The enchant.js Animation Engine in 5 Minutes

Introductions

Ryohei Fushimi (@sidestepism)
Ubiquitous Entertainment, Inc. (UEI)
Akihabara Research Center (ARC)
9leap Project Leader

English Translation:
Brandon McInnis (@BrandonUnfathom)
enchant.js Technical Evangelist

If we want to make the bear move...

```
bear.addEventListener(“enterframe”, function(){
    bear.x ++;
});

// moves the bear to the right by one pixel every frame
```

He moves!

But he doesn’t stop...

```
bear.addEventListener(“enterframe”, function(){
    if(bear.x < 100)bear.x ++;
});
```

// moves the bear to the right by one pixel every frame IF the bear’s position on the x-axis is less than 100

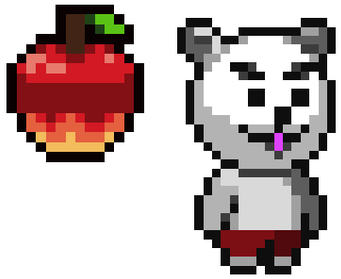
//(x,y coordinates always start at (0,0) in the upper left-hand corner)

This time...

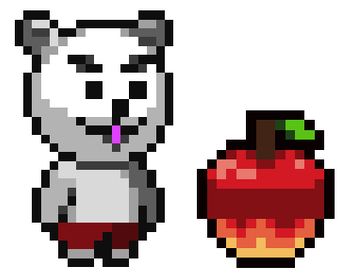
He stopped!

But...

But...

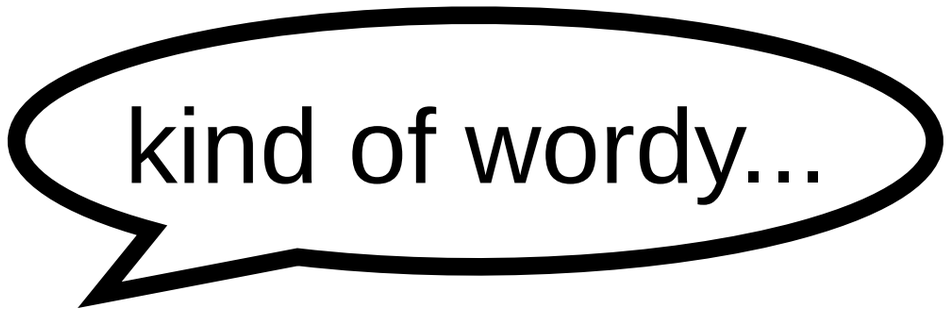


But...



What about movement like this?

We use
animation.enchant.js



We use

animation.enchant.js

We use

animation.enchant.js

tl.enchant.js

We use

Note: As of version v0.6, tl.enchant.js is now part of the enchant.js main library, and not a separate plugin. The following examples still work the same way, however.

tl : abbreviation for TimeLine

We’ll explain this later.

```
// To move the bear to (120,120) over 30 frames (1 second by the default 30 fps)

bear.tl.moveTo(120, 120, 30);
```

