++中文手册++++++++++++++++++++++++++

为什么要用 TWEENLITE 而不用 ADOBE 官方的 TWEEN?

1. 效率(这也是作者所标榜的 TWEENLITE2 大优点之一，呵呵，“标榜”这个词用得可能有点过了，不过人家确实有那个实力)这里有 2 者运行效率对比的例子：tweening-speed-test
2. onComplete, onStart, onUpdate 等回调方法是 TWEEN 所没有的(TWEENLITE 还可以往这些方法中传递任意个参数)
3. 智能的 alpha(当 alpha 达到 0 时，TWEENLITE 会自动将对象的 visible 设为 false)
4. 在一次方法调用中就可以缓动多个属性
5. 可以设置每一次缓动的延时(对有先后顺序的缓动很有效)
6. 实现任何 MovieClip/Sprite 的变色效果非常简单
7. 可以缓动 MovieClip 的声音
8. 唯一的 from()方法使你可以使用当前的属性值作为缓动的目标值

9. 使用相关联的值
10. 在一次方法调用中就可以缓动多个数组的值
11. TWEENLITE 默认会自动地覆盖同一个对象的缓动以免出现冲突(当然这个特性也是可以关闭的)
12. 强大的 delayedCall()方法使你可以随意设置延时和延时过后所调用的变量，甚至传递任意个数的参数
13. TWEENLITE 有一个更加 powerful 的大哥”TweenFilterLite”，而 TWEEN 只在孤军奋战…当 TWEENLITE 有什么搞不定的时候，直接去找他大哥…

用法：

TweenLite.to(target:Object, duration:Number, variables:Object);

Description：将 target 对象的属性从调用方法时的值缓动到 variables 中所指定的值

Parameters：

- target:要缓动的对象(这里注意类型是 Object, 并不仅仅是 MovieClip 和 Sprite)

- duration:持续的时间(单位是秒)
- variables:一个 Object,包含你想要缓动的所有属性(在 TweenLite.from()方法中，这些变量表示的则是开始缓动时的值),如果你给某个属性加引号，它将关联到当前的值。例如 y:”xxx”,无论你引号里指定多少，它只会缓动到当前的值或者从当前的值开始缓动(在 TweenLite.from()方法中)
- TweenLite.to(mc, 5, {x:"10", y:"20", ease:Elastic.easeOut});
  TweenLite.to(mc, 5, {x:"20", y:"30", ease:Elastic.easeOut}); *//2 者效果是一样的(通常情况下是不需要加引号的)*

  特殊属性：

  - delay:Number-延时几秒后开始缓动,这在有先后顺序的缓动效果中很有用
  - ease:Function-应用在 variables 上的缓动函数，比如 gs.easing.Elastic.easeOut。默认值是 Regular.easeOut.当然你也可以应用 CustomEase 来定义自己的缓动函数(CustomEase 属于收费内容)

- easeParam:Array-给缓动函数提供额外参数的数组。这在使用 Elastic 缓动函数时控制其他参数比图振幅和周期会非常有用(ADOBE 官方文档中，Elastic.easeXX()方法是可以有 6 个参数的，但是在 TweenLite 中只提供 4 个参数，那么另外 2 个参数：周期和振幅就可以在这个数组中给定)。大部分的缓动函数是只有 4 个参数的，所以通常情况下是不需要传入 easeParams 参数的。
- autoAlpha:Number-效果和改变”alpha”值一样，但是多了一个特性：如果 alpha 最终变为 0，则自动将 visible 设为 false。同样如果 autoAlpha 的值大于 0，则在开始缓动前会自动将 visible 设为 true
- visible:Boolean-缓动效果结束时 DisplayObject 对象的 visible 属性
- volume:Number-缓动对象的 SoundTransform 属性(例如 MovieClip/SoundChannel/NetStream 等)
- tint:uint-改变 DisplayObject 对象的色调，设置一个要缓动到的 16 进制的颜色值(在 from()函数中表示起始值)

- removeTint:Boolean-表示是否要取消一个应用在 DisplayObject 对象的 tint 属性
- frame:int-缓动 MovieClip 到指定的帧(在 from()函数中表示起始帧)
- onStart:Function-在缓动开始时触发此方法
- onStartParams:Array-数组，装有传递给 onStart 方法的参数
- onUpdate:Function-当属性值发生改变时(缓动进行中的每一帧，每一秒)触发此方法
- onUpdateParams:Array-数组，装有传递给 OnUpdate 方法的参数
- onComplete:Function-在缓动效果结束时触发此方法
- onCompleteParams:Array-数组,装有传递给 onComplete 方法的参数
- persist:Boolean-如果设为 true,在缓动效果结束时，TweenLite 实例将不会被垃圾回收器自动回收。当然，persist 为 true 的时候，这个缓动效果仍然可以被其他的缓动覆盖。默认值是 false

