MC Tween: it saves the world.™

Introduction

MC Tween works by adding new methods and functions to existing Actionscript classes - namely, the MovieClip, Sound and TextField classes. Since you won't have to instantiate new objects or extend anything, the new methods are all one need to know in order to create new tweenings with this extension.

Please select one of the methods, functions or event handlers for an explanation of what each one does. If you're using AS2, also be sure to read the AS2 usage notes.

Core methods

Core methods are the methods that make tweening work, and eventually, stop it if needed. They are all you need to get most of the functionality out of **MC Tween.**

Tweening starting (MovieClip, Sound, TextField)	» tween()
Tweening ending (MovieClip, Sound, TextField)	» stopTween()

Shortcut methods

Shortcut methods are methods that help you do tweenings faster, easier, and keep the code more readable. They also provide ways of tweening attributes that otherwise aren't direcly available as a property, like a `MovieClip` tinting or a `Sound` volume, for example.

Opacity control (MovieClip, TextField)	» alphaTo()
Moving around (MovieClip, TextField)	» slideTo() » xSlideTo() » ySlideTo() » bezierSlideTo()
Rotating (MovieClip, TextField)	» rotateTo()
Scaling (MovieClip, TextField)	» stopTween()
Color tinting (MovieClip, TextField)	» scaleTo() » xScaleTo() » yScaleTo()
Color transforming (MovieClip)	» colorTo()
Timeline controlling (MovieClip)	» colorTransformTo()
Volume/panning control (Sound)	» volumeTo() » panTo()
Scrolling (TextField)	» scrollTo()

Shortcut methods cont.

Rounded shortcut methods are just like shortcut methods, but create an animation using rounded values only. This is useful when siliding movieclips that must always be featured on rounded x and y positions, like movieclips with pixel font content.

Tweening starting (MovieClip, Sound, TextField)	» roundedTween()
Moving around (MovieClip, TextField)	» roundedSlideTo() » roundedXSlideTo() » roundedYSlideTo() » roundedBezierSlideTo()

Flash 8 filters shortcut methods

Flash 8 filters shortcut methods are methods used to apply Flash 8 filters tweenings to an object.

Blur (MovieClip, TextField)	» blurTo() » xBlurTo() » yBlurTo() » xyBlurTo()
Glow (MovieClip, TextField)	» glowTo() » xGlowTo() » yGlowTo() » xyGlowTo()
Bevel (MovieClip, TextField)	» bevelTo() » xyBevelTo()

Auxiliary functions

Auxiliary functions are functions that work by dealing helping the user find out whether a tweening is being executed or not on an object.

Tweenings (MovieClip, Sound, TextField)	» getTweens() » isTweening()

Auxiliary methods

Auxiliary methods are methods that control existing tweenings - pausing or resuming them - or how an object deals with them - locking or unlocking an object from being tweened. is being executed or not on an object.

Tweening locking (MovieClip, Sound, TextField)	» lockTween() » unlockTween()
Tweening pausing/resuming (MovieClip, Sound, TextField)	» pauseTween() » resumeTween()

Auxiliary methods

Finally, additional events are new event handlers that can be added to objects to respond to a tweening update (once a frame) or completion (when the tweening has ended). Much like MovieClip event handlers, these additional events are directly added to the objects, with no listening or broadcasting involved.

Tweening events (MovieClip, Sound, TextField)	» onTweenUpdate » onTweenComplete

Core methods

.tween()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
MovieClip|Sound|TextField>.tween(property(ies), ending value(s) [, seconds, animation
type, delay, callback, extra1, extra2]);
```

Parameters

property(ies) : Property or properties to be tweened. This can be a single string containing one property name (as in "_x"), or an array containing a list of property names (as in ["_x", "_y"]). Properties need not be standard instance properties like "_x" or "_alpha" - they can be simple variables and refer to object-specific attributes like "counter" or "myPointer".

ending value(s) : New value or values that you want your properties to have at the end of the tween execution. This is either a numeric parameter (as in 10) or an array of numeric parameters (as in [10, 20]). The ending values count has to match the properties count - one ending value for each property.

seconds : An optional parameter indicating the time it will take for your tweening animation to be executed, in seconds. Fractional values are accepted, but zero is not. The default value is 2.

animation type : An optional parameter indicating the type of tweening you want to be executed, as a string. This indicates which of the easing equations will be used. See animation types for a reference on the available transition types. The default value is "easeOutExpo".

delay : An optional parameter indicating the time in which the animation should wait before starting, in seconds. Fractional values are accepted. With this, it's possible to create sequential animations by using several tweening methods with different delays. The default value is 0.

callback : An optional reference to a callback function that should be executed immediately after the tweening has completed. This is not a string parameter, but rather a direct reference to the function - that is, the function name, without the ending parenthesis or parameters. If parameters should be passed, wrap the function around an anonymous function (examples below). The function is executed from the tweened object's scope.

extra1 : First extra value to be used on the easing equation, if applicable. This depends on the animation type used. On "elastic" animation types, this is the amplitude and defaults to the same ammount of change on the transition value needed (for example, if there's a _x transition from 10 to 15, the default amplitude value is 5); on "back" animation types, this is the overshoot ammount and defaults to 1.70158.

extra2 : Second extra value to be used on the easing equation, if applicable. This depends on the animation type used. On "elastic" animation types, this is the period, and defaults to one third (0.3) of the total transition time (0.45 on easeInOutElastic's case).

Returns

Nothing.

Description

Method; use this to create new tweenings on any MovieClip, TextField or Sound property or attribute. This is the core of MC Tween, and it is used by all other tweening methods.

Sample

Next Page

Core methods cont.

.tween()

```
// Simple movieclip sliding in 0.5 seconds
<MovieClip>.tween("_x", 10, 0.5);

// Two different transitions
<MovieClip>.tween(["_x", "_alpha"], [10, 100], 0.5);

// Other animation methods
<MovieClip>.tween("_alpha", 0.5, "linear");

// The same applies to TextFields
<TextField>.tween("_y", 200, 0.7, "easeOutExpo");

// Using a delay to create a chain animation to make a MovieClip move right and then
left
// Notice the use of 'undefined' when a parameter should be skipped
<MovieClip.tween("_x", 20, 0.5);
<MovieClip.tween("_x", 10, 0.5, undefined, 0.5);

