

macromedia FLASH 5 QUICK ACTIONSCRIPT

I wrote this document while learning the new actionscript. I like to write things down, it helps me understand what's going on. Only later I realized that it might help others too. This quick reference is certainly not intended to replace any good book, manual or online help. What it does is that if you print it, you have quick access to all key elements of actionscript. For each command, property, function or method, you get the full syntax, a short explanation, and examples. Keep in mind that this reference makes no attempt to explain basic concepts in Flash programming, so if you don't know what a movie clip is, this won't teach you. The examples provided are a combination of real-life scripts with something made up for demonstration purposes only. The aim was to present you with as many different situations as possible and to show you how to use the syntax properly. The examples are not optimized and they were never meant to be.

My hope is that by printing this quick reference, and using it with Flash, you'll learn new stuff and be more productive. Please let me know if you find this document helpful or maybe I just wasted my time?

*Armand Niculescu
armand@media-division.com*

Disclaimer: This document is provided as is and without any warranties of any kind, whether express or implied with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Digital Vision Multimedia shall not be liable for errors contained herein or for any damages in connection with this document. Macromedia and Flash are registered trademarks of Macromedia Inc.

Basic Actions

<code>gotoAndPlay ([scene], frame);</code>	Go to the specified frame (or label) and play.
<code>gotoAndStop ([scene], frame);</code>	Go to the specified frame (or label) and stop.
<code>nextFrame ();</code>	Go to the next frame.
<code>prevFrame ();</code>	Go to the previous frame.
<code>nextScene ();</code>	Go to the next scene.
<code>prevScene ();</code>	Go to previous scene.
<code>play ();</code>	Play or resume playing from the current frame.
<code>stop ();</code>	Stop the movie.
<code>toggleHighQuality ();</code>	Toggle this global property. Deprecated. (See <code>_quality</code>)
<code>stopAllSounds;</code>	Stop all sounds that are currently playing.
<code>getUrl (url, [window], ["GET" "POST"]);</code>	Streaming sound will resume on next frame. Tell the browser to go to the specified URL, optionally sending variables (i.e. if the URL is a CGI script)
<code>fscommand (command, arguments);</code>	Send a command to the movie container (Browser, Flash player, Visual C++ program)
FS Commands for Flash Standalone player:	
<code>Fullscreen(true false)</code>	Launch the player in full screen mode;
<code>allowscale(true false)</code>	The movie may be resized;
<code>showmenu(true false)</code>	Show the right-click context menu;
<code>exec(parameter)</code>	Execute an external program;
<code>quit</code>	Quit the player.
<code>loadMovie (url, target, ["GET" "POST"]);</code>	Load a movie from an URL, optionally send variables to it and place it inside an existing movie clip.
<code>loadMovieNum (url, level, ["GET" "POST"]);</code>	Load a movie from an URL, optionally send variables to it and place it on the specified level.
<code>unloadMovie (target);</code>	Unload a previously loaded movie from inside a movie clip.
<code>unloadMovieNum (level);</code>	Unload a previously loaded movie from a level.
<code>tellTarget (target) { }</code>	Send commands to the specified movie clip. Deprecated. (See <code>MovieClip</code> object)
<code>ifFrameLoaded ([scene], frame) { }</code>	Test whether the specified frame is loaded. Deprecated. (See <code>_framesloaded</code>)

```
//some simple actions
gotoAndPlay (20);
gotoAndStop ("myLabel");
getUrl ("http://www.server.dom/search.cgi", "", "POST");
fsCommand ("fullscreen", "true");
loadMovieNum ("newmovie.swf", 0);
```

```
//working with button events
//old way
on (rollOver) {
    tellTarget ("myClip") {
        gotoAndPlay ("fadeIn");
    }
}
on (rollOut) {
    tellTarget ("myClip") {
        gotoAndPlay ("fadeOut");
    }
}
//new way
on (rollOver) {
    myClip.gotoAndPlay ("fadeIn");
}
on (rollOut) {
    myClip.gotoAndPlay ("fadeOut");
}
```

Actions

<code>break;</code>	Break out of an enclosing loop.
<code>continue;</code>	Continue at the start of the enclosing loop.
<code>call (frame);</code>	Calls a subroutine at a frame. Deprecated. (See <code>function</code>)
<code>function name (parameters);</code>	Declare a function with optional parameters.
<code>return value;</code>	Return a variable from a function.
<code>do { } while (condition);</code>	Do a set of actions while the condition is true. End test.
<code>while (condition) { }</code>	Do a set of actions while the condition is true. Beginning test.
<code>duplicateMovieClip (target, new name, depth);</code>	Make a duplicate of a movie clip.
<code>removeMovieClip (target);</code>	Remove a clip created with Duplicate Movie Clip.
<code>for (init; condition; next) { }</code>	Execute a set of actions for a number of times.
<code>for (iterator in object) { }</code>	Loop through the properties of an object.
<code>if (condition) { } else if { } else { }</code>	Perform actions if a condition is true, else other actions.
<code>#include "path"</code>	Include a script from a file.
<code>loadVariables (url, target, ["GET" "POST"]);</code>	Load variables from an URL in a movie clip.

```
//using loops, if, break and continue
i = 0;
while (true) {
    ++i;
    if (i > 9) {
        break;
    }
    if ((i/2) != int(i/2)) {
        continue;
    }
    trace (i);
}
Result:
2 4 6 8
```