- renderOnStart:Boolean-在调用 TweenLite.from()函数并且还有一个延时的时候，如果想要让指定的起始属性值在延时结束之后才展现出来的话，将 renderOnStart 设为 true,相反，如果设为 false,在延时开始之时就会立即展现指定的起始属性值。默认值是 false
- TweenLite.from(mc, 5, {y:20, ease:Elastic.easeOut, delay:3, renderOnStart:**false**}); *//mc 会在 y=20 处停留 3 秒，然后在 5 秒内缓动到当前位置*

  TweenLite.from(mc, 5, {y:20, ease:Elastic.easeOut, delay:3, renderOnStart:**true**}); *//mc 会在当前位置置停留 3 秒,然后跳到 y=20 的位置开始缓动*

- overwrite:Boolean-如果不想让这个缓动效果被应用在同一个对象上的其他缓动效果自动覆盖的话，请将这个值设为 false

TweenLite.from(target:Object, duration:Number, variables:Object);

基本上同 TweenLite.to()方法一样，唯一不同的是所指定的缓动的属性是起始值

TweenLite.delayedCall(delay:Number, onComplete:Function, onCompleteParams:Array);
提供一个简单的方法来实现在指定的秒数之后调用指定的方法(其作用相当于 setTimeout()方法，如果单为了实现 setTimeout 的功能而给你的代码增加 3K 的重量，实在不值，但是如果项目中已经引入了 TweenLite,使用这个方法是个不错的主意),可以传递任意个数的参数
Parameters:略…

TweenLite.killTweensOf(target:Object, complete:Boolean);
提供一个简单的方法来移除应用在对象上的所有缓动效果，可以指定是否立即强制结束
Parameters:
target:目标对象
complete:如果设为 true,目标对象的缓动效果将会立即结束(缓动的属性直接变为缓动效果结束时的值，同时调用 onComplete 方法如果定义了的话)，若设为

false,同样会立即结束缓动效果 ,但是属性值将停留在当前缓动到的位置，onComplete 函数也不会执行

TweenLite.killDelayedCallsTo(function:Function);

提供一个简单的方法来移除 TweenLite.delayedCall 方法中设置了的方法(其作用相当于 clearTimeout)

Parameters:略..

TweenLite.removeTween(tween:TweenLite):void

回收指定的 TweenLite 的实例

Parameters:略.

示例：

```
1 import gs.TweenLite;import gs.easing.Back;
2
3 TweenLite.to(clip_mc, 5, {alpha:0.5, x:120,
4 ease:Back.easeOut, delay:2,
5 onComplete:onFinishTween,
6 onCompleteParams:[5, clip_mc]});
7
8 function onFinishTween(parameter1_num:Number,
9 parameter2_mc:MovieClip):void {      //注意这
```

```
    里的参数，分别对应前面 onCompleteParams 数组中的类型，而不是 Array 型

        trace("The tween has finished! parameters: " + parameter1_num + ", and " + parameter2_mc);

    }
```

源码包中也有 2 个很不错的例子~包含了大部分的用法在里面

About TweenLiteVars

Jack Doyle 在 7 月 15 日的更新中加入了类 TweenLiteVars，目的是为了让人们可以在 FB，FD,FDT 等编辑器中可以使用代码提示而不用去记众多的属性，并且能提供更加严谨的数据类型控制。

