

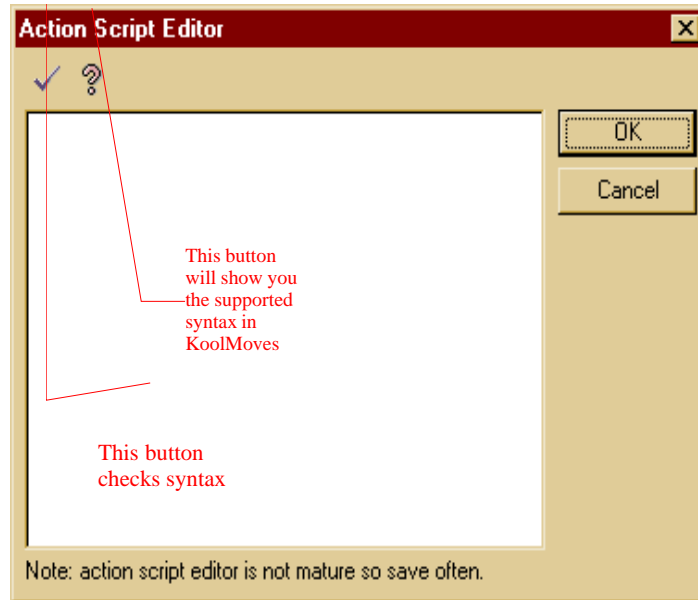
KoolMoves Actionscripting

This Tutorial is meant to be a basic introduction to the KoolMoves Actionscripting feature and is in no way meant to be an encompassing tutorial on Actionscripting but rather a basic guide to get you going.

First things first, The KoolMoves Actionscript feature and language is based on the ECMA (262) standard and uses the limited published Flash Objects and Functions in its syntax. You are probably wondering what that means to me? Well it means that the KoolMoves Actionscript are very Flash like and are almost, if not completely, identical to Flash 5 Actionscript (In expert Mode). This has many advantages;

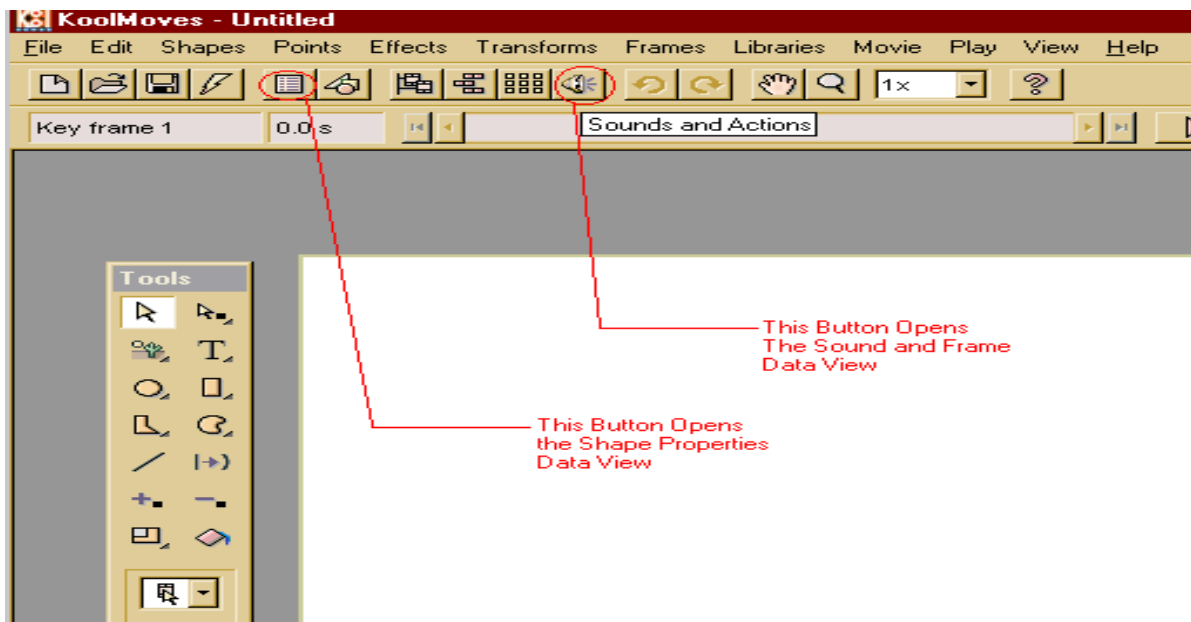
- What you learn in KoolMoves will migrate to Flash if you ever have a desire to use the two programs together. Your KoolMoves scripts have a very high likelihood to work in Flash 5 so if you choose to use Flash along with KoolMoves you won't have to unlearn your scripting techniques.
- There are already many tutorials and books written on the Flash 5 actionscripting language. You can use the Flash 5 resources and references to aide you in your scripting. In fact I recommend www.actionscripts.org, www.Flash5actionscript.com, and the Actionscripting tutorials at www.flashkit.com for a more complete reference. I also recommend the Flash 5 actionscripting for the Complete Idiot, Flash 5 Actionscripting for Dummies, and Flash 5 Actionscripting in a Nutshell books for further reference.
- This also means for all the Flash users out there this tutorial may also help them.