// Using a callback with a named function
// Remember: "this" inside the callback function refers to the tweened object itself, in
this case the <MovieClip>
myCallBack = function() {
    trace ("ok, completed");
    this.play();
};
<MovieClip>.tween("_alpha", 100, 0.5, "linear", 0, myCallBack);

// Using a callback with an anonymous (inline) function
<MovieClip>.tween("_alpha", 100, 0.5, "linear", 0, function() {
    trace ("ok, completed);
    this.play();
});

// Using an anonymous function to wrap around a function with parameters
<MovieClip>.tween("_alpha", 100, 0.5, "linear", 0, function() { this.gotoAndPlay(25);
});
```

See Also

roundedTween()

Core methods

.stopTween()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|Sound|TextField>.stopTween([property(ies), property, ...);
```

Parameters

property(ies) : Property or properties to be stopped. This can be a single string containing one property name (as in "_x"), an array containing a list of property names (as in ["_x", "_y"]), or a list of properties as parameters (as in "_x", "_y"). It can be omitted, too; in this case, all tweenings for that particular object will be stopped.

Returns

Nothing.

Description

Method; removes tweenings from a particular object. All properties being tweened remain as-is, with the values they had when the method was called.

Sample

```
// Stops an _alpha tweening that's being executed on a TextField
<TextField>.stopTween("__alpha");

// Stops an _x and and _y tweening on a MovieClip
<MovieClip>.stopTween(["_x", "_y"]);

// Also stops an _x and and _y tweening on a MovieClip
<MovieClip>.stopTween("_x", "_y");

// Stops all tweenings being done on a Sound object
<Sound>.stopTween();
```

See Also

pauseTween(), resumeTween()

Shortcut methods

.alphaTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6

Usage

```
<MovieClip|TextField>.alphaTo(opacity [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

opacity : desired `_alpha` value, usually from 0 to 100.

All other parameters are standard `tween()` related. Refer to the .tween() method for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_alpha` property of a `MovieClip` or `TextField` - a fade in or a fade out, in practical words.

Sample

```
// Makes a Movieclip appear from out of nothing, taking one second and using linear transition
<MovieClip>_alpha = 0;
<MovieClip>.alphaTo(100, 1, "linear");

// Makes a MovieClip fade out on 0.8 seconds and then turns it off for better Flash Player performance
<MovieClip>.alphaTo(0, 0.8, "linear", 0, function() { this._visible = false; });
```

Shortcut methods

.bezierSlideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.bezierSlideTo(control point x, control point y, x, y [, seconds, anima-
tion type, delay, callback, extra1, extra2]);
```

Parameters

control point x : x position of the desired control point

control point y : y position of the desired control point

x : Desired `_x` position for this object.

y : Desired `_y` position for this object.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_x` and `_y` properties of a `MovieClip` or `TextField`, sliding it on screen, but using a curve as its path. The curve is a simple curve with one bezier control point (BCP); the syntax and the concept is similar to actionScript's MovieClip.curveTo() drawing method.

Sample

```
// Moves a MovieClip from its current position to 200,200, with the bezier control point
at 100,100, on 0.5 seconds
<MovieClip>.bezierSlideTo(100, 100, 200, 200, 0.5);
```

See Also

roundedBezierSlideTo(), slideTo(), MovieClip.curveTo()

Shortcut methods

.colorTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.colorTo(color [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

color : Desired color for tinting, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. If the value on this parameter is `null`, it removes whichever tinting the object has (`MovieClips` only).

All other parameters are standard `tween()` related. Refer to the .tween() command for reference.

Returns

Nothing.

Description

Shortcut method; does a tinting transition on a MovieClip or set a TextField's textColor parameter. The object's _alpha value is not changed when this method is executed.

Sample

```
// Tints a MovieClip to #ffg1b2 on 0.5 seconds using a linear transition
<MovieClip>.colorTo(0xffg1b2, 0.5, "linear");

// Removes all tinting aplied to a MovieClip in 1 second
<MovieClip>.colorTo(null, 1);

// Changes a TextField's color to #ff0023 in 2 seconds
<TextField>.colorTo(0xff0023, 2);
```

See Also

colorTransformTo(), Color.setRGB()

Shortcut methods

.colorTransformTo()

Applies To

`MovieClip`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip>.colorTransformTo(ra, rb, ga, gb, ba, bb, aa, ab [, seconds, animation type, delay,
callback, extra1, extra2]);

or

<MovieClip>.colorTransformTo(colorTransformObject [, seconds, animation type, delay, callback,
extra1, extra2]);
```

Parameters

ra : Percentage desired for the R (red) color channel, usually from 0 to 100 (values from -1000 to 1000 are accepted). This parameter informs how much of the original channel will be maintained. The normal value for a static, non-colored MovieClip would be 100.

rb : The offset of tinting for the red channel, usually from 0 to 255 (values from -512 to 512 are accepted). This is added to the channel after the percentage transformation has taken place. The normal value for a static, non-colored MovieClip would be 0.

ga : Percentage desired for the G (green) color channel, usually from 0 to 100 (values from -1000 to 1000 are accepted). This parameter informs how much of the original channel will be maintained. The normal value for a static, non-colored MovieClip would be 100.

gb : The offset of tinting for the green channel, usually from 0 to 255 (values from -512 to 512 are accepted). This is added to the channel after the percentage transformation has taken place. The normal value for a static, non-colored MovieClip would be 0.

ba : Percentage desired for the B (blue) color channel, usually from 0 to 100 (values from -1000 to 1000 are accepted). This parameter informs how much of the original channel will be maintained. The normal value for a static, non-colored MovieClip would be 100.

bb : The offset of tinting for the blue channel, usually from 0 to 255 (values from -512 to 512 are accepted). This is added to the channel after the percentage transformation has taken place. The normal value for a static, non-colored MovieClip would be 0.

aa : Percentage desired for the A (alpha) channel, usually from 0 to 100 (values from -1000 to 1000 are accepted). This parameter informs how much opacity the object will have - it's just the same as the MovieClip's _alpha property, really. The normal value for a static, non-colored MovieClip would be 100.

ab : The offset opacity for the object's transparency. This is a bit tricky, but what it does is add more opacity when needed, even on areas where a MovieClip object is totally transparent. The normal value for a static, non-colored MovieClip would be 0.

colorTransformObject : This is a different way of doing the same thing. In this case, you can use color transformation objects (like you would with Color.setTransform()) to pass the parameters. The colorTransformObject would then be an object with properties called ra, rb, ga, gb, ba, bb, aa, and ab.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Also notice that all color parameters are optional - use undefined when a parameter has to be skipped. If a parameter value is not defined, it will not be tweened.

Returns

Nothing.

Continued Next Page

Shortcut methods

.colorTransformTo()

Continued

Description

Shortcut method; does a tinting transition on a MovieClip or set a TextField's textColor parameter. The object's _alpha value is not changed when this method is executed.

Sample