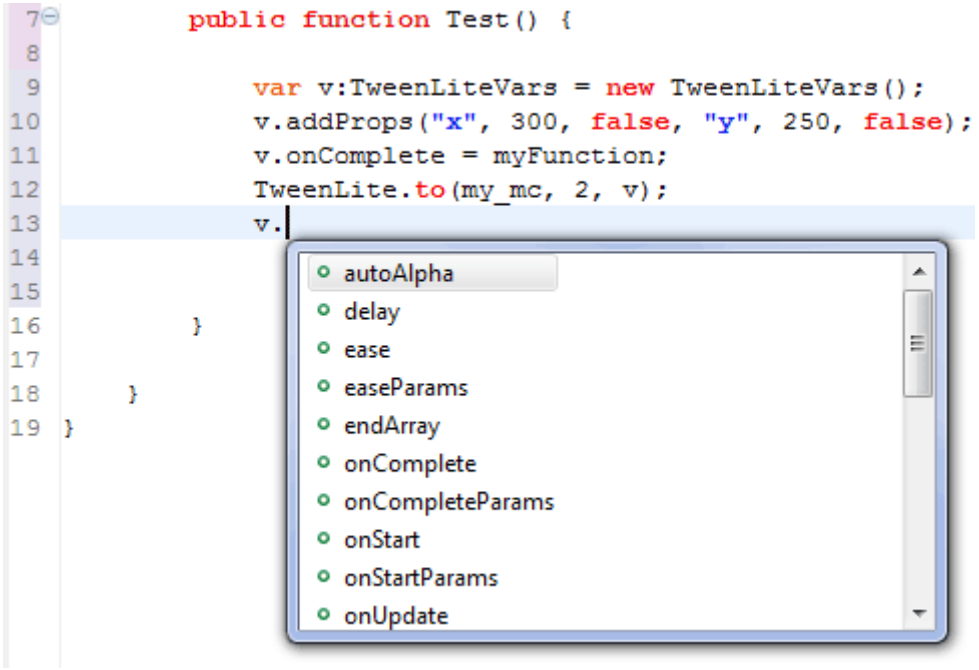
TweenLiteVars 类可以在 TweenMax 源码包中找到。

TweenLiteVars 的用法如下：

```
1 import gs.TweenLite;
2 import gs.utils.tween.TweenLiteVars;
3 import gs.easing.*;
4
```

```
5  var v:TweenLiteVars = new TweenLiteVars();
6  v.addProps("x", 300, false, "y", 100, true);
7  //with addProps(), you can add up to 15 dynamic
8  properties at a time. addProp() adds one at a
9  time.
10 v.ease = Elastic.easeOut;
   v.onComplete = myFunction;

   TweenLite.to(my_mc, 2, v);
```



++

+++++参数说明++

更新 及 更多文档请访问: http://www.TweenLite.com (这里的链

接指向 AS3 版)

描述:

缓动。 我们都在做。我们很多人都知道除了 Adobe's Tween 类之外，还有很多更好的动画引擎,（比如 Tweener）. 每种引擎都有它们各自的优缺点。

最近几年，为了得到一个更紧凑的，跑得更快，效率更高的引擎，我创建了 TweenLite （我无法接受其它的一些引擎带来的文件尺寸上的负担）.它很快就融入到我的所有工作中。我告诉其它人，让大家能够从中获益，最终，我将它发布了出来。在过去的几年中，TweenLite 越来越受欢迎，超乎了我的想像.

基于此，我又添加了一些新的功能，并且尽量保持这个文件的尺寸,让它小于 3K。TweenFilterLite 扩充了 TweenLite 并且加入了滤镜缓动，包含了 ColorMatrixFilter 的一些效果，比如饱和、对比、增亮、色调，甚至是着色，但文件的尺寸始终没有超过 3K。与 TweenLite 的做法相似，提供有 AS2 版和 AS3 版的类包下载。

TweenMax 比 TweenFilterLite 增加了更多的特性，包含 bezier 缓动，暂停/恢复，顺序执行等等。(见 www.TweenMax.com)

我猜你会想“如果这个是‘轻量级的’，那么它一定会丢掉很多特性，让我用的时候会有点担心”。这种想法是对的，在这里缺少一些其它缓动引擎所包含的特效，但是我可以肯定的

说，在过去几年我的工程（很多获奖的 flash 程序以及 500 强企业的项目中）中，我几乎一直都在用它，而它从没有让我失望过。

我还真没发现过我还需要其它的功能。你可以对任何的属性（包括 DisplayObject 对象的音量和颜色）使用缓动函数，内置的延迟时间，回调函数，以及传递参数给这些回调函数，甚至根据数组进行缓动，统统只在一行代码中完成。如果你需要更多的特效，你可以装上 TweenFilterLite 或者 TweenMax 来用用。

我也从来没有发现比这个更快的引擎。不同引擎执行效率的比较，请访问 http://blog.greensock.com/tweeing-speed-test/.

