color translucent fullylit](#)
- [for \(var i = 0; i < 20; i++\) {isosurface offset @ {point\(-5, -5, -i/4.0\)};delay 0.05 }](#)

#### 37. set phongExponent

Jmol 11.9.13 adds the **phongExponent** parameter. This is a standard parameter relating to specular reflection (the bright dot on a ball) and is related to Jmol's **specularExponent** as 2^(specularExponent)

#### 36. set saveProteinStructureState

Jmol 11.9.13 adds the option to not save helix/sheet/turn data to the state. Generally this information is unnecessary, and in certain situations it can be a large amount of information. The flag is generally set TRUE by default.

#### 35. axes labels

Jmol 11.9.12 allows specification of axis labels. Either three or six labels can be indicated. If only three axes labels are given, the axes will start at 0.

- [axes molecular; axes on;](#)
- [axes labels "a" "b" "c"](#)
- [axes labels "a" "b" "c" "-a" "-b" "-c"](#)

#### 34. scales for axes, boundingbox, measures, and unitcells

Jmol 11.9.12 adds optional scales to axes, boundingbox, unit cells, and measures. There are three levels of ticks - major, minor, and "subminor." Only the major ticks have labels. Which of these tick levels are displayed and the distance between ticks depends upon the parameter that looks like a point given after the keyword TICKS in the AXES, BOUNDBOX, UNITCELL, or MEASURE command. For a specific axis, include "x" "y" or "z" just after TICKS. The additional optional keyword FORMAT defines the format of the labels for the major ticks. These are based on an array of strings given after the FORMAT keyword. If the array is shorter than the number of ticks, the formats in the array are repeated. The label numbers and tick separations do not have to be angstroms. For the UNITCELL command, simply use fractional coordinates for the ticks to provide a scale in unit cell coordinates. For the other commands, scaling is accomplished using the SCALE keyword followed by a general scaling number or a point indicating how much scaling to apply in each of the directions x, y, and z. For MEASURE you can also add a FIRST keyword followed by a starting value.

- [load caffeine.xyz](#)
- [boundingbox ticks {2.0 1.0 0.2} # major, minor, subminor](#)
- [measure ticks {1.0 0.2 0} format \["%0.0f"\] {-3 0 0} {3 0 0}](#)
- [measure ticks {1.0 0.2 0} format \["%0.0f"\] first -1.3 {-3 0 0} {3 0 0}](#)
- [boundingbox off;measures delete;](#)
- [axes ticks {1 0 0} format \["%0.0f"\]](#)
- [axes ticks none](#)
- [axes ticks x {2 0 0} format \["%0.0f"\] scale 2](#)
- [axes ticks y {1 0 0} format \["%0.0f"\] scale 1.5](#)
- [axes ticks z {3 0 0} format \["%0.0f"\] scale 3](#)
- [axes ticks none; axes ticks {1 0 0} format \["%0.0f"\] scale {1 2 3}](#)
- [axes ticks none; axes ticks x @ {point\(3.14159/2.0.0\)} format \["π/2"\] "π" # Note that the origin for axes and unit cells is not given a tick or a label. The π character is \u03C0.](#)
- [load quartz.cif packed;rotate x -60;axes off;unitcell ticks {1/2 1/10 0}](#)

#### 33. new multi-touch interface

Jmol 11.9.11 introduces a modified [Sparsh-UI](#) gesture server interface that allows Jmol to talk to a special multi-touch driver (our first C++ component of Jmol). The interface so far allows multi-touch action only

an HP TouchSmart computer (or any computer with [NextWindow](#) technology). The stand-alone Jmol application, Jmol embedded in other Java programs, and the signed applet are supported. One must first start [JmolMultiTouchDriver.exe](#). After this small C++ program is running, all messages from the touch-screen are passed to Jmol and interpreted as "gestures." The signed applet must be started with the "multitouchSpashUI" parameter set to "true". The application must be started with the -Mspashui flag, and embedded Jmol must be passed the command option -multitouch-sparshui. Gestures include a standard two-finger spread/pinch for zoom in/out, a two-finger slide for translation, and a one-finger "flick" to start spinning.