```
// Creates a "dodge" look on a MovieClip in 2 seconds, using a "linear" transition
<MovieClip>.colorTransformTo(200, 0, 200, 0, 200, 0, undefined, undefined, 2, "linear");

// Tweens just the red channel of a MovieClip, in 0.6 seconds
<MovieClip>.colorTransformTo(200, undefined, undefined, undefined, undefined, undefined, un-
defined, undefined, 0.6, "linear");

// Inverts the MovieClips color altogether, in 2 seconds
<MovieClip>.colorTransformTo(-100, 256, -100, 256, -100, 256, 100, 0, 2, "linear");

// Does an alpha transition on a MovieClip - this is equivalent to alphaTo(100, 1)
<MovieClip>.colorTransformTo(undefined, undefined, undefined, undefined, undefined, undefined,
100, undefined, 1);

// Resets a MovieClip to its normal state, on 0.5 seconds
<MovieClip>.colorTransformTo(100, 0, 100, 0, 100, 0, 100, 0, 0.5);

// The same as above, but using a colorTransformObject
myCTO = new Objec();
myCTO.ra = 100;
myCTO.rb = 0;
myCTO.ga = 100;
myCTO.gb = 0;
myCTO.ba = 100;
myCTO.bb = 0;
myCTO.aa = 100;
myCTO.ab = 0;
<MovieClip>.colorTransformTo(myCTO, 0.5);

// The same as above, but using an in-line colorTransformObject
<MovieClip>.colorTransformTo({ra:100, rb:0, ga:100, gb:0, ba:100, bb:0, aa:100, ab:0},
0.5);
```

See Also

colorTransformTo(), Color.setRGB()

Shortcut methods

.frameTo()

Applies To

MovieClip

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip>.frameTo(frame [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

frame : Frame that the movieclip playhead must be moved to.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a playback transition on a MovieClip, moving its playhead based on the animation type selected. This is useful to create different animation effects based on a timeline animation, usually a keyframe animation.

Also, a warning: while this works nicely to create frame seeking, it's not recommended if you're trying to create playback controls like backwards playing. In this case, it's faster and more accurate to use some simple onEnterFrame scripts to move the playhead back one frame per pass. Also, since frameTo() is based on an arbitrary execution time, it will skip and repeat frames as necessary. If you're building a timeline animation to be used with this method, try creating it with as many frames as possible for better playback.

Sample

```
// Rewinds a MovieClip to frame 1 in 0.5 seconds
<MovieClip>.frameTo(1, 0.5);

// Playbacks a Movieclip from start to finish in exactly 2 seconds
<MovieClip>.gotoAndStop(1);
<MovieClip>.frameTo(<MovieClip>._totalframes, 2, "linear");
```

Shortcut methods

.panTo()

Applies To

`Sound`

Availability

Flash 6 and above, using AS1.

Usage

```
<Sound>.panTo(panning [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

panning : Desired panning value. This ranges from -100 to 100; -100 concentrates all sound on the left speaker, 100 concentrates on the right speaker, and 0 is the normal setting, using both speakers.

All other parameters are standard `tween()` related. Refer to the `.tween()` method for reference.

Returns

Nothing.

Description

Shortcut method; does a sound panning on a `Sound` object, going to an specific panning setting. This the tweening equivalent of using `Sound.setPan()`, and can be used to emulate moving/locational audio in Flash.

Sample

```
// Makes a movie sound like something is moving from the left to the right sideof the screen
<Sound>.setPan(-100);
<Sound>.panTo(100);
```

See Also

Sound.setPan()

Shortcut methods

.rotateTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.rotateTo(rotation [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

rotation : Desired rotation value, in degrees.

All other parameters are standard `tween()` related. Refer to the `.tween()` method for reference.

Returns

Nothing.

Description

Shortcut method; rotates the `TextField` or `MovieClip` object to an specific angle, using its `_rotation` property.

Sample

```
// Rotates a MovieClip 180 degrees (to make it stand upside-down) using an easeInOutBack animation for 2 seconds
<MovieClip>.rotateTo(180, 2, "easeinoutback");
```

Shortcut methods

.scaleTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.scaleTo(scale [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

scale : Desired scale value for the object in percent (100 being the standard size).

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on both the `_xscale` and `_yscale` properties of a `MovieClip` or `TextField` at the same time, resizing it based on its original control point.

Sample

```
// Scales a MovieClip to 200%, in one second, using an elastic easing animation
<MovieClip>.scaleTo(200, 1, "easeoutelastic");
```

See Also

xScaleTo(), yScaleTo()

Shortcut methods

.scrollTo()

Applies To

`TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<TextField>.scrollTo(line [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

line : Desired line that the textfield must scroll to.

All other parameters are standard `tween()` related. Refer to the `.tween()` method for reference.

Returns

Nothing.

Description

Shortcut method; scrolls the TextField to an specific line. The line should be between 1 and the value from the TextField's .max-scroll property, inclusive. When using this method, remember that the .scroll property - the one that is used by Flash to control the TextField's current line - should always be an integer. Because of this, scrolling doesn't work on a per-pixel basis but on a per-line basis, so animations done on this property may look odd depending on the TextField's number of lines, size, and font size.

Sample

```
// Scroll a textfield to line 100 in 0.5 seconds using a linear transition
<TextField>.scrollTo(100, 0.5, "linear");
```

Shortcut methods

.slideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.slideTo(x, y [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

x : Desired `_x` position for this object. Can be omitted (use `undefined`) but then the `y` parameter is mandatory. If omitted, no `_x` transition occurs.

y : Desired `_y` position for this object. Can be omitted (use `undefined`) but then the `x` parameter is mandatory. If omitted, no `_y` transition occurs.

All other parameters are standard tween() related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_x` and `_y` properties of a `MovieClip` or `TextField`, sliding it on screen.