In order to use the Actionscript feature inside of KoolMoves you use either goto the frames Data View or the Shapes Properties Data View and Select Advanced>actionscript.



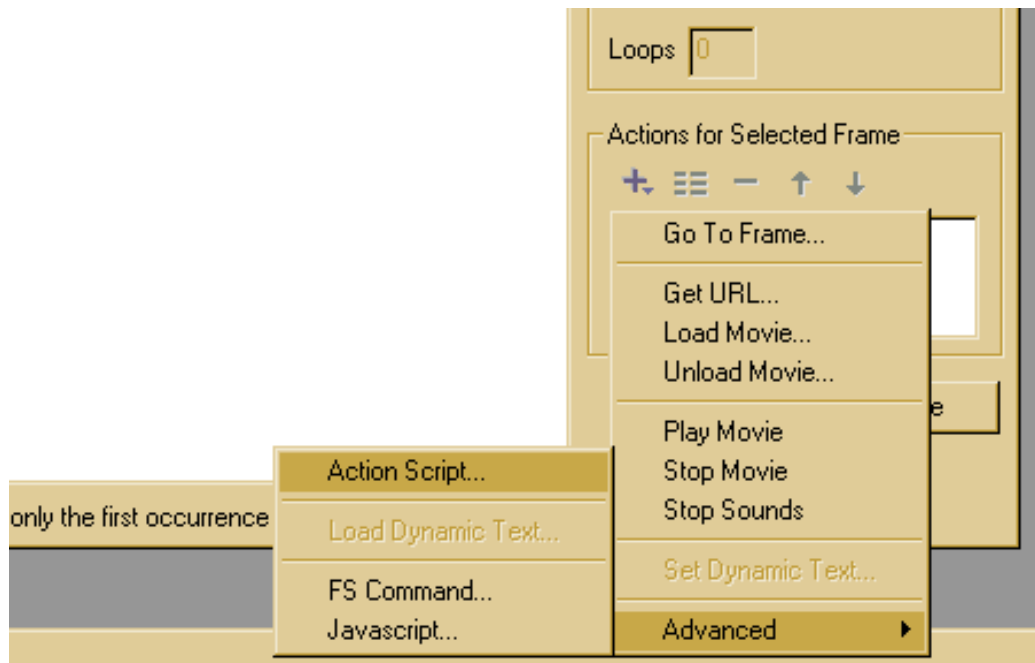
The KoolMoves Actionscript Interface is simple yet effective and consists of;

- Of a Box to type your script into.
- A button to check script syntax
- A button to show you all of the supported syntax in KoolMoves.



(This is Image demonstrates where the Data View buttons are located. The Shape Properties

Data view can also be alternately selected by double clicking a shape or button with the selection tool while the Frame and Sound Data view can also be popped up by hitting ctrl+e)



(This is an image of the sound and actions interface)

Imagine for a moment that you have a magic filmstrip that can tell the projector to do something besides play the picture. This filmstrip would play the First Frame then goto the next frame and so forth. On the Second frame you have instructions for it to open a box, name it, and store a value in it. Now that box didn't exist on the first frame so if you asked the projector on the first frame what that value was it would tell you that it didn't know. Likewise if the strip looped completely over again on the first frame it wouldn't know what was in that box, as it didn't exist until the second frame, and it would have completely recreated the box on the second frame. This is analogy is exactly how the Flash Player processes Actionscript. It processes the information hierarchy from when it was created along the timeline. So ,if you for instance, check for a value of a variable (don't worry I will explain variables in a minute) before that variable exists in the timeline you will return an error or a Null (meaning nothing) value. Actionscript are nothing but instructions along the timeline (strip) that tell the player to do things.