loadVariablesNum (url, level, ["GET" | "POST"]); Load variables from an URL in a level.

on

- press** Perform action when an event occurs on the button:
- release** The mouse left button is pressed on the button;
- releaseOutside** The mouse left button is released on the button
- rollOver** The mouse left button is released outside the button;
- rollOut** The mouse pointer enters the button area;
- dragOver** The mouse pointer leaves the button area;
- dragOut** A movie clip is dragged over the button;
- KeyPress** ("key") A movie clip is dragged out of the button;

onClipEvent

- load** A key is pressed.
- unload** Perform action when an event occurs on a movie clip:
- enterFrame** The movie clip is loaded in movie;
- mouseUp** The movie clip is removed from movie;
- mouseDown** Any frame is played;
- mouseMove** The mouse pointer is released anywhere on the screen;
- keyUp** The mouse left button is pressed anywhere on the screen;
- KeyDown** The mouse pointer moves;
- Data** A key is released

print (target, location, method); Data is received (with loadVariables or loadMovie).

printNum (level, location, method); Print a movie clip in vector format.

printAsBitmap (target, location, method); Print a level in vector format.

printAsBitmapNum (level, location, method); Print a movie clip, treating it as bitmap.

target Print a level, treating it as bitmap.

The instance name of movie clip to print.

To print only specific frames, assign them "#P" frame label

Print methods:

- bmovie** Designate the bounding box of a frame in a movie as print.
- bmax** Assign "#b" to the frame whose bounding box you want to use as the print area;
- bframe** Designate a composite of all of the bounding boxes, of all the printable frames, as the print area;

variable = value Designate that the bounding box of each printable frame be used as the print area for that frame.

set (var_expression, value); Set a variable.

setProperty (target, property, value); Set a variable, where variable's name is an expression.

startDrag (target, [true], [left, top, right, bottom]); Set a property for a movie clip

Drags a movie clip. [true] = LockMouse pointer to center of movie clip. With left, top, right, bottom, assign the draggable area.

stopDrag (); Stop the Drag action.

// comment Add a comment to the code.

delete variable | object; Delete a variable or an object.

trace (expression); Send a message to the Output window. Used mostly to debug

var variables; Declare local variables. Best used within functions.

with (object) { } Operations will affect specified object.

```
//working with simple functions
function average ( min, max ) {
    var avg;
    avg = ( min + max ) / 2;
    return avg;
}

trace (average ( 2, 4 ));
```

Result:
3

```
//event attached on a movie clip
onClipEvent (mouseDown) {
    startDrag ("", true, 0, 0, 50, 50);
}
onClipEvent (mouseUp) {
    stopDrag ();
}
```

```
//define a new empty object
myObject = new Object ();
//add some elements to it
myObject.property1 = "red";
myObject.property2 = 10;
myObject.property3 = new Array (1, 2, 3);
//iterates the elements like this
for (property in myObject) {
    trace ("myObject." + property + " : " + myObject[property]);
}
//or set the variables like this (not very effective)
with (myObject) {
    for (i=1; i<=3; i++) {
        set ("property"+i, 7 );
        trace (myObject["property"+i]);
    }
}
```

Result:
myObject.property3 = 1,2,3
myObject.property2 = 10
myObject.property1 = red
7
7
7