Sample

```
// Moves a MovieClip to 10, 10 in 0.5 seconds
<MovieClip>.slideTo(10, 10, 0.5);

// Moves a TextField to _y 20 in 1 second, using a linear transition. The TextField's _x position is not changed.
<TextField>.slideTo(undefined, 20, 1, "linear");
```

See Also

roundedSlideTo(), bezierSlideTo(), xSlideTo(), ySlideTo()

Shortcut methods

.volumeTo()

Applies To

`Sound`

Availability

Flash 6 and above, using AS1

Usage

```
<Sound>.volumeTo(volume [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

volume : Desired volume, in percent. Usually from 0 to 100.

All other parameters are standard `tween()` related. Refer to the `.tween()` method for reference.

Returns

Nothing.

Description

Shortcut method; does a sound fade on a `Sound` object, going to an specific volume setting. This the tweening equivalent of using `Sound.setVolume()`.

Sample

```
// Turns off a sound object by fading it out for 0.5 seconds, using a linear transition
<Sound>.volumeTo(0, 0.5, "linear");
```

See Also

Sound.setVolume()

Shortcut methods

.xScaleTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.xScaleTo(scale [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

scale : Desired scale value for the object's `_xscale` attribute, in percent (100 being the standard size).

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_xscale` property of a `MovieClip` or `TextField` at the same time, resizing it based on its original control point.

Sample

```
// Scales a MovieClip horizontally to 150%, in two seconds, using an easeOutExpo transition
<MovieClip>.xScaleTo(150, 2, "easeoutexpo");
```

See Also

scaleTo(), yScaleTo()

Shortcut methods

.xSlideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.xSlideTo(x [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

x : Desired _x position for this object.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_x` property of a `MovieClip` or `TextField`, sliding it on screen horizontally.

Sample

```
// Moves a MovieClip to _x=10.
<MovieClip>.xSlideTo(10, 0.5);
```

See Also

scaleTo(), yScaleTo()

Shortcut methods

.yScaleTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.yScaleTo(scale [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

scale : Desired scale value for the object's `_yscale` attribute, in percent (100 being the standard size).

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_yscale` property of a `MovieClip` or `TextField` at the same time, resizing it based on its original control point.

Sample

```
// Scales a MovieClip vertically to 250%, in one seconds, using a linear transition
<MovieClip>.xScaleTo(250, 1, "linear");
```

See Also

scaleTo(), xScaleTo()

Shortcut methods

.ySlideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.ySlideTo(y [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

y : Desired `_y` position for this object.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_y` property of a `MovieClip` or `TextField`, sliding it on screen.

Sample

```
// Moves a MovieClip to _y=20.
<MovieClip>.ySlideTo(20, 0.5);
```

See Also

roundedYSlideTo(), slideTo(), xSlideTo()

Rounded Shortcut Methods

.roundedBezierSlideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.bezierSlideTo(control point x, control point y, x, y [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

control point x : `x` position of the desired control point

control point y : `y` position of the desired control point

x : Desired `_x` position for this object.

y : Desired `_y` position for this object.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_x` and `_y` properties of a `MovieClip` or `TextField`, sliding it on screen, but using a curve as its path (like the bezierSlideTo() method), but using only rounded positions when updating. This is useful when tweening "pixel" text or other type of pixel-aligned content. The curve is a simple curve with one bezier control point (BCP); the syntax and the concept is similar to actionScript's `MovieClip.curveTo()` drawing method.

Sample

```
// Moves a MovieClip from its current position to 200,200, with the bezier control point at 100,100, on 0.5 seconds
<MovieClip>.roundedBezierSlideTo(100, 100, 200, 200, 0.5);
```

See Also

bezierSlideTo(), roundedSlideTo(), MovieClip.curveTo()

Rounded Shortcut Methods

.roundedSlideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.roundedSlideTo(x, y [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

x : Desired `_x` position for this object. Can be omitted (use `undefined`) but then the `y` parameter is mandatory. If omitted, no `_x` transition occurs.

y : Desired `_y` position for this object. Can be omitted (use `undefined`) but then the `x` parameter is mandatory. If omitted, no `_y` transition occurs.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_x` and `_y` properties of a `MovieClip` or `TextField`, sliding it on screen (like the `slideTo()` method), but using only rounded positions when updating. This is useful when tweening "pixel" text or other type of pixel-aligned content.

Sample

```
// Moves a MovieClip to 10, 10 in 0.5 seconds
<MovieClip>.roundedSlideTo(10, 10, 0.5);

// Moves a TextField to _y 20 in 1 second, using a linear transition. The TextField's _x position is not changed.
<TextField>.roundedSlideTo(undefined, 20, 1, "linear");
```

See Also

slideTo(), roundedBezierSlideTo(), roundedXSlideTo(), roundedYSlideTo()

Rounded Shortcut Methods

.roundedBezierSlideTo()

Applies To
`MovieClip, TextField`

Availability
Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|Sound|TextField>.roundedTween(property(ies), ending value(s) [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

property(ies) : Property or properties to be tweened. This can be a single string containing one property name (as in "_x"), or an array containing a list of property names (as in ["_x", "_y"]). Properties need not be standard instance properties like "_x" or "_alpha" - they can be simple variables and refer to object-specific attributes like "counter" or "myPointer".

ending value(s) : New value or values that you want your properties to have at the end of the tween execution. This is either a numeric parameter (as in 10) or an array of numeric parameters (as in [10, 20]). The ending values count has to match the properties count - one ending value for each property.

seconds : An optional parameter indicating the time it will take for your tweening animation to be executed, in seconds. Fractional values are accepted, but zero is not. The default value is 2.

animation type : An optional parameter indicating the type of tweening you want to be executed, as a string. This indicates which of the easing equations will be used. See animation types for a reference on the available transition types. The default value is "`easeOutExpo`".

delay : An optional parameter indicating the time in which the animation should wait before starting, in seconds. Fractional values are accepted. With this, it's possible to create sequential animations by using several tweening methods with different delays. The default value is 0.

callback : An optional reference to a callback function that should be executed immediately after the tweening has completed. This is not a string parameter, but rather a direct reference to the function - that is, the function name, without the ending parenthesis or parameters. If parameters should be passed, wrap the function around an anonymous function (examples below). The function is executed from the tweened object's scope.

extra1 : First extra value to be used on the easing equation, if applicable. This depends on the animation type used. On "elastic" animation types, this is the amplitude and defaults to the same ammount of change on the transition value needed (for example, if there's a _x transition from 10 to 15, the default amplitude value is 5); on "back" animation types, this is the overshoot ammount and defaults to 1.70158.

extra2 : Second extra value to be used on the easing equation, if applicable. This depends on the animation type used. On "elastic" animation types, this is the period, and defaults to one third (0.3) of the total transition time (0.45 on easeInOutElastic's case).

Returns
Nothing.

Description
Method; use this to create new tweenings on any `MovieClip`, `TextField` or Sound property or attribute. This works just like the tween() method, but using only rounded positions when updating. This is useful when tweening "pixel" text or other type of pixel-aligned content.

Sample