You can also attach Actionscripts to buttons as a button action . There doesn't appear to be a limit on the number of button actions that a button may have so for debugging purposes I suggest dividing your total number of actions on a button over multiple action statements. I also highly suggest naming your buttons using the shape name interface so that you can find them quickly as once you assign a button an action that action is permanently attached to the button throughout the entire movie. Button actions require that you start them with an **on(button action){ script goes between here and }** . You can assign multiple actions to any script either button or frame by using the ; to signify that there is more script, just like in Ecma script. It is important to

remember the braces as your scripts will not work right without them.

Now that we have a basic understanding of what Actionscript are and what they can be assigned to lets talk about variables. Variables are pretty much like boxes or file cabinets. When you define a variable you tell the Flash player to create a space in the computer memory that contains a value and has a name. Now in Flash you can have 3 types of values assigned to a variable:

- A numerical Value EG: 1, 2, 3, 4 ... n
- A String EG: Bob, Frank, Suzy, Fred, ECT
- A Boolean Value EG: True or False.

The Flash Player will only store the type of data of your variable that was present when you created it. So if for example you create a variable and assign it a value of "Bob" EG: a=Bob; then that variable will only contain string data (Note anytime you place quotes around a variables contents it is a string). If you create a variable and assign it a numerical value EG; b = 1; then it will contain numerical data. You can also define numerical data as a mathematical function EG: A = 1 + 1. Variables are case sensitive so it is important that you pay attention to the uppercase and lowercase spellings. It is also important to note that variables retain their values as long as they are not redefined or the movie loops over to the very first frame.

Short Quiz:

Okay here is a list of variables and data types, try to decide the type of data

```
A = 1;
a = "1";
b = frank;
C = true
a = 1 + 1
```

In the first example the data type is numerical, in the second it is a string because of the "", in the third example it is a string type also, the fourth is a Boolean, and the last one is numerical.

We can also do things with these variables like add and subtract them. If we can add variables together for example c = a + b. If these variables have numerical values in them then it will add them and return the value EG script;

```
a = 1;
b = 2;
c = a + b;
```

will return a value of 3 for c. However if those variables contained strings it will conocate the strings EG script;

```
a = " I am ";
b ="Kool";
c = a + b;
```

will return the Value: " I am Kool".

You can also use the increment and decrement functions, e.g., ++I when you use this feature it will either add 1 or subtract one from your script.

You can manipulate variables with Operators. An example of an operator is $a + 1$. The + is a mathematical operator. This is a list of the KoolMoves operators and what they do;

=

sets the value of a variable or property globally, e.g., $x = 200$

+

add, e.g., $100 + b$

-

subtract, e.g., $100 - b$

*

multiply, e.g., $100 * b$

/

divide, e.g., $100 / b$

+=

add assign, e.g., $b += 5$ (this is equivalent to $b = b + 5$)

-=

subtract assign, e.g., $b -= 5$ (this is equivalent to $b = b - 5$)

*=

multiply assign, e.g., $b *= 5$ (this is equivalent to $b = b * 5$)

/=

divide assign, e.g., $b /= 5$ (this is equivalent to $b = b / 5$)

++

increment, e.g., ++i; **note: i++ is not supported yet in KoolMoves**

--

decrement, e.g., --i; **note: i-- is not supported yet in KoolMoves**

<

less than test, e.g., a < b

<=

less than or equal test, e.g., a <= b

>

greater than test, e.g., a > b

>=

greater than or equal test, e.g., a >= b

==

equals test, e.g., a == b

Now if you want to be technical something like $a = b + 1$ is called an expression. Expressions are made from Operands (your variables, numbers, or strings) and Operators (like + or -). An expression is something that the Flash Player Actionscript interpreter can evaluate to a value, string or what have you.

Now that we have the basic variable syntax down we will look at how to display variables inside of KoolMoves. In order to display a variable value you need to create dynamic text boxes. Dynamic text boxes both act as input and output boxes inside the flash player. Each text box is tied to a variable. KoolMoves, for the sake of simplicity names the text box and variable attached to it the same thing. This is important because once you create the text box on the timeline you have to remember that it gets formatted at that time. You can rename any text object by double clicking on the dynamic text objects name in the Shapes in Frame Data view.

Okay are you ready for the tried and tired “Hello World” Example?

This will be your first full script in KoolMoves:

1. Draw A Text Box using the Dynamic Text Drawing tool.



2. Create 3 Key Frames.
3. Enter into the Sound and Actions interface.
4. On the second frame add this action;

```
txt1 = "hello world";
```

5. On the 3 third frame add the stop movie action.

If you preview this inside of a Flash 5 enabled web browser or inside a Flash 5 projector file you should see a text box with the contents hello world in it.