Operators

!	Logical NOT.
&&	Logical AND.
	Logical OR.
*	Multiply.
+	Add.
++	Increment a variable.
-	Subtract.
--	Decrement a variable.
/	Division.
%	Remainder of division x/y
!=	Test inequality.
<	Less than.
<=	Less than or equals.
<>	Not equals. Deprecated. (See !=)
==	Test equality.
>	Greater than.
>=	Greater than or equal.
and	Logical And. Deprecated. (See &&)
or	Logical Or. Deprecated. (See)
not	Logical Not. Deprecated. (See !)
typeof (expression);	Return the type of expression.
void (expression);	Evaluate expression, return undefined.
&	Bitwise AND.
<<	Shift left bits.
>>	Shift right bits.
>>>	Shift right, unsigned.
^	Bitwise XOR.
	Bitwise OR.
~	Bitwise complement.
%=	Equivalent to A = A % B.
&=	Equivalent to A = A & B.
*=	Equivalent to A = A * B.
+=	Equivalent to A = A + B.
-=	Equivalent to A = A - B.
/=	Equivalent to A = A / B.
<<=	Equivalent to A = A << B.
>>=	Equivalent to A = A >> B.
>>>=	Equivalent to A = A >>> B.
^=	Equivalent to A = A ^ B.
=	Equivalent to A = A B.
add	concatenate strings. Deprecated. (See +)
eq	test string equality. Deprecated. (See ==)
ge	string greater than or equal. Deprecated. (See >=)
gt	string greater than. Deprecated. (See >)
le	string less than or equal. Deprecated. (See <=)
lt	string less than. Deprecated. (See <)
ne	string not equal. Deprecated. (See !=)

```
//How to use logical
connectors
```

```
a = true;
b = false;
```

```
c = !a;
d = a && b;
e = a || b;
```

Result:

```
a = true
b = false
```

```
c = false
d = false
e = true
```

```
//How to use left and
right shift
```

```
a = 1;
b = a << 8;
c = b >> 1;
trace ("a = " + a);
trace ("b = " + b);
trace ("c = " + c);
```

Result:

```
a = 1
b = 256
c = 128
```

```
//Simple example on how to use bitwise operators
```

```
a = 1;
b = 1;
c = 0;
e = a | b;
f = a & b;
g = a ^ b;
h = a | c;
i = a & c;
j = a ^ c;
trace (a + " OR " + b + " = " + e);
trace (a + " AND " + b + " = " + f);
trace (a + " XOR " + b + " = " + g);
trace (a + " OR " + c + " = " + h);
trace (a + " AND " + c + " = " + i);
trace (a + " XOR " + c + " = " + j);
```

```
1 OR 1 = 1
1 AND 1 = 1
1 XOR 1 = 0
1 OR 0 = 1
1 AND 0 = 0
1 XOR 0 = 1
```

```
//Simple shortcut
```

```
A = 5;
B = 2;
A += B; //equivalent of writing a = a + b
Trace (a);
```

Result: 7

Functions

<code>Boolean</code> (expression);	Convert argument to Boolean.
<code>Number</code> (expression);	Convert argument to number.
<code>String</code> (expression);	Convert argument to string.
<code>escape</code> (string);	Convert characters in a string with escape codes.
<code>unescape</code> (string);	Convert the escape codes in a string to characters.
<code>eval</code> (variable);	Return the value of variable named by expression.
<code>false</code> ;	The value False (0).
<code>true</code> ;	The value True (1).
<code>getProperty</code> (target, property);	Return a property of the specified movie clip.
<code>getTimer</code> ();	Number of milliseconds passed since the movie started.
<code>getVersion</code> ();	Version number of the player plugin (e.g. WIN 5,0,30,0).
<code>random</code> (number);	Generate a random value between 0 and number.
	Deprecated. (see <code>Math.random</code>)
<code>targetPath</code> (movieclip);	Returns the target path string for a movie clip (dot notation).
<code>int</code> (number);	Integer part of a number. Deprecated. (See <code>parseInt</code>)
<code>isFinite</code> (number);	Test whether a number is finite
<code>isNaN</code> (number);	Test whether a number is Not a number
<code>.maxscroll</code> ;	Maximum value of <code>.scroll</code> in a text field
<code>.scroll</code> ;	Index of first visible line in a text field
<code>newline</code> ;	The 'newline' character
<code>parseFloat</code> (string);	Parse string into floating point number
<code>parseInt</code> (string, radix);	Parse a string into an integer with radix conversion (2, 10, 16)
<code>updateAfterEvent</code> (event);	Immediately refreshes the display
<code>chr</code> (asciiCode);	Convert an ASCII code to character. Deprecated. (See <code>.fromCharCode</code>)
<code>length</code> (string);	Return the length of a string. Deprecated. (See <code>.length</code>)
<code>ord</code> (character);	Convert a character to an ASCII code. Deprecated. (See <code>.charCodeAt</code>)
<code>substring</code> (string, index, count);	Extract from string count characters, starting from index. Deprecated. (See <code>.substr</code>)
<code>mbchr</code> (asciiCode);	Convert an ASCII code (multibyte) to a character. Deprecated
<code>mbLength</code> (string);	Return the length of a string (multibyte). Deprecated
<code>mbOrd</code> (character);	Convert a character to an ASCII code (multibyte). Deprecated
<code>mbSubstring</code> (string, index, count);	Extract from string (multibyte) count characters, starting from index. Deprecated

```
Trace (escape ("C:\Windows\My Documents"));
Trace (unescape ("This%20is%20a%20test"));
```

```
Result:
C%3A%5Cindows%5Cy%20Documents
This is a test
```

```
//working with objects - Flash5 style
onClipEvent (mouseDown) {
    // "this" is the equivalent of "this" in Flash4. It means
    // "myself" (the current object)
    this.startDrag ();
}
```

```
// "_root" is the equivalent of "/" in Flash4. It means
// an object (variable, movie clip, and so on) located
// on top of hierarchy (the Stage). Used for absolute
// referencing
_root.myclip.subclip.gotoAndPlay (2);
x= _root[bigClip.smallclip+i]._x;
```

```
// "_parent" is the equivalent of "." in Flash4. It
// leads to an object higher one level in hierarchy
// (object's parent). Used for relative referencing.
myVar += _parent.otherVar;
```