```
// Simple movieclip sliding in 0.5 seconds
<MovieClip>.roundedTween("_x", 10, 0.5);

// Two different transitions
<MovieClip>.roundedTween(["_x", "_alpha"], [10, 100], 0.5);
```

See Also
bezierSlideTo(), roundedSlideTo(), MovieClip.curveTo()

Rounded Shortcut Methods

.roundedXSlideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.roundedXSlideTo(x [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

x : Desired `_x` position for this object.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_x` property of a `MovieClip` or `TextField`, sliding it on screen horizontally (like the xSlideTo() method), but using only rounded positions when updating. This is useful when tweening "pixel" text or other type of pixel-aligned content.

Sample

```
// Moves a MovieClip to _x=10.
<MovieClip>.roundedXSlideTo(10, 0.5);
```

See Also

xSlideTo(), roundedSlideTo(), roundedYSlideTo()

Rounded Shortcut Methods

.roundedYSlideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.roundedYSlideTo(y [, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

x : Desired `_y` position for this object.

All other parameters are standard tween() related. Refer to the .tween() command for reference.

Returns

Nothing.

Description

Shortcut method; does a transition on the `_x` and `_y` properties of a `MovieClip` or `TextField`, sliding it on screen, but using a curve as its path (like the bezierSlideTo() method), but using only rounded positions when updating. This is useful when tweening "pixel" text or other type of pixel-aligned content. The curve is a simple curve with one bezier control point (BCP); the syntax and the concept is similar to actionScript's `MovieClip.curveTo()` drawing method.

Sample

```
// Moves a MovieClip to _y=20.
<MovieClip>.roundedYSlideTo(10, 0.5);
```

See Also

ySlideTo(), roundedSlideTo(), roundedXSlideTo()

Flash 8 Filters Shortcut Methods

.bevelTo()

Applies To

`MovieClip, TextField`

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.bevelTo(BevelFilter [, seconds, animation type, delay, callback, extra1, extra2]);

or

<MovieClip|TextField>.bevelTo([distance, angle, highlightColor, highlightAlpha, shadowColor, shadowAlpha, blurRadius, strength, quality, type, knockout, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

BevelFilter : A flash.filters.BevelFilter object that will be applied to this object's filters array.

distance : Desired bevel distance (acts like the "height" of the bevel). This is equivalent to this filter's distance parameter.

angle : Desired bevel angle, in degrees. Default is 45. This is equivalent to this filter's angle parameter.

highlightColor : Desired light color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's highlightColor parameter.

highlightAlpha : Desired light alpha. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's highlightAlpha parameter.

shadowColor : Desired shadow color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's shadowColor parameter.

shadowAlpha : Desired shadow alpha. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's shadowAlpha parameter.

blurRadius : A numeric value indicating the radius of the desired bevel blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurX and blurY properties.

strength : Desired strength for this filter. This value usually ranges from 0 to 1 (default) and beyond. This is equivalent to this filter's strength parameter.

quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If omitted, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.

type : The type of bevel to apply. This is a string with either "inner" (default), "outer" or "full" as values. This is equivalent to this filter's type parameter.

knockout : A boolean value indicating whether or not this filter knocks out its object, hiding the original content and showing the filter instead. This is equivalent to this filter's knockout parameter.

All other parameters are standard `tween()` related. Refer to the .tween() command for reference.

Returns

Nothing.

Continued Next Page

Flash 8 Filters Shortcut Methods

.bevelTo()

Continued

Returns

Nothing.

Description

Shortcut method; applies a Bevel filter to an existing object, tweening its properties.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Examples

```
// Creates a very simple Bevel on a movieclip on 2 seconds, with a distance of 10 and
using default values, using in-line commands
<MovieClip>.bevelTo(10, undefined, undefined, undefined, undefined, undefined, un-
defined, undefined, undefined, undefined, 2);

// Creates a blurry bevel and applies it to a movieclip using a linear animation, using
a Bevel filter object
var bvl = new flash.filters.BevelFilter(20, 45, 0xffffff, 0.5, 0x000000, 0.5, 10, 10, 1,
2, "inner", false);
<MovieClip>.bevelTo(bvl, 2, "linear");
```

See Also

xyBevelTo(), flash.filters.BevelFilter

Flash 8 Filters Shortcut Methods

.roundedYSlideTo()

Applies To

`MovieClip, TextField`

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.blurTo(BlurFilter [, seconds, animation type, delay, callback, extra1,
extra2]);

or

<MovieClip|TextField>.blurTo(blurRadius [, quality, seconds, animation type, delay, callback,
extra1, extra2]);
```

Parameters

BlurFilter : A flash.filters.BlurFilter object that will be applied to this object's filters array.

blurRadius : A numeric value indicating the radius of the desired blur. The bigger this value, the more blurred the object will look. This is equivalent to this filter's blurX and blurY properties.

quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If ommited, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; applies a Blur filter to an existing object, tweening its properties.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Creates a filter template and applies it, tweening in 2 seconds with a liner transfor-
mation
var myBlurry = new flash.filters.BlurFilter(0, 0, 4);
<MovieClip>.blurTo(myBlurry, 2, "linear");