TweenLite 参数说明:

1) $ target : Object - 作为目标的对象， MovieClip 或者其它对象

2) $ duration : Number - 动画的时间长度（单位：秒）

3) $ vars : Object – 对象，通过属性值，来存贮各种属性参数用于缓动。（如果你使用 TweenLite.from() 方法，这里的参数表示缓动的初始值）

该对象所具有的属性：

alpha: alpha 目标对象应该完成 (或开始，当使用 TweenLite.from()时)的透明度级别.如果 target.alpha 是 1，当缓动被执行的时候，你指定参数为 0.5，它将把透明度从 1

缓动到 0.5.

x: 改变 MovieClip 的 x 位置,把这个值设置成你希望的 MovieClip 的结束位置(如果你使用的是TweenLite.from()这个值表示开始位置).

(y scaleX scaleY rotation 等属性不再重复说明）

特别的属性 (**可选的**):

delay : Number - 延迟缓动 (以秒为单位).

ease : Function - 缓动函数. 例如，fl.motion.easing.Elastic.easeOut 函数。默认的是 Regular.easeOut 函数。

easeParams : Array - 用来存贮缓动公式所需要的额外数据. 当使用 Elastic 公式并且希望控制一些额外的参数，比如放大系数和缓动时间。大多数的缓动公式是不需要参数的，因此，你不需要给其它的缓动公式传递参数。

autoAlpha : Number - 用它来代替 alpha 属性，可以获得一些副加的效果，比如当 alpha 值缓动到 0 时，自动将 visible 属性改为 false。当缓动开始前，autoAlpha 大于 0 时，它将会把 visible 属性变成 true 。

visible : Boolean - 在缓动结束时，想要指定 DisplayObject 的 visible 属性，请使用这个参数。

volume : Number - 对 soundTransform

（MovieClip/SoundChannel/NetStream 等）对象中的 volume 属性（音量大小）进行缓动

tint : Number - 改变 DisplayObject 的颜色，设置一个 16 进制颜色值之后，当缓动结束时，目标对象将被变成这个颜色，（如果使用的是 TweenLite.from()，这个值将表示目标对象开始缓动时的颜色)。举个例子，颜色值可以设定为：0xFF0000。

removeTint : Boolean - 要移除 DisplayObject 颜色，将这个参数设成 true 。

frame : Number - 将 MovieClip 缓动到指帧频。

onStart : Function - 在缓动开始时想要执行某个函数，就将函数的引用（通常是函数名）放到这里。如果缓动是带延迟的，那么在缓动开始前该函数不会被执行。

onStartParams : Array - 为缓动开始时要执行的函数传递参数。(可选的)

onUpdate : Function - 缓动过程中，每次更新时调用这里指定的函数(缓动开始后，每一帧被触发一次),

onUpdateParams : Array - 给 onUpdate 参数指定的函数传递参数 (可选的)

onComplete : Function - 缓动结束时执行的函数。

onCompleteParams : Array - 给 onComplete 参数指定的函数传递参数 (可选的)

persist : Boolean - 值为 true 时，TweenLite 实例将不会自动被系统的垃圾收集器给收走。但是当新的缓动出现时，它还是会被重写（overwritten）默认值为 false.

renderOnStart : Boolean - 如果你使用带有延迟缓动的 TweenFilterLite.from() ，并且阻止缓动的渲染（rendering ）效果，直到缓动真正开始，将这个值设为 true. 默认情况下该值为 false ，这会让渲染效果立即被执行，甚至是在延迟的时间还没到之前。

overwrite : int - 当前的缓动被创建以后，通过这个参数可以限制作用于同一个对象的其它缓动，可选的参数值有：

- 0 (没有): 没有缓动被重写。这种模式下，运行速度是最快的，但是需要注意避免创建一些控制相同属性的缓动，否则这些缓动效果间将出现冲突。

- 1 (全部): (这是默认值，除非 OverwriteManager.init() 被调用过)对于同一对象的所有缓动在创建时将会被完全的覆盖掉。

```
TweenLite.to(mc, 1, {x:100, y:200});

TweenLite.to(mc, 1, {x:300,
```

delay:2}); //后创建的缓动将会覆盖掉先前创建的缓动，（可以起到这样的作用：缓动进行到一半时被中断，执行新的缓动 译者注）

- 2 (自动): (当 OverwriteManager.init() 被执行后,会根据具体的属性值进行选择)只覆盖对同一属性的缓动。

TweenLite.to(mc, 1, {x:100, y:200});

TweenLite.to(mc, 1, {x:300}); //only "x" 属性的缓动将被覆盖

- 3 (同时发生): 缓动开始时，覆盖全部的缓动。

TweenLite.to(mc, 1, {x:100, y:200});