Properties

<code>_alpha</code>	Alpha transparency of a movie clip (percentage).
<code>_visible</code>	Visibility of a movie clip (true/false).
<code>_currentframe</code>	Current frame of a movie clip.
<code>_totalframes</code>	Total number of frames in a movie clip.
<code>_framesloaded</code>	Number of loaded frames in a movie clip.
<code>_url</code>	The URL location of a movie clip.
<code>_height</code>	Height of a movie clip.
<code>_width</code>	Width of a movie clip.
<code>_xscale</code>	Scale factor of a movie clip on X axis (percentage).

```
//get a movie clip's properties and find the percentage
//of frames played so far
with (myClip) {
    cf = _currentframe;
    tf = _totalframes;
    percentPlayed = cf/tf * 100;
}
```

<code>_yscale</code>	Scale factor of a movie clip on Y axis (percentage).
<code>_rotation</code>	Rotation of a movie clip (degrees).
<code>_focusrect</code>	Focus rectangles for buttons, fields with TAB key (true/false).
<code>_highquality</code>	High quality rendering (true/false). Deprecated. See <code>_quality</code>
<code>_quality</code>	Quality as string: "low", "medium", "high", "best".
<code>_soundbuftime</code>	Seconds of sound to prebuffer streaming sound (default = 5).
<code>_droptarget</code>	The movie clip below the currently dragging movie clip.
<code>_name</code>	The name of the movie clip.
<code>_target</code>	Target path of a movie clip.
<code>_x</code>	Position (X) of a movie clip.
<code>_y</code>	Position (Y) of a movie clip.
<code>_xmouse</code>	Position (X) of the mouse pointer.
<code>_ymouse</code>	Position (Y) of the mouse pointer.

```
// change clip's properties if it is dragged-n-dropped
// over another clip
if (this._droptarget = "trash") {
    this._x = 50;
    this._y = 50;
    this._visible = false;
}

//if move clip's name is let's say "car", it will tell a
sibling clip, "cars", to stop.
myName = this._name;
_parent[myName + "s"].stop ();
```

Objects

Array

<code>new Array ();</code>	Create a new array object.
<code>.concat (array1, ... , arrayN);</code>	Concatenate this array to others and return new array.
<code>.join (separator);</code>	Join all array elements into string, separated by separator.
<code>.length;</code>	Length of array.
<code>.pop ();</code>	Return the last item in array and remove it.
<code>.push (value);</code>	Append new values at the end of array.
<code>.shift ();</code>	Return the first element in array and removes it.
<code>.reverse ();</code>	Reverse the elements in array.
<code>.sort (compareFunction);</code>	Sort an array. CompareFunction can be 1, 0, -1.
<code>.slice (indexA, indexB);</code>	Returns new array of elements from indexA to indexB.
<code>.splice (index, count, elem1, ..., elemN);</code>	Delete count items at index, append elements if specified.
<code>.unshift (value);</code>	Insert new items at the beginning of array.

Boolean

<code>new Boolean (value);</code>	Create a new Boolean object.
<code>.toString();</code>	Convert Boolean object to string.
<code>.valueOf();</code>	Convert the Boolean object.

Color

<code>new Color (target);</code>	Create a color object.
<code>.getRGB ();</code>	Get the color RGB offsets transform.
<code>.getTransform (transformobject);</code>	Get the color transform..
<code>.setRGB (0xRRGGBB);</code>	Set the RGB offsets for the color transform.
<code>.setTransform (transformobject);</code>	Set the color transform.
<code>transformobject = { ra: '0', rb: '0', ga: '0', gb: '0', ba: '0', bb: '0', aa: '0' }</code> ra, ga, ba, aa: percentage of red, green, blue, alpha (-100, 100) rb, gb, bb, ab: offset of red, green, blue, alpha (-255, 255)	

```
//working with arrays
myArray = new Array(5, 10, 15);
myArray.push(20);
extract = myArray.shift();
myArray.reverse();
myArray.unshift(3);
extract = myArray.pop();

myNewArray = new Array(8, 14, 23);
bigArray = myArray.concat( myNewArray );
bigArray.sort();
len = bigArray.length;
bigArray.splice( 1, 2, 1 );
smallArray = bigArray.slice( 1, bigArray.length -1 );

myString = smallArray.join( "|" );
```