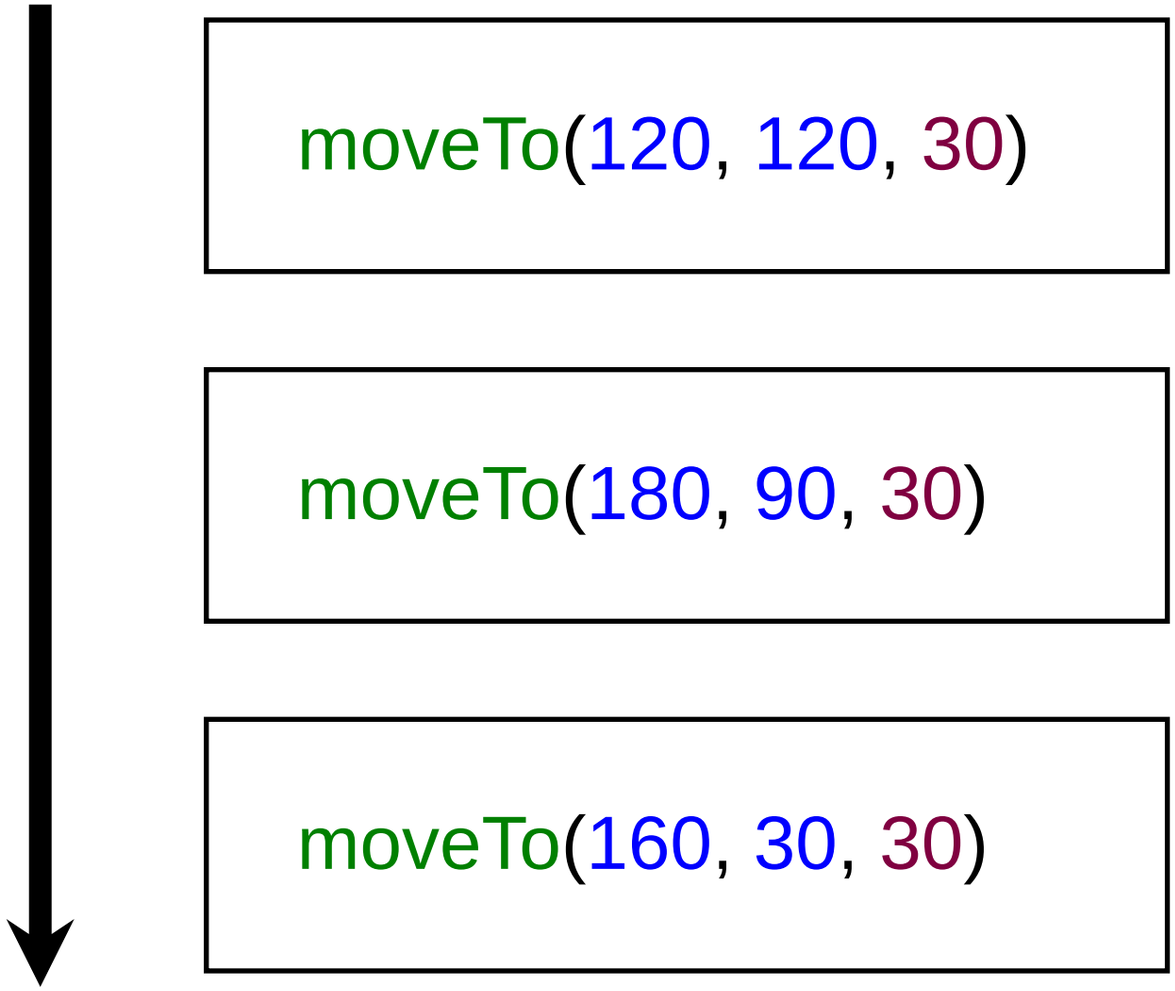
```
// First move the bear to (120, 120) over 30 frames,
// then move the bear to (60,180) over 30 frames.

bear.tl.moveTo(120, 120, 30).moveTo(60, 180, 30);
```

```
// Move the bear to (120, 120) over 30 frames,
// then move the bear to (60, 180) over 30 frames,
// then move the bear to (180, 90) over 30 frames.

bear.tl.moveTo(120, 120, 30)
        .moveTo(60, 180, 30)
        .moveTo(180, 90, 30);
```

Order of processing



Representation of how timeline commands are stacked in a queue.

