TweenLite.to(mc, 1.5, {x:100}); 里面的 mc 指所作用的对象,1.5 指运动的时间，{x:100} 表示 mc 的 x 属性变化，最终停下来时 x 的值为 100. (即 mc 从当前位置,经过 1.5 秒，匀速移动到 x=100 的位置)

TweenLite.from(mc, 1.5, {x:100}) 里面的 mc 指所作用的对象,1.5 指运动的时间, 这里是指，mc 从当前位置,经过 1.5 秒， 从 x=100 的位置移动到当前 mc 所在的位置。“注意与上面的区别，他们恰好相反”)

下面我们以上面的 **TweenLite**.to() 来讲解相关属性：

TweenLite.to() **TweenLite**.from() 返回的都是 **TweenLite** 类型

TweenLite.to(mc, 1.5, {x:100});我们还可以在{}里加些其他相关的属性

下面假如 delay 属性 其表示 延迟 delay 时间后才发生 Tween :

TweenLite.to(mc, 1, {x:100, delay:2}); 表示执行 **TweenLite**.to(mc, 1, {x:100, delay:2})后，需要经过 2 秒，mc 才发生移动

我们还可以在{}加入 alpa ease(缓动) onComplete(所调用的方法名)等属性：

TweenLite.to(mc, 1.5, {x:100, ease:Elastic.easeOut, delay:0.5, onComplete:myFunction});

function myFunction():void {

trace("tween finished");}

上面的 ease:Elastic.easeOut 是缓动类的一种类型 onComplete:myFunction 表示 tween 执行完后就会调用 myFunction 方法(即执行完 Tween 后，紧接着就调用 myFunction 方法，)。

TweenLite 还有一方法，如：pause(), resume(), reverse(), restart())

pause() 表示暂停 。resume() 表示继续播放缓动。restart()表示重头开始播放缓动。reverse()表示按与原方向相反的方向缓动(例如：缓动了 2 秒后，调用该方法，就会经过相同的时间(2 秒)按原路返回)。

下面是一个对各个属性进行验证的小例子：

start_btn 是开始按钮，即按下它才会执行缓动

btn 按钮是对一些方法的验证 ，你可以改变里面的相关方法进行验证 如将 tween.reverse();改成 tween.resume();

```
import com.greensock.*;

import com.greensock.easing.*;

var isPause:Boolean;

var tween:TweenLite;

btn.enabled=false;

start_btn.addEventListener(MouseEvent.CLICK,begin);

btn.addEventListener(MouseEvent.MOUSE_DOWN,onDown);

function begin(e:MouseEvent) {

btn.enabled=true;

tween=TweenLite.from(mc,10,{x:300,y:300,alpha:0.5,delay:2,onComplete:completeIt});

}

function completeIt() {

trace("缓动执行完毕，开始调用此方法");

trace(tween);//TweenLite 类型

}

function onDown(e:Event):void {

if (isPause==false) {

tween.pause();

isPause=true;

} else {

tween.reverse();

isPause=false;
```

}

}

本 文 来 自 CSDN 博 客 ， 转 载 请 标 明 出 处 ：http://blog.csdn.net/hu36978/archive/2009/11/02/4755037.aspx

overwrite 属性 ： 默认值是 2 即 auto

一般用法：

TweenLite.to(mc, 1, {x:100, overwrite:2}); //推荐 使用这种， 2 代表的是模式 2 即 AUTO (2) 模式 其也是默认值

// 或者

TweenLite.to(mc, 1, {x:100, overwrite:true});

overwrite 属性来自 OverwriteManager 类

其有五种模式

0：速度最快的模式

1： 适合按钮使用的模式 ,按钮发生 roll_over/roll_out 事件

2 ：默认模式 。除了无速度(选择 0)要求，和作用对象为按钮(选择 2)外，一般用默认模式。

还有三种模式 3，4，5 详细介绍: http://blog.greensock.com/overwritemanager/

在使用模式时 ，需要初始化 即：

OverwriteManager.init(2) // 里面的模式 2 只是起个初始化作用， 你可以将其修改为其他的模式，但是上面的初始化必不可少，否则无效(无效则用默认模式 2) 。