Explanation:
 At first, it looks like this: 5,10,15
 Then we add a new item at the end: 5,10,15,20
 And extract the first item (5): 10,15,20
 We reverse the elements: 20,15,10
 Add a new item at the beginning: 3,20,15,10
 But remove the last item (10): 3,20,15
 Now we have one big array: 3,20,15,8,14,23
 And we sort it: 3,8,14,15,20,23
 Its length is 6
 We delete the 2nd & 3rd elements and add '1' in place:
 3,1,15,20,23
 Then we remove the first and last items: 1,15,20
 Finally, we convert what's left to a string: 1|15|20

Date

new Date (year, month, date, hour, min, sec, ms); Create a new Date object.

.getDate (); Get day of month (local).

.getDay (); Get day of week (0..6) (local).

.getFullYear (); Get the year (local).

.getHours (); Get hours of the day (0..23) (local).

.getMilliseconds (); Get milliseconds in current second (local).

.getMinutes (); Get minutes of current hour (local).

.getMonth (); Get month of the year (0..11) (local).

.getSeconds (); Get seconds in minute (local).

.getTime (); Get seconds since midnight Jan 1 1970 UTC.

.getTimezoneOffset (); Get offset of local time from UTC time in minutes (local).

.getUTCDate (); Get day of month (UTC time).

.getUTCDay (); Get day of week (0..6) (UTC time).

.getUTCFullYear (); Get the year (UTC time).

.getUTCHours (); Get hours of the day (0..23) (UTC time).

.getUTCMilliseconds (); Get milliseconds in current second (UTC time).

.getUTCMinutes (); Get minutes of current hour (UTC time).

.getUTCMonth (); Get month of the year (0..11) (UTC time).

.getUTCSeconds (); Get seconds in minute (UTC time).

.getYear (); Get year -1900 (local).

.setDate (date); Set Day of month (local).

.setFullYear (year, month, date); Set the full year (local).

.setHours (hours, minutes, seconds, ms); Set hour of day (local).

.setMilliseconds (ms); Set millisecond in the current second (local).

.setMinutes (minutes, seconds, ms); Set minute of hour (local).

.setMonth (month, date); Set month of the year (local).

.setSeconds (seconds, ms); Set seconds of minute (local).

.setTime (value); Set seconds since Jan 1 1970 UTC.

.setUTCDate (date); Set Day of month (UTC).

.setUTCFullYear (year, month, date); Set the full year (UTC).

.setUTCHours (hours, minutes, seconds, ms); Set hour of day (UTC).

.setUTCMilliseconds (ms); Set millisecond in the current second (UTC).

.setUTCMinutes (minutes, seconds, ms); Set minute of hour (UTC).

.setUTCMonth (month, date); Set month of the year (UTC).

.setUTCSeconds (seconds, ms); Set seconds of minute (UTC).

.setYear (year, month, date); Set the year -1900 (UTC).

.toString (); Return a string.

Date.UTC (year, month, date, hour, min, sec, ms); Return a time value for specific UTC date and time.

Key

Key.getAscii (); Get the ASCII code of the last pressed/released key.

Key.getCode (); Get the key code of the last pressed/released key.

Key.isDown (keyCode); Return true if key is down.

Key.isToggled (keyCode); Return true if key is CapsLock or NumLock is activated.

Key.BACKSPACE; Backspace key code.

Key.CAPSLOCK; CapsLock key code.

```
//working with the color object
clipColor = new Color("colorClip");

redValue = 128;
greenValue = 128;
blueValue = 255;

clipColor.setRGB(redValue<<16|greenValue<<8|blueValue);
```

```
//working with time & date and arrays
monthsArray = new Array("Jan", "Feb", "Mar", "Apr",
"May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
daysArray = new Array("Sun", "Mon", "Tue", "Wed", "Thu",
"Fri", "Sat");

myDate = new Date();
with (myDate) {
    year = getFullYear();
    month = getMonth();
    day = getDate();
    dayOfWeek = getDay();
    hour = getHours();
    mins = getMinutes();
    secs = getSeconds();
}

monthString = monthsArray[month];
dayString = daysArray[dayOfWeek];
dateString = dayString + ", " + monthString + " " + day
+ " " + year;
trace (dateString);
timeString = hour + ":" + mins + ":" + secs;
trace (timeString);

Result:
Sun, Oct 1 2000
23:34:41
```

```
//working with keys and movie clips
onClipEvent (enterFrame) {
    if(Key.isDown(Key.RIGHT)) {
        setProperty ("", _x, _x+1);
        updateAfterEvent ();
    }
}
```


Key.CONTROL;	Control key code.
Key.DELETEKEY;	Delete key code.
Key.DOWN;	Down Arrow key code.
Key.END;	End key code.
Key.ENTER;	Enter key code.
Key.ESCAPE;	Escape key code.
Key.HOME;	Home key code.
Key.INSERT;	Insert key code.
Key.LEFT;	Left Arrow key code.
Key.PGDN;	Page Down key code.
Key.PGUP;	Page Up key code.
Key.RIGHT;	Right Arrow key code.
Key.SHIFT;	Shift key code.
Key.SPACE;	Space key code.
Key.TAB;	Tab key code.
Key.UP;	Up Arrow key code.