Feature #0

Movement
moveTo/moveBy

```
bear.tl.moveTo(120, 120, 30);
// absolute movement to (120, 120)

bear.tl.moveBy(40, 40, 30);
// relative movement by (+40, +40)
```

Feature #1 Fade in
fadeIn

```
bear.tl.fadeIn(30);

// fade in over 30 frames
```

Feature #2

Changing size scaleTo/scaleBy

```
bear.tl.scaleTo(3, 30);

// scale the bear by a factor of 3 (3x normal size) over 30 frames
```

Feature #3 Specified time delay

delay

```
bear.tl.delay(30).fadeIn(30);

// wait 30 frames, then fade the bear in over 30 frames
```

Feature #4 Run a function then

```
bear.tl.delay(30).then(function(){
    scene.removeChild(bear);
});

// wait 30 frames, then remove the bear from the screen
```

Feature #5

Run multiple functions cue

```
bear.tl.cue({
    10: function(){ ... },
    20: function(){ ... },
    30: function(){ ... },
    50: function(){ ... }
});

// execute functions at specified frames (i.e. at 10 frames after execution, run first function; 20 frames after execution, run second function; etc.)
```

Feature #6

Tweening
tween

```
bear.tl.tween({
    x: 120,
    y: 120,
    scaleX: 3,
    scaleY: 3,
    time: 100
});

// perform multiple tl operations simultaneously over 100 frames

// Did you know? The word “tween” comes from the word “inbetweening,” meaning to create animation frames between two keyframes. http://en.wikipedia.org/wiki/Inbetweening
```

Feature #7 Parallel Execution and

```
bear.tl.fadeIn(30).moveTo(120, 120, 30)
// fade in over 30 frames, and then move to (120,120) over
30 frames

bear.tl.fadeIn(30).and().moveTo(120, 120, 30)
// fade in over 30 frames and move to (120,120) over 30
frames simultaneously!
```

Feature #8 loops
loop

```
bear.tl.fadeIn(30).fadeOut(30).loop();
// loop an animation of fading in over 30 frames, and then
fading out over 30 frames
```

Feature #9

Fast-forward skip

```
bear.tl.skip(100);

// fast-forward by 100 frames
```

Feature#10

Execute a function repeatedly

repeat

Feature#11

Pause/resume the queue

pause / resume

Feature#12

Erase the entire queue

clear

Feature#13

Execute a function, wait for it to complete, and then move on

waitUntil

Feature#14

Define an action

action

Feature#15

Erase from a scene

removeFromScene

Feature#16 Time-based animation
setTimeBased

```
bear.tl.setTimeBased();
bear.tl.fadeIn(3000)

// make all tl animation functions accept duration values
in milliseconds instead of frames, then fade in the bear
over 3000 ms (3 seconds)

// this can be reversed with setFrameBased();

// for more information on time-based animation with
enchant.js, see Kevin Kratzer's tutorial at
http://bit.ly/XcLHDC
```

You must remember

EASING

If the rate at which a sprite moves changes over time...

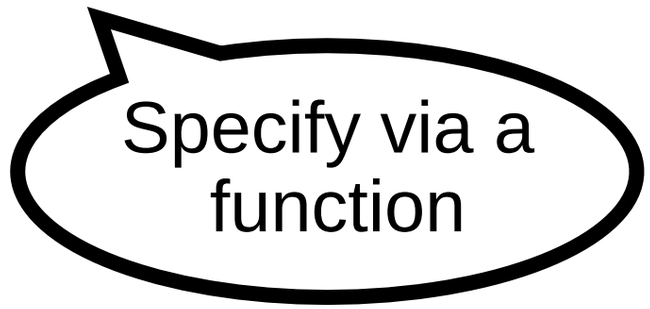
this is called EASING!

If this is confusing, please see the first part of http://bit.ly/109CYbP for an explanation of the concept of easing, using Lego animation.

```
// Easing

bear.tl.moveTo(120, 120, 30, enchant.Easing.QUAD_EASEINOUT);

bear.tl.tween({
    x: 120,
    y: 120,
    scaleX: 3,
    scaleY: 3,
    time: 100,
    easing: enchant.Easing.QUAD_EASEINOUT
}));
```



Several preset easing functions can be chosen from.

enchant.Easing.