### 32. zoom in/out; zoomTo in/out

Jmol 11.9.11 adds **in** and **out** as parameters to the **zoom** and **zoomTo** commands. These parameters are synonymous with "\*"2" and "/"2", respectively.

- [zoom in](#)
- [zoom out](#)
- [zoom \\*2](#) #same as zoom IN
- [zoomTo in](#)
- [zoomTo out](#)
- [zoomTo 3 {1} in](#)
- [zoomTo 1 {30} out](#)

### 31. matrix math

Jmol 11.9.10 allows a broader range of matrix math, including two new functions, `m.col(n)`, and `m.row(n)`.

- [q = axisangle\({1 0 0},30\);m = q%-9;print q;show @m](#)
- [print m \\* {1 0 0}](#)
- [print {1 0 0} \\* m](#)
- [show @m](#)
- [show @{-m}](#) # transpose
- [show @{!m}](#) # inverse
- [show @{m.col\(1\)}](#)
- [show @{m.row\(2\)}](#)
- [show @{m.col\(3\)}](#)

### 30. draw lineData

Since Jmol 11.7.1 users have been able to draw arbitrary lines using **draw LINE** followed by a set of points. Starting with Jmol 11.9.10, you can draw line segments based on a set of start/stop pairs. The command was designed to allow for saving the state of the **draw INTERSECTION** command, but may have other practical uses.

- [load caffeine.xyz;](#)
- [draw lineData \[{0 0 0} {1 1 1}, {0 0 0} {1 1 -1}, {0 0 0} {1 -1 1}\]](#)

### 29. draw INTERSECTION \$myIsosurfaceID PLANE[HKL [plane definition]

Jmol 11.9.10 allows drawing of the intersecion of an isosurface with a plane.

- [set antialiasdisplay;load caffeine.xyz;isosurface s1 molecular;](#)
- [draw intersection \\$s1 plane x=0 color red](#)
- [draw intersection \\$s1 plane x=1 color green](#)
- [draw intersection \\$s1 plane x=2 color blue](#)
- [moveto /\\* time,axisAngle \\*/ 1.0 {-962 -199 188 105.82} /\\* zoom, translation \\*/ 100.0 0.0 0.0 /\\* center, rotationRadius \\*/ {-0.23140144 0.61279976 0.071704745} 5.580881 /\\* navigation center, translation, depth \\*/ {0.0 0.0 0.0} 1.814305 -1.5505937 0.0;isosurface off;for \(var i = -3; i < 3; i+=0.1\){draw intersection \\$s1 plane x = @i;delay 0.1}](#)
- [draw diameter 0.02 intersection \\$s1 plane z = 0 color blue](#)
- [for \(var i = -3; i < 3; i+=0.1\){draw ID @{"x"+i} intersection \\$s1 plane x = @i color red;delay 0.1}](#)

### 28. isosurface color MESH [color]

Jmol 11.9.10 adds the option to color the mesh aspect of an isosurface differently from the isosurface itself. Just indicate the color of the mesh prior to defining the object itself.

- [load caffeine.xyz;isosurface color mesh red molecular mesh fill](#)

### 27. new mouse action: LEFT-DOUBLE-CLICK-DRAG

Jmol 11.9.10 adds a new mouse action. Pressing the mouse twice and dragging translates the model.