Math

Math.abs (number);	Absolute value of a number.
Math.cos (number);	Cosine of angle (radians).
Math.sin (number);	Sine of angle (radians).
Math.tan (number);	Tangent.
Math.acos (number);	ArcCosine.
Math.asin (number);	ArcSine.
Math.atan (number);	ArcTangent.
Math.atan2 (y, x);	ArcTangent (y/x).
Math.ceil (number);	Round Up number.
Math.floor (number);	Round down number.
Math.round (number);	Round to nearest integer
Math.E;	Euler constant.
Math.PI;	"π" (Pi) constant.
Math.exp (number);	Exponential (E to the power of number).
Math.pow (base, exponent);	number at the power of exponent
Math.sqrt (number);	Square root.
Math.SQRT1_2;	Square root of ½
Math.SQRT2;	Square root of 2.
Math.LN10;	Natural logarithm of 10.
Math.LN2;	Natural logarithm of 2.
Math.log (number);	Natural logarithm of number.
Math.LOG10E;	Base10 logarithm of E.
Math.LOG2E;	Base 2 logarithm of E.
Math.max (x, y);	Maximum of two numbers.
Math.min (x, y);	Minimum of two numbers.
Math.random ();	Random number between 0 and 1.

```
//working with Math and functions
function degToRad ( degree ) {
    radian = Math.PI/180 * degree;
    return radian;
}

function radToDeg ( radian ) {
    degree = 180/Math.PI * radian;
    return degree;
}

//compute weight of the sphere, based on density and radius
volume = Math.PI * 4/3 * Math.pow (radius, 3);
mass = density * volume;
weight = 0.98 * mass;

// decompose velocity on its X and Y axes
radAngle = degToRad (angle);
vectorx = velocity * Math.cos (radAngle);
vectory = velocity * Math.sin (radAngle) + weight;

//recompose velocity and angle with the new values.
velocity = Math.sqrt (vectorx*vectorx+vectory*vectory);
radAngle = Math.atan2 (vectory, vectorx);
angle = radToDeg (radAngle);

//create a random value between -50 and 50
noise = Math.random () * 100 - 50;
noise = Math.round (noise);
```

```
//you can create a custom object like this...
coords = new Object ();
coords.x = 10;
coords.y = 10;

//or like this...
coords2 = {x:10, y:10};

//and modify their properties like this.
with (coords) {
    x = 0;
    y = 0;
}
```


Movie Clips

<code>.attachMovie</code> (idName, newName, depth);	Create instance name of clip from library (Linkage/export)
<code>.duplicateMovieClip</code> (newName, depth);	Duplicate this movie clip.
<code>.removeMovieClip</code> ();	Remove a duplicated movie clip.
<code>.swapDepths</code> (depth);	Swap the depth level with the movie currently at depth level.
<code>.swapDepths</code> (target);	Swap the depth level with the movie specified by target.
<code>.hitTest</code> (x, y, shapeFlag);	Detect collision between the movie clip and a point. ShapeFlag = true for evaluating the exact shape
<code>.hitTest</code> (target);	Detect collision between the movie clip and another clip. Bounding box is used.
<code>.getBounds</code> (targetCoordinateSpace);	Return object {xMin, xMax, yMin, yMax} of the MovieClip for the target coordinate space.
<code>.globalToLocal</code> (point);	Convert the point object from movie clip to Stage coords.
<code>.localToGlobal</code> (point);	Convert the point object from Stage to movie clip coords.
<code>.getBytesLoaded</code> ();	Returns the bytes loaded of the movie clip.
<code>.getBytesTotal</code> ();	Returns the total number of bytes in the movie clip.
<code>.getURL</code> (url, window, method);	Navigate to an URL, sending variables (optional).
<code>.gotoAndPlay</code> (frame);	Go to a frame of movie clip and play.
<code>.gotoAndStop</code> (frame);	Go to a frame of movie clip and stop.
<code>.loadMovie</code> (url, method);	Load a file (.swf) in a movie clip.
<code>.loadVariables</code> (url, method);	Load variables from URL in movie clip.
<code>.unloadMovie</code> ();	Unload a loaded movie clip.
<code>.nextFrame</code> ();	Go to next frame of movie clip.
<code>.prevFrame</code> ();	Go to previous frame of movie clip.
<code>.play</code> ();	Play the movie clip from the current frame.
<code>.stop</code> ();	Stop playing the movie clip.
<code>.nextFrame</code> ();	Go to next frame of movie clip.
<code>.prevFrame</code> ();	Go to previous frame of movie clip.
<code>.startDrag</code> (lockCenter, left, top, right, bottom);	Start a drag operation on a movie clip.
<code>.stopDrag</code> ();	Stop the drag operation.