// Adds a blur to a textfield, using 20 pixels of blur, using "medium" quality, and in 1
second
<TextField>.blurTo(20, 2, 1);
```

See Also

xyBlurTo(), xBlurTo(), yBlurTo(), flash.filters.BlurFilter

Flash 8 Filters Shortcut Methods

.roundedYSlideTo()

Applies To

MovieClip, TextField

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.glowTo(GlowFilter [, seconds, animation type, delay, callback, extra1,
extra2]);
or
<MovieClip|TextField>.glowTo([color, alpha, blurRadius, strength, quality, inner, knockout,
seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

GlowFilter : A `flash.filters.GlowFilter` object that will be applied to this object's filters array.
color : Desired glow color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's color parameter.
alpha : Desired alpha for this glow. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's alpha parameter.
blurRadius : A numeric value indicating the radius of the desired glow blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurX and blurY properties.
strength : Desired strength for this filter. This value usually ranges from 0 to 1 (default) and beyond. This is equivalent to this filter's strength parameter.
quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If omitted, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.
inner : A boolean value indicating if this is an inner glow instead of the default outer glow. This is equivalent to this filter's inner parameter.
knockout : A boolean value indicating whether or not this filter knocks out its object, hiding the original content and showing the filter instead. This is equivalent to this filter's knockout parameter.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; applies a Glow filter to an existing object, tweening its properties.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Creates a red aura around a MovieClip on 2 seconds, with a glow filter object
var myAura = new flash.filters.GlowFilter(0xff0000, 1, 20, 20, 1, 2, false, false);
<MovieClip>.glowTo(myAura, 2);

// Creates a red aura around a MovieClip on 2 seconds, with an in-line command
<MovieClip>.glowTo(0xff0000, 1, 20, 1, 2, false, false, 2);
```

See Also

xyGlowTo(), xGlowTo(), yGlowTo(), flash.filters.GlowFilter

Flash 8 Filters Shortcut Methods

.xBlurTo()

Applies To

`MovieClip, TextField`

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.xBlurTo(blurX [, quality, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

blurX : Desired horizontal radius of the blur. The bigger this value, the more blurred the object will look. This is equivalent to this filter's blurX property.

quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If ommited, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; applies a horizontal Blur filter to an existing object, tweening its properties.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Adds a horizontal blur of 2 pixels to a textfield (making it difficult to read), using "high" quality, in 1 second
<TextField>.xBlurTo(2, 3, 1);
```

See Also

blurTo(), xyBlurTo(), yBlurTo(), flash.filters.BlurFilter

Flash 8 Filters Shortcut Methods

.xBlurTo()

Applies To

MovieClip, TextField

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.xGlowTo([color, alpha, blurX, strength, quality, inner, knockout, sec-
onds, animation type, delay, callback, extra1, extra2]);
```

Parameters

BlurFilter : A flash.filters.GlowFilter object that will be applied to this object's filters array.
color : Desired glow color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's color parameter.
alpha : Desired alpha for this glow. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's alpha parameter.
blurX : Desired horizontal radius of the glow blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurX property.
strength : Desired strength for this filter. This value usually ranges from 0 to 1 (default) and beyond. This is equivalent to this filter's strength parameter.
quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If ommited, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.
inner : A boolean value indicating if this is an inner glow instead of the default outer glow. This is equivalent to this filter's inner parameter.
knockout : A boolean value indicating whether or not this filter knocks out its object, hiding the original content and showing the filter instead. This is equivalent to this filter's knockout parameter.

All other parameters are standard tween() related. Refer to the .tween() command for reference.

Returns

Nothing.

Description

Shortcut method; applies a horizontal Glow filter to an existing object, tweening its properties.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Creates a blue horizontal aura around a MovieClip on 3 seconds
<MovieClip>.glowTo(0x0000ff, 1, 20, 1, 2, false, false, 3);
```

See Also

glowTo(), xyGlowTo(), yGlowTo(), flash.filters.GlowFilter

Flash 8 Filters Shortcut Methods

.xBlurTo()

Applies To

`MovieClip, TextField`

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.xGlowTo([color, alpha, blurX, strength, quality, inner, knockout, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

BlurFilter : A flash.filters.GlowFilter object that will be applied to this object's filters array.
color : Desired glow color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's color parameter.
alpha : Desired alpha for this glow. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's alpha parameter.
blurX : Desired horizontal radius of the glow blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurX property.
strength : Desired strength for this filter. This value usually ranges from 0 to 1 (default) and beyond. This is equivalent to this filter's strength parameter.
quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If ommited, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.
inner : A boolean value indicating if this is an inner glow instead of the default outer glow. This is equivalent to this filter's inner parameter.
knockout : A boolean value indicating whether or not this filter knocks out its object, hiding the original content and showing the filter instead. This is equivalent to this filter's knockout parameter.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; applies a horizontal Glow filter to an existing object, tweening its properties.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Creates a blue horizontal aura around a MovieClip on 3 seconds
<MovieClip>.xGlowTo(0x0000ff, 1, 20, 1, 2, false, false, 3);
```

See Also

glowTo(), xyGlowTo(), yGlowTo(), flash.filters.GlowFilter

Flash 8 Filters Shortcut Methods

.yBlurTo()

Applies To

`MovieClip, TextField`

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.yBlurTo(blurY [, quality, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

blurY : Desired vertical radius of the blur. The bigger this value, the more blurred the object will look. This is equivalent to this filter's blurY property.

quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If ommited, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.

All other parameters are standard tween() related. Refer to the .tween() command for reference.

Returns

Nothing.

Description

Shortcut method; applies a vertical Blur filter to an existing object, tweening its properties.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Adds a vertical blur of 10 pixels to a movieclip, using "high" quality, in 0.5 seconds
<MovieClip>.yBlurTo(10, 3, 0.5);
```

See Also

blurTo(), xyBlurTo(), xBlurTo(), flash.filters.BlurFilter

Flash 8 Filters Shortcut Methods

.yGlowTo()

Applies To

MovieClip, TextField

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.yGlowTo([color, alpha, blurY, strength, quality, inner, knockout, seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

BlurFilter : A flash.filters.GlowFilter object that will be applied to this object's filters array.
color : Desired glow color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's color parameter.
alpha : Desired alpha for this glow. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's alpha parameter.
blurY : Desired vertical radius of the glow blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurY property.
strength : Desired strength for this filter. This value usually ranges from 0 to 1 (default) and beyond. This is equivalent to this filter's strength parameter.
quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If ommited, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.
inner : A boolean value indicating if this is an inner glow instead of the default outer glow. This is equivalent to this filter's inner parameter.
knockout : A boolean value indicating whether or not this filter knocks out its object, hiding the original content and showing the filter instead. This is equivalent to this filter's knockout parameter.

All other parameters are standard tween() related. Refer to the .tween() command for reference.

Returns

Nothing.

Description

Shortcut method; applies a horizontal Glow filter to an existing object, tweening its properties.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Creates a green vertical aura around a MovieClip on 3 seconds
<MovieClip>.glowTo(0x00ff00, 1, 20, 1, 2, false, false, 3);
```