### 26. new commands BIND and UNBIND

Jmol 11.9.9 introduces two new commands, BIND and UNBIND. These commands allow customized connections between user mouse actions and Jmol actions. In addition, you can now tie a mouse action to a script of your choice, and Jmol will insert into that script the variable names `_X`, `_Y`, `_DELTA_X`, `_DELTA_Y`, `_TIME`, and `_MODE`. Current Jmol actions that can be bound to mouse actions include: `_clickFrank`, `_depth`, `_dragDrawObject`, `_dragDrawPoint`, `_dragLabel`, `_dragSelected`, `_navTranslate`, `_pickAtom`, `_pickIsosurface`, `_pickLabel`, `_pickMeasure`, `_pickNavigate`, `_pickPoint`, `_popupMenu`, `_reset`, `_rotate`, `_rotateSelected`, `_rotateZ`, `_rotateZorZoom`, `_select`, `_selectAndNot`, `_selectNone`, `_selectOr`, `_selectToggle`, `_selectToggleOr`, `_setMeasure`, `_slab`, `_slabAndDepth`, `_slideZoom`, `_spinDrawObjectCCW`, `_spinDrawObjectCW`, `_swipe`, `_translate`, and `_wheelZoom`

- [unbind "CTRL-ALT-LEFT";bind "CTRL-ALT-LEFT" " \\_rotate"](#) # `_rotate` is a special keyword now press CTRL-ALT-LEFT and drag the model -- same as just LEFT alone
- [unbind "CTRL-ALT-LEFT"](#) #remove that binding from all Jmol actions
- [unbind " \\_popupMenu"](#); [unbind " \\_clickFrank"](#) # now try to pull up the popup menu
- [rotate x @ {random\(\)}\\*90;rotate x @ {random\(\)}\\*90;bind "ALT-SHIFT-LEFT" "boundbox on;moveto LEFT;boundbox off"](#) # press ALT-SHIFT-LEFT
- [set echo bottom left;bind "ALT-LEFT" "echo \\_X \\_Y \\_MODE"](#) # now press ALT-LEFT and drag the cursor around the screen. Note, this could be a function call, such as `myfunc( _X, _Y)"`
- [unbind](#) # resets the binding to Jmol defaults

### 25. getProperty mouseInfo

Jmol 11.9.9 adds the `MOUSEINFO` property to `getProperty`. This information includes detailed information about how ALL mouse actions correlate with Jmol actions.

- [getProperty mouseInfo](#)
- [getProperty mouseInfo.bindingName](#)
- [getProperty mouseInfo.bindings](#)
- [print getProperty\("mouseInfo.actionNames"\)](#)
- [print getProperty\("mouseInfo.actionInfo"\)\[10\]](#)

### 24. show mouse [option]

Jmol 11.9.9 introduces a command that lists the current Jmol action/mouse button bindings. The **option** provides a quick way to search for any word in the command description.

- [show mouse](#)
- [show mouse pick](#)
- [show mouse move](#)
- [show mouse select](#)

### 23. right-edge zoom

In Jmol 11.9.9, a vertical motion in the right 2% of the window interpreted as a zoom motion (as in a touchpad).

### 22. set allowGestures

Jmol 11.9.9 represents a major upgrade in the way mouse events are handled internally. Events are "bound" to Jmol "actions" in a dynamic way. This allows for future expansion to include customized mouse action. A Java programmer using Jmol can now completely rewrite Jmol's mouse button mapping in just a

few minutes. This upgrade also allows for "gestures" -- dragging motions that are bound to specific actions based on their temporal or spacial pattern. One gesture is included in Jmol 11.9.9: a swipe of the screen that starts the model spinning on an axis perpendicular to the swipe and parallel to the screen. Additional multi-touch gestures can now also be bound to specific actions. For example an iPod-like "pinch" could indicate "zoom out", or a two-finger swipe could mean "advance animation". The gesture capability is turned on using **set allowGestures TRUE**. It is turned off by default to be compatible with previous versions of Jmol.

- [set allowGestures](#) # now swipe the screen with the mouse (that is, press the LEFT button and move the mouse, but let go of the LEFT button during the motion). Clicking anywhere on the screen stops this motion.

## 21. label HIDE and DISPLAY

Jmol 11.9.8 adds the capability to temporarily hide or display selected labels. Select the desired atoms, then issue the command **label HIDE** or **label DISPLAY**. Unlike label ON and label OFF, these options do not reset the label to the default label. Jmol 11.9.8 also changes the action of **set toggleLabel** to toggle labels hidden or displayed without resetting them.