Text objects have a few properties that you can both read and set. Two special properties are scroll and set. This property tells the Flash Player which line to display the text on. The scroll property is set by using the equal sign EG: `txt1.scroll = 5` while you can also get the property of the scroll by setting the property as a value of another variable EG: `a = txt1.scroll`.

Another special text property that you can get and set is the maxscroll property. This property tells the Flash Player how many lines a text object can scroll down. Again you can read the property by setting it as another variable EG: `a = txt1.maxscroll` and can also be set the same way as the scroll property EG: `txt1.maxscroll = 5`.

Here is a useful script that makes a scroller.

```
txt1.scroll += 1;
```

```
if(txt1.scroll == txt1.maxscroll){txt1.scroll = 0 };
```

Okay let's talk about paths to variables and objects. Believe it or not every time you access a variable you are using a path. So for instance if I type `a = 10` what the flash Player reads is `this.a = 10`. The player doesn't require the `this`. To be included although there may come a time when it is (at the time of this writing `this` and `_parent` are not support in the syntax).

This is a list of target identifiers:

- `_parent` : This is the relative path for a higher movie or level.
- `this` : This is the level or object itself- it is redundant.
- `_level#` : This points to a specific level in the movie. `_level0` is the main movie
- `_root` : This points to the main stage of the movie regardless of where you are.
- MovieClip: You can name your MovieClip in the target.

Okay so this is where it gets complicated. If you are dealing with multiple layers in Flash you must reference the target layer by identifying the each layer. So if we had an SWF object, named `swf1`, on the 2 level of our movie to reference it we would need to type `_level2.swf1` or we could also reference it as `_root.level2.swf1`. Now if we wanted to control the properties of something on the leveled movies timeline from `swf1` we could access it like this `_root.Level2` or `_parent`.

Now that we know how to target things we can start adjusting things. As of this time these are the properties that KoolMoves supports and what they do:

- `_x` The X position of the center point of a movie clip (left to right)
- `_y` The Y position of the center point of a movie clip (top to bottom)
- `_width` the width of a movie clip
- `_height` the height of a movie clip
- `_rotation` the rotation of a movie clip (in degrees)
- `_target` the target path of a movie clip (includes FULL path)
- `_name` the name of an instance of a movie clip
- `_url` the full URL of the .swf that contains the movie clip
- `_xscale` the scale of the x axis (in %) of a movie clip
- `_yscale` the scale of the y axis (in %) of a movie clip
- `_currentframe` the current frame of a movie clip
- `_totalframes` the total number of frames in a movie clip
- `_framesloaded` the number of frames in a movie clip that are loaded
- `_alpha` The alpha (transparency) of a movie clip (measured in %, 0 is fully transparent)
- `_visible` the visibility of a movie clip (true/false)
- `_droptarget` when a movie clip is dragged, this is equal to the path of the target beneath the mouse pointer. Or, if no path is given, it equals "/", the target of the main timeline
- `_xmouse` is the position from the left hand corner of the target
- `_ymouse` is the position from the top of the target
- `_highquality` High Quality
- `_focusrect` Show Focus Rectangle
- `_soundbuftime` Sound Prebuffer time

Most of these properties are only good for SWF Objects (Movie Clips). In order to set the properties of an Object you type: `target.property =` in order to read the value of one of these properties you assign it to a variable EG: `a = swf1._x`

Here is an example of both reading and assigning properties:

1. Create a new movie in KoolMoves and import a SWF as object and create 3 more frames.
2. Click on the Sounds and Frames Dialog on the second frame.
3. Select an Actionscript action and enter;

```
swf1._x = _root._xmouse;  
swf1._y = _root._ymouse;
```

4. Select a Goto and Play action and set it to frame 2.

If you did this correctly you should have a 1 frame movie in which your swf object will follow your mouse.

The last part of actionscripting is to use the functions. KoolMoves has a standard set of basic Flash Functions they are;

eval()

evaluates a string for a value

geturl(url, [window, [string variables]])

variables (optional) is get or post

gotoandplay(frame)

frame argument is an expression, frame number or label

gotoandstop(frame)

frame argument is an expression, frame number or label

if (expression) { statements }

statements are executed if the expression evaluates to true

loadmovie(url, location, [string variables])

location is a level or movie clip; variables (optional) is get or post

loadvariables(url, location, [string variables])

location is a level or target; variables (optional) is get or post

nextframe()

go to next frame

on (mouse event) { statements }

statements are executed if mouse event (release, rollover, etc) occurs

play()

play the movie

prevframe()

go to previous frame

stop()

stop the movie

stopallsounds()

stop all sounds

unloadmovie(location)

location is a level or movie clip

var

define locally, e.g., var x;