

# Tween Engine Comparisons

## [KitchenSync](#) 1.1

### Syntax

```
package {
    import com.mimswright.easing.*;
    import com.mimswright.sync.*;

    import flash.display.*;

    public class KSTutorial extends Sprite {
        public function KSTutorial() {
            // set framerate to 30 frames per second
            stage.frameRate = 30;
            // start the Synchronizer. You'll only need to do this once
            Synchronizer.initialize(this);

            // display a message after 5 seconds.
            var message:SynchronizedTrace = new SynchronizedTrace(5, "Hello!");
            message.start();

            // Draw a rectangle to use for demonstrations.
            var sprite:Sprite = new Sprite();
            sprite.graphics.beginFill(0);
            sprite.graphics.drawRect(0,0,20,20);
            addChild(sprite);

            // move the rectangle from left to right.
            var tweenX:Tween = new Tween(sprite, "x", 300, 0, 1000);
            // apply the same animation to the y property.
            var tweenY:Tween = tweenX.cloneWithTarget(sprite, "y");

            // add the x and y tweens to a new sequence.
            var sequence:Sequence = new Sequence(
                tweenX,
                tweenY
            );
            // why stop there... let's make the complete trip back
            sequence.addAction(tweenX.cloneReversed());
            sequence.addAction(tweenY.cloneReversed());
        }
    }
}
```

```
        // begin the sequence.
        sequence.start();
    }
}
```

## Pros

- Sequences tweens, functions, groups, events, and more with one framework.
- Lots of documentation.
- Smart, flexible OOP design.
- Lots of features.
- New (non-penner) tweens.
- Apply tweens to any numeric property of any object

## Cons

- large. 20-50KB.
- Unoptimized choppy animation. Rated mid-range on Moses' TweenBencher
- May be intimidating to developers not as familiar with OOP.
- Lacks some features that developers have come to expect from a tween engine such as color and filter tweens.
- Tweens operate on a single property at a time making them inefficient.
- Small user base.
- Non-standard interface. Uses custom tween libraries

## Other Notable Features

- Less animation-centric than other classes. Has more built-in functionality for functions, events, and sequences.

[TweenLite](#) / [TweenFilterLite](#) / [TweenMax](#)

## Syntax

```
// Tween an mc to 46, 43 in 1 second
TweenLite.to(mc, 1, {x:46, y:43});
```

## Pros

- Very small filesize (3-8KB)
- Very fast (although, am i crazy or does the animation look a little jittery in firefox?)
- straightforward syntax. Can tween multiple properties on a single line.
- Strong community support ;)

## Cons

- No sequences except with max.uses delays instead.
- Uses only object syntax so no strong typing or compile time checking.
- Parsing text from objects means learning new string-based syntax. For TweenFilterLite doubly so.
- Confusing method names and a lack of great documentation
- Uses callbacks rather than events

## Notable Features

- Tweens are started as soon as they're created. The tweens aren't tracked individually
- Only one class.
- Can control volume of audio.
- Allows tweening between different frames in a movie clip

## Tweener

### Syntax

```
// Using delays to create animation sequences
Tweener.addTween(myMovieClip, {_x:20, time:0.5});
Tweener.addTween(myMovieClip, {_x:0, time:0.5, delay: 0.5});
```

### Pros

- Simple to use
- been around since AS1 so people are very familiar with the syntax
- Uses objects to define tween parameters
- Modifies any properties, not just MCs
- Tweener keeps track of other tweens happening on a property
- easeOutIn easingFunctions

- Small filesize ~10KB
- Fast and flexible
- Documentation walks users through the basics of classes etc. Lots of example snippets.

## Cons

- No sequences
- Uses only object syntax so no strong typing or compile time checking.
- Uses callbacks
- Simple syntax

## Notable Features

- Tweens are started as soon as they're created.
- Uses a base object to inherit properties from another object {base=template}
- Allows looping of functions at set periods
- Parameter objects can be stored and used as a template
- Allows Frames or time but uses a useFrames parameter to determine which

## [Boostworthy Animation System 2.1](#)

### Syntax

```
// Store a global reference to the stage.
// This is needed for objects to have reference to the enter
// frame event.
Global.stage = stage;

// Create a new timeline object.
objTimeline = new Timeline(RenderMethod.ENTER_FRAME);
// Create and add each tween to the timeline.
objTimeline.addTween(new Tween(m_spBox, "x",          500, 1,
15, Transitions.SINE_IN_AND_OUT));
objTimeline.addTween(new Tween(m_spBox, "y",          100, 7,
22, Transitions.SINE_IN_AND_OUT));
objTimeline.addTween(new Tween(m_spBox, "rotation", -180, 7,
22, Transitions.SINE_IN_AND_OUT));
objTimeline.addTween(new Tween(m_spBox, "x",          275, 16,
30, Transitions.SINE_IN_AND_OUT));
```

```
objTimeline.addTween(new Tween(m_spBox, "scaleX", 0.5, 16,
30, Transitions.SINE_IN_AND_OUT));
objTimeline.addTween(new Tween(m_spBox, "scaleY", 0.5, 16,
30, Transitions.SINE_IN_AND_OUT));
objTimeline.play();
```

## Pros

- Uses Flex metadata tags extensively
- Turbo is supposedly the best performing animation engine.
- Allows animation along paths
- Optimized by marking values as dirty and updating only once.

## Cons

- One enumeration for all easing functions
- Uses `Global.stage = stage;` to set up the stage

## Notable Features

- Ability to use Timer events or EnterFrame events to trigger animation
- Has an explicit Timeline class which allows you to add several things and play them back and forward keeping track of the time for all of them as though it was on a timeline.

## [FuseKit 2.1](#)

### Syntax

```
import com.mosesSupposes.fuse.*;
ZigoEngine.register(FuseItem);
var f:Fuse = new Fuse( { _x:'100', controlY:'-50' },
{ _y:'100', controlX:'50' },
{ _x:'-100', controlY:'50' },
{ _y:'-100', controlX:'-50' });
f.target = box_mc;
f.start();
```

## Pros

- Parses text from objects which makes it very flexible.
- Large user base. Been around a while.
- Small. Only 10KB
- Large set of features.
- Relative tweens handled easily, albeit inelegantly
- Allows animation along bezier curves

## Cons

- Uses multiple packages. Requires you to register every package you want to use.
- Fairly complicated interface
- Can parse several different words for some properties. e.g. duration/seconds/time.
- Parsing text from objects means learning new string-based syntax
- No AS3 version

## Notable Features

- Documentation addresses beginners and advanced coders separately and discusses introductory topics for beginners.
- Labels for every tween so they can be kept in memory by a static class and looked up by ID
- Allows tweens to fail or continue when an error occurs on a per-action basis

## [Go](#)

### Pros

- Well optimized.
- Designed by someone with lots of tween experience
- video tutorial
- instructs developers how tweens are built
- Seems to allow tweens to be controlled in very specific ways

### Cons

- Not an actual tweening engine but more of a meta-framework. Requires a lot of work on the developers end to get it going.
- seems like lots of people will be writing the same code
- Too advanced for most casual users.

- Confusing class names / architecture

## **Notable Features**

- Labels for every tween so they can be kept in memory by a static class and looked up by ID
- Automatically detects and prevents you from creating two tweens on the same target/property.
- Could become wrapped up with future iterations of flash.