TweenLite.to(mc, 1, {x:300, delay:2}); //不会覆盖先前的缓动，因为每二个缓动开始时，第一个缓动已经结束了。

举例:

将实例名为 "clip_mc" 的 MovieClip 透明度降到 50% (0.5) ，并将它 x 轴位置移动到 120 ，将音量将到 0，缓动总共用时 1.5 秒，代码如下：

```
import gs.TweenLite;
TweenLite.to(clip_mc, 1.5, {alpha:0.5, x:120, volume:0});
```

如果希望使用更高级的缓动函数在 5 内，将 alpha 变到 0.5，将 x 移动 到 120 ，使用 "easeOutBack" 弹性函数，缓动整体延迟 2 秒发生，并且在缓动结束时，执行 "onFinishTween" 函数，并且为这个函数传递几个参数，(一个数值 5 以及对 clip_mc 的引用)，代码如下：

```
import gs.TweenLite;
import fl.motion.easing.Back;
TweenLite.to(clip_mc, 5, {alpha:0.5, x:120, ease:Back.easeOut, delay:2, onComplete: onFinishTween, onCompleteParams:[5, clip_mc]});
function onFinishTween(argument1:Number, argument2:MovieClip):void {
    trace("The tween has finished! argument1 = " + argument1 + ", and argument2 = " + argument2);
}
```

如果你的舞台上的 MovieClip 已经停在了它的结束位置，你只想让它花上 5 秒种回到这个位置，(只需要改变 y 属性，

比当前位置高 100 像素的位置，让它从那里下落), 代码如下（这次使用的是 TweenLite.from 译者注):

```
import gs.TweenLite;
import fl.motion.easing.Elastic;
TweenLite.from(clip_mc, 5, {y:"-100", ease:Elastic.easeOut});
```

说明:

- TweenLite 类会让你的 Flash 文件增加 3kb 大小。

- 给参数值加上引号，表示对指定的属性进行相应操作。比如，使用 TweenLite.to(mc, 2, {x:"-20"}); 它将 mc.x 向左移动 20 像素，与此相同效果的代码是：TweenLite.to(mc, 2, {x:mc.x - 20});

- 你可以用别的缓动函数替换 TweenLite 默认的缓动函数: Regular.easeOut.

- 必须使用 Flash Player 9 或之后版本的播放器 (ActionScript 3.0)

- 可以对任何 MovieClip 使用 "volume" 缓动,就比如：TweenLite.to(myClip_mc, 1.5, {volume:0});

- 可以将 MovieClip 设定成某种颜色，使用 "tint" 参数,比如: TweenLite.to(myClip_mc, 1.5, {tint:0xFF0000});

- 想要对数组内容进行缓动，将数值放到一个叫 endArray 的数组中即可，例如:

```
var myArray:Array = [1,2,3,4];
TweenLite.to(myArray, 1.5, {endArray:[10,20,30,40]});
```

- 可以在任何时候终止缓动，使用 TweenLite.killTweensOf(myClip_mc); 函数。如果想强制终止缓动，可以传递一个 true 做为第二个参数，比如 TweenLite.killTweensOf(myClip_mc, true);

- 取掉延迟回调函数，用 TweenLite.killDelayedCallsTo(myFunction_func);这项功能可以用来控制回调函数的优先级。

- 使用 TweenLite.from() 方法，可以使用对象从别的位置回到当前的位置。例如，你可以将对象在舞台上摆放整齐（缓动结束时的位置），然后利用
缓动，让它们跑到那个位置上去，你可以将缓动的初始位置值 x 或 y 或 alpha (或者其它你需要的属性)当做参数传递给这个方法函数。

TweenLite 下载链接：

http://www.greensock.com/ActionS ... S3/TweenLiteAS3.zip

下载下来的类包中，有一个 TweenLiteAS3_Sample_1.swf ，初学者可以用它来观察各种缓动的效果，并且直接得到相关的执行代码。算是一个可视化设计的工具，不要错过。

下载到类包以后，解压缩到一个目录比如：d:\AS3Class ，在 flash9 的首选参数－>ActionScript->ActionScript3.0 设置中添加类目录，d:\AS3Class\TweenLiteAS3 即可正确引用到相关的类。

应用举例：

```
import gs.TweenLite;
import gs.easing.*;

stage.addEventListener(MouseEvent.CLICK, onCK);
function onCK(evt) {
        TweenLite.to(mc, 0.5, {x:mouseX, y:mouseY, rotation:360});
}
```

在舞台上点击，会让 mc 元件旋转并跑动到鼠标位置。