See Also

glowTo(), xyGlowTo(), yGlowTo(), flash.filters.GlowFilter

Flash 8 Filters Shortcut Methods

.xyBevelTo()

Applies To

MovieClip, TextField

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.xyBevelTo([distance, angle, highlightColor, highlightAlpha, shadowColor,
shadowAlpha, blurX, blurY, strength, quality, type, knockout, seconds, animation type, delay,
callback, extra1, extra2]);
```

Parameters

BevelFilter : A flash.filters.BevelFilter object that will be applied to this object's filters array.
distance : Desired bevel distance (acts like the "height" of the bevel). This is equivalent to this filter's distance parameter.
angle : Desired bevel angle, in degrees. Default is 45. This is equivalent to this filter's angle parameter.
highlightColor : Desired light color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's highlightColor parameter.
highlightAlpha : Desired light alpha. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's highlightAlpha parameter.
shadowColor : Desired shadow color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's shadowColor parameter.
shadowAlpha : Desired shadow alpha. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's shadowAlpha parameter.
blurX : Desired horizontal radius of the glow blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurX property.
blurY : Desired vertical radius of the glow blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurY property.
strength : Desired strength for this filter. This value usually ranges from 0 to 1 (default) and beyond. This is equivalent to this filter's strength parameter.
quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If omitted, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.
type : The type of bevel to apply. This is a string with either "inner" (default), "outer" or "full" as values. This is equivalent to this filter's type parameter.
knockout : A boolean value indicating whether or not this filter knocks out its object, hiding the original content and showing the filter instead. This is equivalent to this filter's knockout parameter.

All other parameters are standard tween() related. Refer to the .tween() command for reference.

Returns

Nothing.

Description

Shortcut method; applies a Bevel filter to an existing object, tweening its properties, with different vertical and horizontal blurring radius.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Creates a Bevel on a movieclip on 2 seconds, with a horizontal blur of 20 and verti-
cal blur of 10
<MovieClip>.bevelTo(20, 45, 0xffffff, 0.5, 0x000000, 0.5, 20, 10, 1, 2, "inner", false,
2);
```

See Also

glowTo(), xyGlowTo(), yGlowTo(), flash.filters.GlowFilter

Flash 8 Filters Shortcut Methods

.xyBlurTo()

Applies To

MovieClip, TextField

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.xyBlurTo([blurX, blurY, quality, seconds, animation type, delay, call-
back, extra1, extra2]);
```

Parameters

blurX : Desired horizontal radius of the blur. The bigger this value, the more blurred the object will look. This can be omitted (use undefined) but then the blurY parameter is mandatory. This is equivalent to this filter's blurX property.

blurY : Desired vertical radius of the blur. The bigger this value, the more blurred the object will look. This can be omitted (use undefined) but then the blurY parameter is mandatory. This is equivalent to this filter's blurY property.

quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If omitted, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.

All other parameters are standard tween() related. Refer to the .tween() command for reference.

Returns

Nothing.

Description

Shortcut method; applies a Blur filter to an existing object, tweening its properties, with different vertical and horizontal blurring radius.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Adds a kind of a motion blur to a MovieClip, using 20 pixels of horizontal blur,
// 3 pixels of vertical blur, "medium" quality, and in 1 second, with a "easeoutback"
animation
<MovieClip>.xyBlurTo(20, 3, 2, 1, "easeoutback");
```

See Also

blurTo(), xBlurTo(), yBlurTo(), flash.filters.BlurFilter

Flash 8 Filters Shortcut Methods

.xyGlowTo()

Applies To

`MovieClip, TextField`

Availability

Flash 8 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.yGlowTo([color, alpha, blurX, blurY, strength, quality, inner, knockout,
seconds, animation type, delay, callback, extra1, extra2]);
```

Parameters

BlurFilter : A flash.filters.GlowFilter object that will be applied to this object's filters array.

color : Desired glow color, in numeric value. This numeric value works similarly to the Color.setRGB color parameter. This is equivalent to this filter's color parameter.

alpha : Desired alpha for this glow. Notice, though, that this value ranges from 0 (transparent) to 1 (opaque) instead of from 0 to 100. This is equivalent to this filter's alpha parameter.

blurX : Desired horizontal radius of the glow blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurX property.

blurY : Desired vertical radius of the glow blur. The bigger this value, the more blurred the glow it will look. This is equivalent to this filter's blurY property.

strength : Desired strength for this filter. This value usually ranges from 0 to 1 (default) and beyond. This is equivalent to this filter's strength parameter.

quality : A numeric value indicating the quality of this blur, usually from 1 ("Low") to 3 ("High"). It's, roughly, the number of times this filter is applied - the higher the number, the better the quality, but also the higher the time it takes to be applied. If omitted, this value defaults to 2. This value is not tweenable. This is equivalent to this filter's quality property.

inner : A boolean value indicating if this is an inner glow instead of the default outer glow. This is equivalent to this filter's inner parameter.

knockout : A boolean value indicating whether or not this filter knocks out its object, hiding the original content and showing the filter instead. This is equivalent to this filter's knockout parameter.

All other parameters are standard `tween()` related. Refer to the `.tween()` command for reference.

Returns

Nothing.

Description

Shortcut method; applies a Blur filter to an existing object, tweening its properties, with different vertical and horizontal blurring radius.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Creates a green vertical aura around a MovieClip on 3 seconds,
// with 20 of horizontal glow and 10 of vertical glow
<MovieClip>.glowTo(0x00ff00, 1, 20, 10, 1, 2, false, false, 3);
```

See Also

glowTo(), xGlowTo(), yGlowTo(), flash.filters.GlowFilter

Auxiliary Functions

.getTweens()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|Sound|TextField>.getTweens()
```

Parameters

None.

Returns

The number of tweenings being performed by the object in that moment. Future tweenings (created with delays) are also counted.

Description

Shortcut method; applies a Bevel filter to an existing object, tweening its properties, with different vertical and horizontal blurring radius.

If the object doesn't have any filter of this type applied, it creates a new filter with default values and then tweens to the new one. If the object already has a filter of this type applied, it will tween the existing filter's properties to match the new one. If the object has two different filters of this same type applied, it will tween the first one.

Some values are not tweenable and will be rewritten on the new filter if they differ from the original value.

Sample

```
// Shows how many tweenings a MovieClip object has
trace ("The MovieClip has " + <MovieClip>.getTweens() + " tweenings at this moment.");
```

See Also

isTweening()

Auxiliary Functions

.isTweening()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|Sound|TextField>.isTweening()
```

Parameters

None.