- [load caffeine.xyz; label Atom %e-%i](#)
- [select carbon:labels HIDE](#)
- [select carbon:labels DISPLAY](#)
- [select oxygen;set toggleLabel #click me multiple times](#)
- [for \(var i = 0; i < 10; i++\) {set toggleLabel; delay 0.2}](#)
- [label "ON" # note that you can use quotes now to force the label even if it is a keyword](#)

## 20. load single vibration

Jmol 11.9.8 allows loading of a single vibration by number, starting with 1. Simply include the negative of the vibration number after the file name. (A positive number here is a model number.)

- [load "C6H6.smol" -1; rotate x -70; vibration on](#)
- [load "C6H6.smol" -3; rotate x -70; vibration on](#)

## 19. JVXL XML format

The JVXL format allows compression factors in the range 10-300 for file delivery of isosurface data. Jmol 11.9.7 is an initial attempt to define an XML standard for JVXL files. To date, the [JVXL file format](#) in its original form has been only marginally extensible -- but has been extended by use of XML-like fields. This version creates a simple XML format that allows for a wide variety of future implementation, including additional methods of ASCII-encoded compression, discretely contour-mapped plane and other surface data, pmesh triangle/vertex data storage capability, and generalized contour containers. The default writing format is now "XJVXL" for these extended types, and in this version use of ".xjvxl" in the file name of the file enables writing of all isosurface types as XML files.

## 18. now()

Jmol 11.9.7 adds the now(i) function to show milliseconds of time since an event. Just assign now() to a variable such as x, and then later check now(x).

- [print now\(\)](#)
- [x=now\(\); delay 3; print "the number of milliseconds was " + now\(x\)](#)

## 17. isosurface ... map CONTOUR DISCRETE and INCREMENT

Jmol 11.9.7 allows creation of specific contour levels for an isosurface, and also changes the visualization of the solid sections between the contours to be solid colors (FILL attribute of the isosurface command). The DISCRETE keyword allows any values; the INCREMENT keyword allows creation of a set of contours based on the set {from, to, step}.

- [load C6H6.smol; isosurface plane {0 0 1 0} map contour 20 mep](#)
- [isosurface plane {0 0 1 0} map contour DISCRETE \[-0.05, -0.04, -0.03, -0.02, -0.01, -0.0, 0.01, 0.02, 0.03, 0.04 0.05\] mep](#)
- [isosurface FILL # contour lines black](#)

- [isosurface plane {0 0 1 0} map contour INCREMENT {-0.05, 0.05, 0.004} mep](#)
- [isosurface FILL NOMESH # contour lines removed](#)

## 16. zshade ("fog")

Jmol 11.9.7 perfects the effect of fog -- a blending of distant areas of the model into the background. To turn on this effect, issue **set zshade**. The standard SLAB and DEPTH values (default 100 and 0, respectively) determine both the clipping plane and the points where the structure appears clearest (slab value) and where it blends into the background (depth value). Note that issuing **slab on** is no longer necessary to enable zShading.

- [load 1crn.pdb; cartoons on; color group; set zshade; spin on](#)
- [background white; zoomto 10 \\*4; delay 10; spin off](#)
- [slab 100; set zshade on; for \(var i = 0; i < 100; i++\) {delay 0.01; depth @i}](#)
- [slab 100; set zshade on; for \(var i = 100; i >= -300; i--\) {delay 0.01; depth @i};](#)

## 15. getproperty shapeInfo.isosurface

Jmol 11.9.7 allows access to the colors and contour values used in contour mapping.

