FLEX - EFFECT

Introduction

The Effect class is an abstract base class that defines the basic functionality of all Flex effects. The Effect class defines the base factory class for all effects. The EffectInstance class defines the base class for all effect instance subclasses.

Class declaration

Following is the declaration for **mx.effects.Effect** class:

```
public class Effect
   extends EventDispatcher
      implements IEffect
```

Public properties

S.N.	Property & Description
1	**className : String** [read-only] The name of the effect class, such as "Fade".
2	**customFilter : mx.effects:EffectTargetFilter** Specifies a custom filter object, of type EffectTargetFilter, used by the effect to determine the targets on which to play the effect.
3	**duration : Number** Duration of the effect in milliseconds.
4	**effectTargetHost : mx.effects:IEffectTargetHost** A property that lets you access the target list-based control of a data effect.
5	**filter : String** Specifies an algorithm for filtering targets for an effect.
6	**hideFocusRing : Boolean** Determines whether the effect should hide the focus ring when starting the effect.
7	**instanceClass : Class** An object of type Class that specifies the effect instance class class for this effect class.

8

isPlaying : Boolean

[read-only] A read-only flag which is true if any instances of the effect are currently playing, and false if none are.

9

perElementOffset : Number

Additional delay, in milliseconds, for effect targets after the first target of the effect.

10

playheadTime : Number

Current time position of the effect.

11

relevantProperties : Array

An Array of property names to use when performing filtering.

12

relevantStyles : Array

An Array of style names to use when performing filtering.

13

repeatCount : int = 1

Number of times to repeat the effect.

14

repeatDelay : int = 0

Amount of time, in milliseconds, to wait before repeating the effect.

15

startDelay : int = 0

Amount of time, in milliseconds, to wait before starting the effect.

16

suspendBackgroundProcessing : Boolean = false

If true, blocks all background processing while the effect is playing.

17

target : Object

The object to which this effect is applied.

18

targets : Array

An Array of objects that are targets for the effect.

19

triggerEvent : Event

The Event object passed to this Effect by the EffectManager when an effect is triggered,

or null if the effect is not being played by the EffectManager.

Protected properties

S.N.	Property & Description
1	**applyTransitionEndProperties : Boolean** This flag controls whether the effect, when run in a transition, automatically applies the property values according to the end state, as opposed to leaving values as set by the effect itself.
2	**endValuesCaptured : Boolean = false** A flag containing true if the end values of an effect have already been determined, or false if they should be acquired from the current properties of the effect targets when the effect runs.

Public methods

S.N.	Method & Description
1	**Effect***target*: *Object* = *null* Constructor.
2	**captureEndValues:void** Captures the current values of the relevant properties on the effect's targets and saves them as end values.
3	**captureMoreStartValues***targets*: *Array*:**void** Captures the current values of the relevant properties of an additional set of targets Flex uses this function when a data change effect is run.
4	**captureStartValues:void** Captures the current values of the relevant properties on the effect's targets.
5	**createInstance***target*: *Object* = *null*:**IEffectInstance** Creates a single effect instance and initializes it.
6	**createInstances***targets*: *Array* = *null*:**Array** Takes an Array of target objects and invokes the createInstance method on each target.

7

deleteInstance*instance*:*IEffectInstance*:**void**

Removes event listeners from an instance and removes it from the list of instances.

8

end*effectInstance*:*IEffectInstance = null*:**void**

Interrupts an effect that is currently playing, and jumps immediately to the end of the effect.

9

getAffectedProperties:Array

Returns an Array of Strings, where each String is the name of a property changed by this effect.

10

pause:void

Pauses the effect until you call the resume method.

11

play*targets*:*Array = null, playReversedFromEnd*:*Boolean = false*:**Array**

Begins playing the effect.

12

resume:void

Resumes the effect after it has been paused by a call to the pause method.

13

reverse:void

Plays the effect in reverse, if the effect is currently playing, starting from the current position of the effect.

14

stop:void

Stops the effect, leaving the effect targets in their current state.

Protected methods

S.N.	Method & Description

1

applyValueToTarget*target*:*Object, property*:*String, value*: * , *props*:*Object*:**void**

Used internally by the Effect infrastructure.

2

effectEndHandler*event*:*EffectEvent*:**void**

Called when an effect instance has finished playing.

3	**effectStartHandler***event*:*EffectEvent*:**void** This method is called when the effect instance starts playing.
4	**effectStopHandler***event*:*EffectEvent*:**void** Called when an effect instance has stopped by a call to the stop method.
5	**filterInstance***propChanges*:*Array*, *target*:*Object*:**Boolean** Determines the logic for filtering out an effect instance.
6	**getValueFromTarget***target*:*Object*, *property*:*String*:* Called by the captureStartValues method to get the value of a property from the target.
7	**initInstance***instance*:*IEffectInstance*:**void** Copies properties of the effect to the effect instance.

Events

S.N.	Event & Description
1	**effectEnd** Dispatched when one of the effect's instances finishes playing, either when the instance finishes playing or when the effect is interrupted by a call to the end method.
2	**effectStart** Dispatched when the effect starts playing.
3	**effectStop** Dispatched when the effect has been stopped, which only occurs when the effect is interrupted by a call to the stop method.

Methods inherited

This class inherits methods from the following classes:

- flash.events.EventDispatcher
- Object