Mouse

<code>Mouse.hide</code> ();	Hide the mouse pointer.
<code>Mouse.show</code> ();	Show the mouse pointer.

Number

<code>new Number</code> (number);	Create a new number object.
<code>Number.MAX_VALUE</code> ;	The largest number (1.79E+308).
<code>Number.MIN_VALUE</code> ;	The smallest number (5E-324).
<code>Number.NaN</code> ;	Not A Number.
<code>Number.NEGATIVE_INFINITY</code> ;	Negative infinity.
<code>Number.POSITIVE_INFINITY</code> ;	Positive infinity.
<code>.toString</code> (radix);	Convert number to string. Radix is used to specify the string's base (2, 8, 10, 16).
<code>.valueOf</code> ();	Return primitive value of number.

//Working with Movie Clips

```
//attach a movie clip from library
_root.attachMovie ("clipFromLibrary", "myInstance", 1);
//duplicates it
myInstance.duplicateMovieClip ("myInstance2", 2);
//swap their depths (Z-order)
myInstance.swapDepths (myInstance2);
//test to see if they intersect
wasHit = myInstance.hitTest (myInstance2);
//get one's location
myX = myInstance._x;
myY = myInstance._y;
//get movie clip's bounds relative to the stage
myBounds = _root.myInstance.getBounds (_root);
//repositions the clip if its right boundary is too high
if (myBounds.xMax > 200) {
    myInstance._x -= 20;
}
//repositions the clip if its right boundary is too high
if (myBounds.xMax > 200) {
    myInstance._x -= 20;
}
```

```
//using mouse.hide() and mouse.show() on a movie clip
//when clip is loaded on stage, mouse cursor is hidden
onClipEvent (load) {
    Mouse.hide();
}
onClipEvent (unload) {
    Mouse.show ();
}
```

```
//converting a number to a string, with radix conversion
dec_a = 128;
bin_a = dec_a.toString (2);
oct_a = dec_a.toString (8);
hex_a = dec_a.toString (16);

Result:
dec_a = 128
bin_a = "10000000"
oct_a = "200"
hex_a = "80"
```

Object

`new Object ()`; Create a new object.
`.toString ()`; Convert object to string.
`.valueOf ()`; Return value of object.

Selection

`Selection.getBeginIndex ()`; Beginning index of edit text selection (-1 if none).
`Selection.getCaretIndex ()`; Caret index of edit text selection (-1 if none).
`Selection.getEndIndex ()`; End index of edit text selection (-1 if none).
`Selection.getFocus ()`; Returns variable name of edit text field.
`Selection.setFocus (variableName)`; Set the focus on edit text field based on variable name.
`Selection.setSelection (beginIndex, endIndex)`; Set the text selection of edit text field.

Sound

`new Sound (target)`; Create a new sound object, optionally applied to target.
`.attachSound (idName)`; Attach a sound located in the library.
`.getPan ()`; Get the pan of sound.
`.getTransform (sxform)`; Get the current sound transform.
`.getVolume ()`; Get the sound volume (percentage).
`.setPan (pan)`; Set the pan of sound (-100 to 100 = left to right).
`.setTransform (sxform)`; Set the sound transform.
`.setVolume (volume)`; Set the sound volume (percentage).
`.start (secondsOffset, loop)`; Start playing the last attached sound at offset, looping.
`.stop ()`; Stop all sounds.