- [load nacl.cif {1 1 1}; isosurface hkl {1 1 1} map contour 6 molecular color "red green blue"](#)
- [getproperty shapeInfo.isosurface](#)
- [print getproperty\("shapeInfo.isosurface\[1\].contours.colors"\)](#)
- [print getproperty\("shapeInfo.isosurface\[1\].contours.values"\)](#)

## 14. isosurface ... COLOR "color1 color2 color3..."

Jmol 11.9.7 adds the capability to specify contour colors directly, without a color scheme. Simply place a set of color names in quotes after the COLOR keyword for the isosurface. If fewer colors are given than contours, the color sequence is repeated as many times as necessary. The FILL keyword fills the spaces between the contour levels with solid colors, as for discrete contours.

- [load nacl.cif {1 1 1}; isosurface hkl {1 1 1} map contour 6 molecular color "red green blue"](#)

## 13. isosurface contour DISCRETE and contour INCREMENT

Jmol 11.9.7 adds two new options for isosurface plane mappings that allow for a more GNUplot-like contouring. The first option, CONTOUR DISCRETE, accepts a set of contour level values in the form of an array. The second, CONTOUR INCREMENT, takes three numbers -- first, last, and step -- in the form of a point {from, to, step}. If the isosurface is given the FILL attribute, the lines are drawn in black, and regions between the contours are given solid colors (that is, with no color blending).

- [load caffeine.xyz; isosurface iz plane z=0 contour DISCRETE \[-2.0, -1.8, -1.6, -1.5, -1.0, -0.5, -0.1\] molecular](#)
- [load C6H6.smol; wireframe 0.05; spacefill off; isosurface plane z=0 color absolute -0.2 0.2 contour increment {-0.2 0.2 0.01} MO 15](#)
- [load C6H6.smol; isosurface molecular contour increment {-0.2 0.2 0.01} MO 15](#)
- [load 1crn.pdb; isosurface sasurface 0 map contour increment {4 30 2} property temperature](#)

## 12. Shape size testing and setting

Jmol 11.9.5 adds the capability to test for sizes of several shapes, including backbone, cartoon, dots, ellipsoid, geosurface, halo, meshRibbon, ribbon, rockets, spacefill, star, strands, and trace. Settable shape sizes include all of the above except dots, ellipsoid, and geosurface.

- [load 1crn.pdb; color {34} yellow; cartoons on](#)
- [print {34} cartoon](#)
- [if \({34} cartoon > 0.5\) print "yes" else print "no" endif](#)
- [{34} cartoon \\*=0.75](#)
- [load caffeine.xyz; dots on](#)
- [print { O} dots](#)
- [print { C} dots](#)

## 11. Jmol Archive Format (.jmol) -- in review

Starting with Jmol 11.9.4, you can use **write "myfile.jmol"** to save a single ZIP-format file that contains all files necessary to load the model state. Drag/dropping this file into Jmol or using **load "myfile.jmol"** recreates the state. The file contains (1) a file JmolManifest.txt, which indicates the date of creation, Jmol version used, and the file to open, (2) a JPG image with embedded script, (3) an SPT state file, and (4) all additional files required (PDB files, Jvxl files, etc.). If a different file extension is desired, the keyword ZIPALL can be used prior to the file name: **write ZIPALL "myfile.zip"**. In addition, rather than using ZIPALL, if the keyword is "ZIP", then only files local to the file system are saved -- any remote file references starting with http:, https:, or ftp: are left as such, and those files will be accessed remotely when Jmol opens the archive. An example of a Jmol Archive Format file is [data/test1.jmol](#). It can also be tested using the [Jmol Crystal Structure Explorer](#), where it is invoked by a JmolButton (lower right panel). The JmolManifest method was introduced in Jmol 11.4; it can be used with this format to load specific files in specific ways. This format is under review, and comments are appreciated.

- [load test1.jmol](#)
- [load test1.jmol MANIFEST "nacl.cif"](#)
- [load test1.jmol MANIFEST "nacl.cif" {3 3 3}](#)

## 10. \$SCRIPT\_PATHS

Jmol 11.9.4 can insert into a script file the path to that file. The special string \$SCRIPT\_PATHS will be replaced. The path will either end with "/" or "\", depending upon whether it is into a subdirectory or a ZIP file. When the replacement is done, "\$SCRIPT\_PATHS/" will first be replaced by the path, then \$SCRIPT\_PATHS alone will be replaced. See, for example, [data/test1.txt](#).