QUAD_
QUBIC_
QUINT_
SIN_
BACK_
CIRC_
ELASTIC_
BOUNCE_
EXPO_

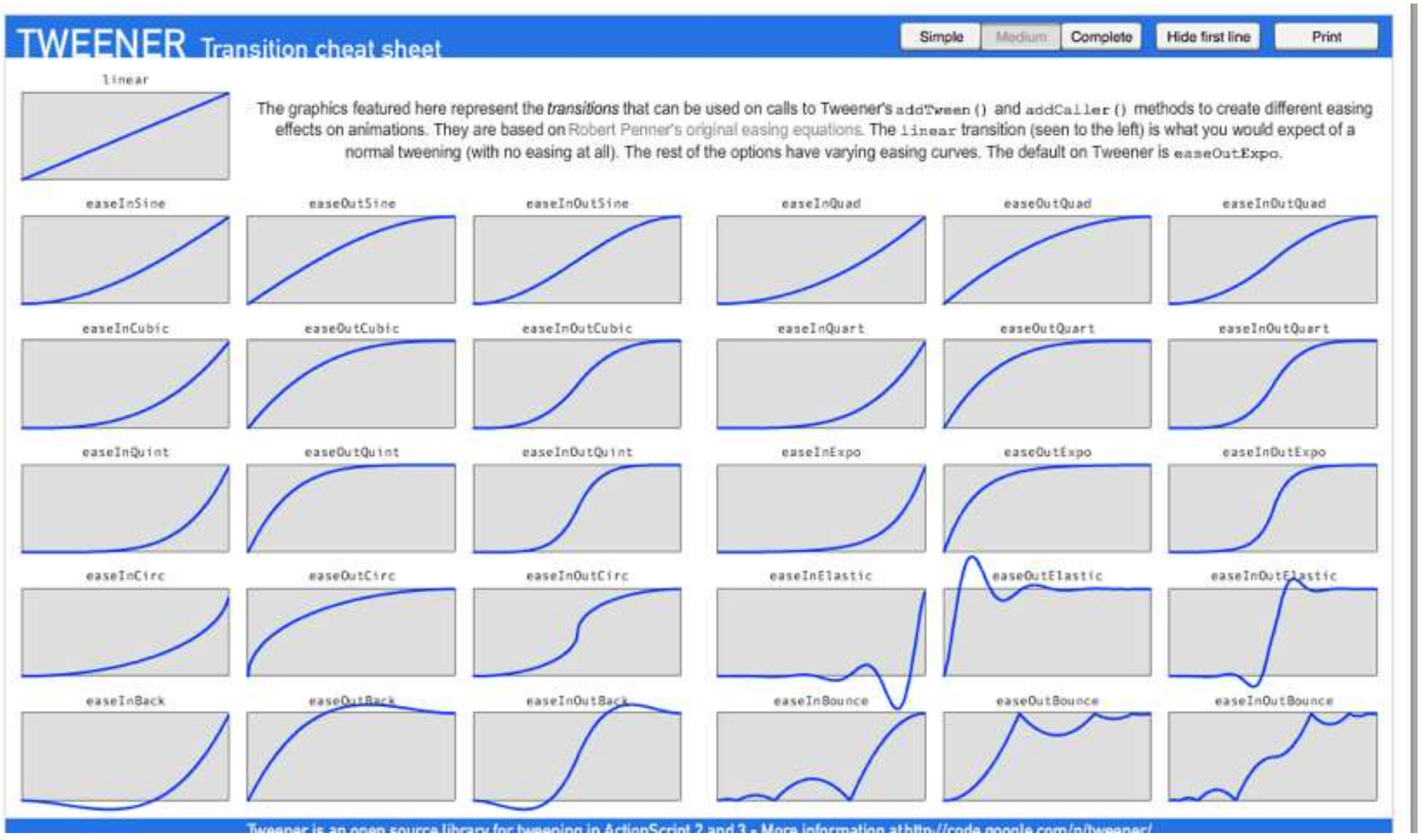
+

EASEIN
EASEOUT
EASEINOUT

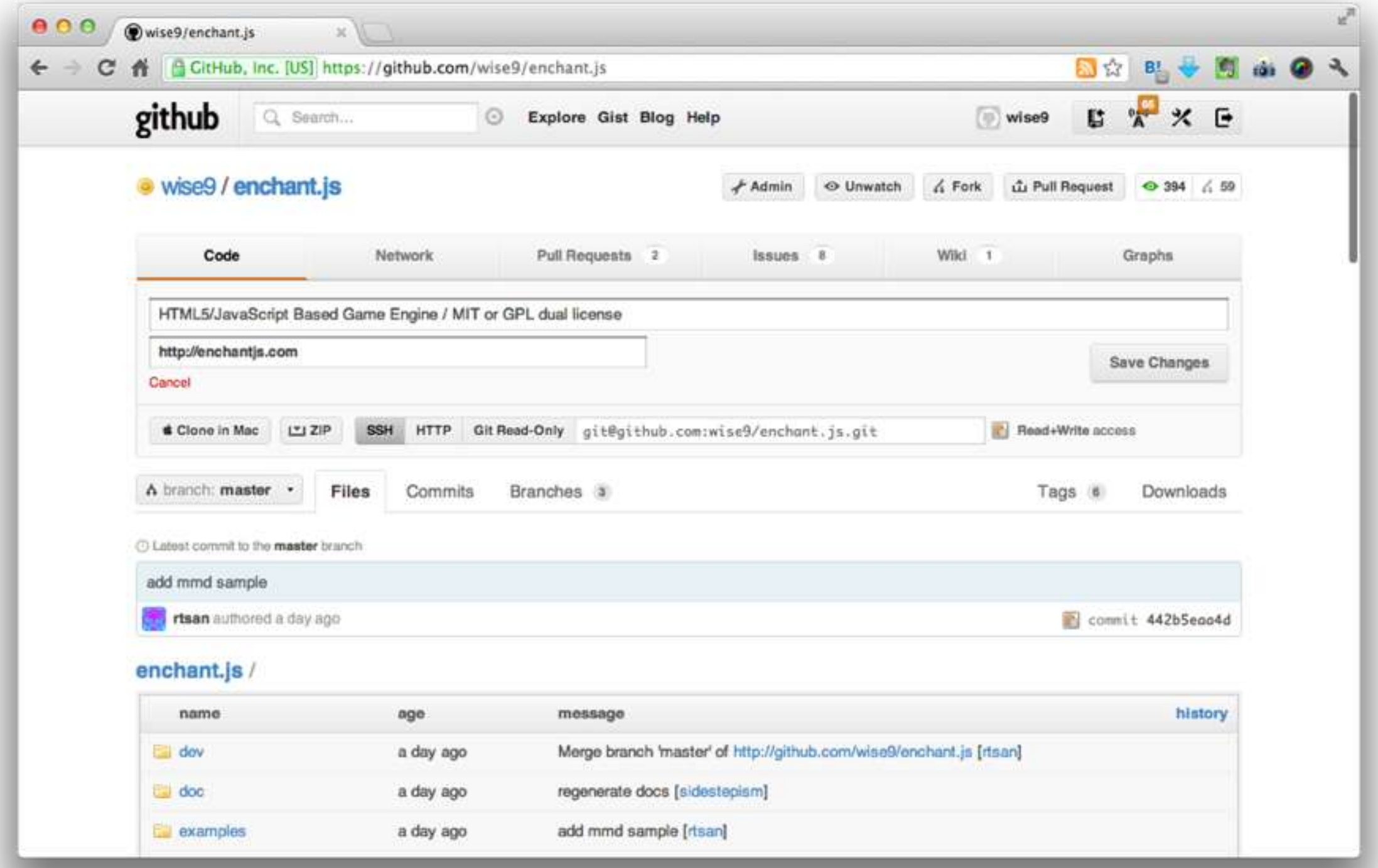
The Tweener Transition Cheat Sheet

http://bit.ly/3u7B6

explains these movements



As you can see, many different functions and features exist for animation and timelines in the animation engine of enchant.js.



Check it out on github if you haven’t already!

The enchant.js animation engine