模式修改是在 overwrite 里进行的。

例如：

OverwriteManager.init(2) //初始化

//修改

TweenLite.to(btn, 1, {x:100, overwrite:1});//初始化模式为 2，现在修改为 1

TweenLite 和 TweenMax 的比较

使用 TweenLite，编译后文件较小，假如对文件大小有要求的话，推荐使用这种。

使用 TweenMax，编译后文件较大 ，该类，功能很多，编译后文件较大。

他们的用法相似：

TweenLite.to(mc, 1.5, {x:100, y:200, onComplete:myFunction, ease:Strong.easeOut});

TweenMax.to(mc, 1.5, {x:100, y:200, onComplete:myFunction, ease:Strong.easeOut});

使用 TimelineLite

可以将其 TimelineLite 看成 MovieClip

它的作用是控制 tween 的运动，其当我们需要 mc 的运动一个接着一个

例子:

var myTimeline:TimelineLite = new TimelineLite();

myTimeline.append(new TweenLite(mc, 1, {x:100}));//添加进 myTimeline

myTimeline.append(new TweenLite(mc, 1, {y:200}));

myTimeline.append(new TweenMax(mc, 1, {tint:0xFF0000}));

上面的代码的结果是： mc 首先移到 x=100 处，然后移动到 y=200 处，最后颜色发生渐变，渐变到 0xFF0000

它们的运动是一个接着一个进行。

下面是一个下例子 ， 测试，例子，并没有发生 tween 运动，当鼠标滑上和滑下按钮 menu 时执行 tween 运动

故我们控制刚测试时不运动 即： var myTimeline:TimelineLite = new TimelineLite({paused:true});

这样开始时就不发生 tween 运动了。TimelineLite 的属性方法和 TweenLite 很多一样。

var tween = new TweenLite({paused:true});//这样开始时也是不发生 tween 运动

下面是具体的代码：

```
var myTimeline:TimelineLite = new TimelineLite({paused:true});

myTimeline.append( new TweenLite(mc, 1, {x:100}) );

myTimeline.append( new TweenLite(mc, 1, {y:200}) );

myTimeline.append( new TweenMax(mc, 1, {tint:0xFF0000}) );

menu.addEventListener(MouseEvent.ROLL_OVER, overHandler);

menu.addEventListener(MouseEvent.ROLL_OUT, outHandler);

function overHandler(event:MouseEvent):void {

myTimeline.play();

}

function outHandler(event:MouseEvent):void {

myTimeline.reverse();

}
```

测试显示，只有滑上和滑出 menu 按钮发生运动

除此之外，我们还可以在某个时刻或者某个标签插入 tween 运动，用到的方法是 TimelineLite 的 insert()方法。

我们还可用 TimelineLite 的 addlabel() 给某个时刻插入标签。

append() 的第二个参数 offset 表示上一个 tween 过后，在过 offset 时间执行本 tween 运动

例子：

```
var myTimeline:TimelineLite = new TimelineLite();

//在时刻为 1 秒的地方插入 tween 运动，即第一秒结束后才运动
```

myTimeline.insert(new TweenLite(mc, 2, {x:100}), 1);

//提前 1.5 秒发生 tween 运动

myTimeline.append(new TweenLite(mc, 1, {y:200}), -1.5);

// 为 4 秒时刻增加标签 spin

myTimeline.addLabel("spin", 4);

//在 spin 标签处添加 tween 动画

myTimeline.insert(new TweenLite(mc, 1, {rotation:"360"}), "spin");

我们也可对多个运动同时进行运动

myArrayOfSprites 是运动对象 mc 的数组

myTimeline.insertMultiple(TweenMax.allFrom(myArrayOfSprites, 1, {y:"-100", autoAlpha:0}));

TweenLite.delayedCall(2, myFunction, [myParam1, myParam2]); //表示经过 2 秒后执行 myFunction()方法，[myParam1, myParam2 是该方法的参数

TweenLite.to(mc, 1, {x:"100"});//在原坐标的基础上增加 100 像素 注意与 x:100 的区别

//or if the value is in a variable, cast it as a String like this:

TweenLite.to(mc, 1, {x:String(myVariable)});