- [script test1.txt;print path](#)
- [print load\(path + "1crn.pdb"\).lines.find\("COMPND"\)](#)

## 9. symop(n) \* symop(m)

Perhaps one of the most difficult aspects of symmetry is seeing that the product of two symmetry operations is still a symmetry operation. And which one? Jmol 11.9.3 allows multiplication of symmetry operations, because they are simply matrices. Combined with the capabilities of the symop() function to calculate symmetry operations from matrices, you have a very powerful symmetry operation "calculator".

- [load quartz.cif packed](#)
- [draw symop 3](#)
- [draw symop @ {!symop\(3\)}](#)
- [draw symop @ {symop\(3\) \\* symop\(3\)}](#)
- [draw symop 4](#)
- [draw symop @ {symop\(4\) \\* symop\(3\)}](#)
- [print "atom 1 is at \n" + {atomindex=1}.xyz](#)
- [print "its related atom is at:\n" + symop\(3,{atomindex=1}\)](#)
- [y = symop\(3,{atomindex=1}\)](#)
- [print "and hopefully we will get atom 1 back from this:\n" + symop\(-3, y\)](#)

## 8. symop(n) and matrix math

Jmol 11.9.3 adds matrix mathematics and, in particular, representation of symmetry operations as matrices. The syntax is the same as for JavaScript:  $[[a, b, c], [d, e, f], [g, h, i]]$  (3x3 matrix) or  $[[a, b, c, x], [d, e, f, y], [g, h, i, z], [j, k, l, w]]$  for a 4x4 matrix.

- [load nacl.cif packed;x = symop\(33\)](#)
- [print x](#)
- [print x.lines](#)
- [print \(x\).lines](#)
- [print x\[1\] # row 1](#)
- [print x\[-1\] # column 1](#)
- [print x\[13\] # row 1, col 3](#)
- [print x\[1\]\[3\] # same thing](#)
- [x\[13\]= 999; print x.lines](#)
- [x\[1\]=\[1,2,3,4\]; print x.lines](#)

- [x\[-1\]=\[1,2,3,4\]; print x.lines](#)
- [print symop\(192,"description"\)](#)
- [print symop\(192\).lines](#)
- [print symop\(-192\).lines](#)
- [print \(!symop\(192\)\).lines # same thing](#)
- [print symop\(symop\(192\)\\*symop\(192\)\).lines](#)
- [print symop\(symop\(192\)\\*symop\(192\)\\*symop\(192\)\).lines](#)
- [print symop\(symop\(192\),"description"\)](#)
- [print symop\(symop\(192\)\\*symop\(192\),"description"\)](#)
- [print symop\(symop\(192\)\\*symop\(192\)\\*symop\(192\),"description"\)](#)

## 7. print symop(n, "...")

Jmol 11.9.3 allows extracting of symmetry operation information directly from the symop() function. The information is in the form of an array, as described below in context.

- [load nacl.cif packed;set echo top right](#)
- [n = 3;draw symop @n:echo @ {"" + n + "- " + symop\(n,"description"\)}](#)
- [n = 6;draw symop @n:echo @ {"" + n + "- " + symop\(n,"description"\)}](#)
- [n = 34;draw symop @n:echo @ {"" + n + "- " + symop\(n,"description"\)}](#)
- [n = 106;draw symop @n:echo @ {"" + n + "- " + symop\(n,"description"\)}](#)
- [n = 192;draw symop @n:echo @ {"" + n + "- " + symop\(n,"description"\)}](#)
- [print symop\(n\) # by itself, its matrix representation](#)
- [print symop\(n, "array"\) # an array](#)
- [print symop\(n,"xyz"\) # xyz description \(from matrix\)](#)
- [print symop\(n,"array"\)\[1\] # xyz description \(original\)](#)
- [print symop\(n,"description"\) # text description](#)
- [print symop\(n,"array"\)\[2\] # translation vector \(fractional\)](#)
- [print symop\(n,"translation"\) # translation vector \(Cartesian\)](#)
- [print symop\(n,"center"\) # inversion point \(Cartesian\)](#)
- [print symop\(n,"axispoint"\) # axis point \(Cartesian\)](#)
- [print symop\(n,"axis"\) # axis vector \(Cartesian, normalized\)](#)
- [print symop\(n,"angle"\) # angle of rotation](#)