Returns

A boolean value (either true or false) informing if the object has a tweening going on or not. Future tweenings (created with delays) are also taken into consideration.

Description

Auxiliary function; it's used to find out if an object has a tweening being performed on it or not.

Sample

```
// Shows whether if a MovieClip has a tweening going on
if (<MovieClip>.isTweening()) {
  trace ("The MovieClip has a tweening going on at this moment.");
} else {
  trace ("The MovieClip doesn't have a tweening going on at this moment.");
}
```

See Also

isTweening()

Auxiliary Methods

.lockTween()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.lockTween();
```

Parameters

None.

Returns

Nothing.

Description

Auxiliary method; locks the object for tweening. After locking, no new tweenings may be added for that object. Current existing tweenings are still executed.

Sample

```
// Locks a MovieClip so it won't tween anymore
<MovieClip>.lockTween();
```

See Also

unlockTween()

Auxiliary Methods

.unlockTween()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|TextField>.unlockTween();
```

Parameters

None.

Returns

Nothing.

Description

Auxiliary method; unlocks the object for tweening. After unlocking, new tweenings can be added for that particular object again.

Sample

```
// Unlocks a MovieClip so tweens can be added again
<MovieClip>.unlockTween();
```

See Also

lockTween()

Auxiliary Methods

.pauseTween()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|Sound|TextField>.pauseTween([property(ies), property, ...);
```

Parameters

property(ies) : Property or properties to be paused. This can be a single string containing one property name (as in "_x"), an array containing a list of property names (as in ["_x", "_y"]), or a list of properties as parameters (as in "_x", "_y"). It can be omitted, too; in this case, all tweenings for that particular object will be paused.

Returns

Nothing.

Description

Method; pauses tweenings on a particular object, so they can be resumed later.

Sample

```
// Pauses an _alpha tweening that's being executed on a MovieClip
<MovieClip>.pauseTween("__alpha");

// Pauses all tweenings being done on a Sound object
<Sound>.pauseTween();

// Pauses an _x and and _y tweening on a MovieClip
<MovieClip>.pauseTween(["_x", "_y"]);

// Also pauses an _x and and _y tweening on a MovieClip
<MovieClip>.pauseTween("_x", "_y");
```

See Also

resumeTween(), stopTween()

Auxiliary Methods

.resumeTween()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|Sound|TextField>.resumeTween([property(ies), property, ...);
```

Parameters

property(ies) : Property or properties to be resumed. This can be a single string containing one property name (as in "_x"), an array containing a list of property names (as in ["_x", "_y"]), or a list of properties as parameters (as in "_x", "_y"). It can be omitted, too; in this case, all paused tweenings for that particular object will be resumed.

Returns

Nothing.

Description

Method; resumes tweenings that have been previously paused on a particular object. Non-paused tweenings are not affected.

Sample

```
// Resumes an _alpha tweening that has been paused on a MovieClip
<MovieClip>.resumeTween("__alpha");

// Resumes all paused tweenings on a Sound object
<Sound>.pauseTween();

// Resumes an _x and and _y tweening on a MovieClip
<MovieClip>.pauseTween(["_x", "_y"]);

// Also resumes an _x and and _y tweening on a MovieClip
<MovieClip>.pauseTween("_x", "_y");
```

See Also

pauseTween(), stopTween()

Additional Events

.onTweenComplete()

Applies To

MovieClip, Sound, TextField

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|Sound|TextField>.onTweenComplete = <function>;
```

Parameters

function : The name of a function, in reference (object) form, or an anonymous function.

Returns

N/A

Description

Event handler; invoked immediately after a tweening has ended. This function is called once for every tweened property, and receives the property name as a single string parameter. This function is defined once per object, not per tweening, so do not mistake this by tween()'s parameter argument.

This event handler works similarly to the MovieClip's built-in event handlers, like .onEnterFrame, .onRollOver, etc.

Sample

```
// Warns every time a tweening ends for a MovieClip object, using a named function
warnOnEnd = function(propName) {
  trace ("Tweening "+propName+" has ended.");
};
<MovieClip>.onTweenComplete = warnOnEnd;

// Warns every time a tweening ends for a MovieClip object, using an anonymous function
<MovieClip>.onTweenComplete = function(propName) {
  trace ("Tweening "+propName+" has ended.");
};

// Do something after a TextField _x tween has ended, using an anonymous function
<TextField>.onTweenComplete = function(propName) {
  if (propName == "_x") {
    trace ("The _x tweening has ended!");
    // do something here
  }
};
```

See Also

onTweenUpdate()

Additional Events

.onTweenUpdate()

Applies To

`MovieClip, Sound, TextField`

Availability

Flash 6 and above, using AS1 or AS2.

Usage

```
<MovieClip|Sound|TextField>.onTweenUpdate = <function>;
```

Parameters

function : The name of a function, in reference (object) form, or an anonymous function.

Returns

N/A

Description

Event handler; invoked when a tweening has update its assigned property. This function is called once for every tweened property, and receives the property name as a single string parameter. This function is defined once per object, not per tweening. This is useful when tweenings on a property must be replicated to other objects or properties.

This event handler works similarly to the `MovieClip's` built-in event handlers, like `.onEnterFrame`, `.onRollOver`, etc.

Sample

```
// Continually traces the value of an _x property of a MovieClip while tweening it
<MovieClip>.onTweenUpdate = function(propName) {
  if (propName == "_x") trace ("_x is now "+this._x);
};
<MovieClip>.xSlideTo(100, 1);

// Warns every time a property has been updated by a tweening on a TextField
<TextField>.onTweenUpdate = function(propName) {
  trace (propName + " has been updated!");
};
```

See Also

onTweenComplete()