String

`new String (value)`; Create a new string object.
`.charAt (index)`; Return character at specified index.
`.charCodeAt (index)`; Return character code at specified index.
`.concat (string1, ..., stringN)`; Concatenate several strings to one string.
`String.fromCharCode (num1, ..., numN)`; Create string from character codes.
`.indexOf (searchString, fromIndex)`; Search string for substring, return index (-1 if none).
`.lastIndexOf (searchString, fromIndex)`; Search string for substring, backwards, return index (-1 if none).
`.length`; Return length of string.
`.slice (indexA, indexB)`; Slice string between two indices.
`.split (separator, limit)`; Split string in array of strings, based on delimiter.
`.substr (start, length)`; Return substring of string starting at index.
`.substring (indexA, indexB)`; Return substring of string, between two indices.
`.toLowerCase ()`; Convert string to lower case.
`.toUpperCase ()`; Convert string to upper case.

```
//selecting a word within a dynamic text field
//find beginning and end of keyword
startHigh = _root.myTextField.indexOf( keyword, 1 );
endHigh = startHigh + keyword.length;
//if not found, (-1) we set endHigh to -1 too
if (startHigh < 0) {
    endHigh = -1;
}
//set the focus on the text field and select the keyword
Selection.setFocus("mytext");
Selection.setSelection( starthigh, starthigh+lenhigh );
```

```
//working with the sound object
mySound = new Sound ();
mySound.attachSound ("soundclip");
mySound.setPan (50);
mySound.setVolume (75);
mySound.start (0, 1000);
```

```
//working with strings
myString = new String("This is a text string");
searchStr = "text";
len = searchStr.length;
position = myString.indexOf(searchStr, 0);
extract = myString.substr(position, len);
sliced = myString.slice(0, 4);
substr = myString.substring(0, 4);
sliced = sliced.toLowerCase();
substr = substr.toUpperCase();
bigString = extract+" "+sliced+" "+substr;
myArray = bigString.split(" ", bigString.length);
```

```
Result:
searchStr = "text"
len = 4
position = 10
extract = "text"
sliced = "this"
substr = "THIS"
bigString = "text, this, THIS"
myArray = [0:"text",
           1:"this",
           2:"THIS"]
```

XML

<code>new XML (source);</code>	Create a new XML object.
<code>.appendChild (child);</code>	Append a child to an XML object.
<code>.attributes;</code>	Associative array of XML element's attributes.
<code>.childNodes;</code>	Array of XML element's child nodes.
<code>.cloneNode (deep);</code>	Clone this node, and optionally children.
<code>.createElement (nodeType);</code>	Create a new XML element.
<code>.createTextNode (text);</code>	Create a new XML text node.
<code>.firstChild;</code>	First child of this XML node.
<code>.hasChildNodes ();</code>	Return true if XML element has child nodes.
<code>.insertBefore (newChild, beforeChild);</code>	Insert a child before another child in an XML element.
<code>.lastChild;</code>	Last child of this XML element.
<code>.load (url);</code>	Load XML into Flash from URL.
<code>.loaded;</code>	Return true if load/sendAndLoad is complete.
<code>.nextSibling;</code>	Next sibling of this XML element.
<code>.nodeName;</code>	Name of this XML element.
<code>.nodeType;</code>	Type of this node (1=element, 3=text).
<code>.nodeValue;</code>	Value (text) of this XML text node.
<code>.onLoad;</code>	Invoked when load/sendAndLoad is complete.
<code>.parentNode;</code>	parent of this XML node.
<code>.parseXML (string);</code>	Parse XML string into XML object.
<code>.previousSibling;</code>	Previous sibling of XML node.
<code>.removeNode ();</code>	Remove node from XML.
<code>.send (url, window);</code>	Send XML from Flash to an URL.
<code>.sendAndLoad (url, resultXML);</code>	Send XML from Flash to URL and load the XML result.
<code>.toString ();</code>	

XML Socket

<code>new XMLSocket ();</code>	Create a new XML socket object.
<code>.close ();</code>	Close current socket connection.
<code>.connect (host, port);</code>	Connect to specified host and port.
<code>.onClose;</code>	Invoked when connection is closed.
<code>.onConnect;</code>	Invoked when connection is established.
<code>.onXML;</code>	Invoked when receiving XML content.
<code>.send (data);</code>	Send XML to server.

```
//create a new XML object with childs and properties
myXML = new XML ("<member><username status=\"employee\"
position=\"manager\">Maverick</username><age>24</age><height
unit=\"m\">1.80</dimension></member>");

//now we read the first Node of the object
firstTag = myXML.firstChild;
firstTagName = firstTag.nodeName;
trace (firstTagName + ":");

// check to see if this Node has any children
if (firstTag.hasChildNodes() == true) {
    card = firstTag.childNodes;
    //check each child. Find name, attributes, value
    for (i=0; i<card.length; i++) {
        cName = card[i].nodeName;
        cAttr = card[i].attributes;

        //See each attribute for name and value
        cString = "";
        for (prop in cAttr) {
            aName = prop;
            aValue = cAttr[prop];
            cString += aName + ":" + aValue + ", ";
        }
        casLen = cString.length;
        if (casLen > 0) {
            cString = cString.substr( 0, casLen - 2);
            cString = " (" + cString + ")";
        }

        cValue = card[i].firstChild.nodeValue;

        //and now we write everything down
        trace ("      "+ cName+ cString+ " = "+ cValue);
    }
}
```

```
Result:
member:
  username (status:employee, position:manager) = Maverick
  age = 24
  height (unit:m) = 1.80
```