## 6. isosurface FULLPLANE

Jmol 11.9.3 adds an option to ISOSURFACE to allow data mapped onto a plane to extend throughout the whole plane, not just in localized spots.

- [load C6H6.smol;moveto /\\* time, axisAngle \\*/ 1.0 { 84 886 -456 176.91 } /\\* zoom, translation \\*/ 100.0 0.0 0.0 /\\* center, rotationRadius \\*/ {0.0 0.0 0.0} 3.657167 /\\* navigation center, translation, depth \\*/ {0.0 0.0 0.0} 0.0 0.0 0.0;](#)
- [isosurface plane {0 0 1 -1} map mo homo](#)
- [isosurface plane {0 0 1 -1} map FULLPLANE mo homo](#)

## 5. script/write state LOCALPATH/REMOTEPath

Jmol 11.9.2 adds two options to the SCRIPT and WRITE STATE commands. When reading or writing a script, the LOCALPATH keyword instructs Jmol to strip all paths beginning with "file:/" down to the indicated path. So, for example, script LOCALPATH "" "myfile.spt" indicates that all local files referenced in the state script should be read from the current default directory. Similarly, REMOTEPath strips paths from file references starting with "http:", "https:", and "ftp:". LOCALPATH and REMOTEPath are designed specifically to be used with scripts created by Jmol in the process of defining its state, but the option can be used with scripts you write yourself. The mechanism is simply looking for instances of /\*file\*/"some\_file\_name". If this construction is found in any script read, the replacement will be made.

- [script test1.spt #this won't work, because the script saved was for a state that had a file read from my laptop.](#)
- [script LOCALPATH "" "test1.spt" # this works, because the "file:" prefix and all subdirectory path information have been removed from the LOAD command. Note that the default directory for this page is "data", so the file read is 1crn.pdb in the data subdirectory.](#)

#### 4. {xxx}.volume("type")

Jmol 11.9.2 introduces a calculation for molecular volume. This calculation depends upon what set of Van Der Waals radii one uses. "type" can be one of "Jmol", "Babel", "Rasmol" or "User". If not provided, "type" defaults to the default type specified by the command **set defaultVanDerWaal**. The calculation is a simple one that just adds up all the Van Der Waals volumes and subtracts off the overlap of spheres associated with bonds as described by A. Bondi (*J. Phys. Chem.* **68**, 1964, 441-451.). The built-in Van der Waals set "Babel" most closely represents this setting.

- [load C6H6.smol](#);
- `print "" + {1.1}.volume()` # default is too large
- `print "" + {1.1}.volume("Jmol")` # same as default
- `print "" + {1.1}.volume("Babel")` # this is better
- `print "" + {1.1}.volume("Rasmol")`; # use unknown
- `{ C}.vdw = 1.65;print {1.1}.volume()`; # custom setting
- `set defaultVDW Babel;show VDW;{*}.vdw = 0;label %3.2[vdw]` # after setting the default you can do more with volume
- `label %3.2[volume]`
- `set defaultVDW RasMol;show VDW;{*}.vdw = 0;label %3.2[vdw]`
- `label %3.2[volume]`
- `set defaultVDW Jmol;show VDW;{*}.vdw = 0;label %3.2[vdw]`
- `label %3.2[volume]`

#### 3. script demo

- [script c6h12.txt](#)

The script [c6h12.txt](#) illustrates several advanced features of Jmol. In this script, a set of 35 cyclohexane conformations are fitted with quaternion frames on C1 and C4 carbons. A calculation is made of the similarity of each pair of quaternion frames within each model, allowing the rings to be colored based on how similar they are to a cyclohexane chair conformation. A bar graph near the bottom of the window is used to allow the user to quickly scan through conformations. The JavaScript callback for this action is created by the script itself, defining a JavaScript function on the page using the **javascript** command within Jmol.

The measure is calculated as

```
grp.property_st = 1.0 - acos(abs(q1.dot(q2)))/90
```

This measure is directly related to the "distance" between the quaternions on the four-dimensional hypersphere and is thus a direct measure of how well the two frames are aligned. If the frames are perfectly aligned, then this measure is 1.0; if they are completely oppositely aligned -- differing by a 180 degree rotation along any axis -- then this measure will be 0. The reason this works is that the dot product of two unit quaternions is the cosine of half their angular distance on the four-dimensional hypersphere: **q1.dot(q2) = cos(theta/2)**, similar to how for two 3-dimensional unit vectors v1 and v2, **v1.dot(v2) = cos(theta)**, where theta is the their angular distance on the unit sphere. We don't care about the sign of this value, because angles 0 and 180 degrees (cosines 1 and -1) for a quaternion relate to rotation angles 0 and 360 degrees, and those are the same for our purposes here. So taking the arccosine of the absolute value of the quaternion dot product returns a quaternion angle between 0 and 90, with 0 being "perfectly aligned" and 90 being "oppositely aligned." Dividing by 90 and subtracting from 1 gives a measure that is 1.0 for perfect alignment and 0.0 for opposite alignment.

#### 2. clickCallback

Starting with Jmol 11.9.1, you can detect clicking and dragging within the applet window, and you can cancel that operation if desired. The callback function takes the form:

```
function myClickCallback(app,x,y,modifiers,clickCount,Ret){}
```

**app** is the applet identifier, usually "JmolApplet0". **x** and **y** are screen coordinates with [0,0] in the bottom left corner. **modifiers** is a set of bits indicating what keys and mouse buttons were pressed: 1(shift), 2(ctrl), 4(right), 8(alt), 16(left). If none of these bits are set, this is a "mouse-up" event. **clickCount** is the number

of clicks. 0 here indicates the mouse was pressed (as in the initiation of a drag operation); negative numbers indicate dragging; positive numbers are standard clicks of the mouse. **Ret** is an array with just one element that allows returning a new modifier. Setting **Ret[0]=0** cancels Jmol processing of the event.

- [load caffeine.xyz;draw \[0 0%\] \[0 10%\] \[100 10%\] \[100 0%\] translucent white](#)
- [set clickcallback "myClickCallback"](#) # now watch the browser title line as you click on the screen
- [set clickcallback "myClickCallback2"](#) # now click or drag within the translucent strip

#### 1. acos(x) and q1.dot(q2)

Jmol 11.9.1 adds the `acos()` function, returning a value in degrees, and quaternion dot product.

- `print acos(0)`
- [print acos\(0.5\)](#)
- `print acos(1)`
- `print acos(-1)`
- `print " " + quaternion(1,2,3,4)`
- `print quaternion(1,2,3,4).dot(quaternion(1,2,3,4))%2` # q.dot(q) = 1
- `print quaternion(1,2,3,4).dot(-quaternion(1,2,3,4))%2` #note that q and -q represent the SAME rotation
- `print quaternion(1,0,1,0)` # 90 degrees around Y
- `print quaternion(1,0,0,1)` # 90 degrees around Z
- `print quaternion(1,0,1,0).dot(quaternion(1,0,0,1))%2`
- `print (quaternion(1,0,1,0) / quaternion(1,0,0,1))%0 %2` # q0 term of q2 / q1 is q1.dot(q2)