Thomas Fuchs

Roll your own JavaScript Effects Framework

(and introducing Émile)

Émile Cohl

“The father of the animated cartoon”

EMILE COHL

Fantasmagorie

—1908—

http://www.youtube.com/watch?v=aEAObel8yIE

Animation & visual effects on webpages are superfluous and don’t add anything useful.

Animation & visual effects on webpages are superfluous, don’t add anything useful and are totally awesome.

Animation & visual effects on webpages are superfluous, don’t add anything useful and are $\boxed{\text{totally awesome.}}$

what’s important to the user?

emile.js

Simple (<50 lines of code)
CSS animations
Timing, chaining, easing
Stand-alone

http://github.com/madrobby/emile

```
// emile.js (c) 2009 Thomas Fuchs
// Licensed under the terms of the MIT license.

(function(emile, object){
  var parseEl = document.createElement('div'),
    props = ('backgroundColor borderBottomColor borderBottomWidth borderLeftColor borderLeftWidth '+
    'borderRightColor borderRightWidth borderSpacing borderTopColor borderTopWidth bottom color fontSize '+
    'fontWeight height left letterSpacing lineHeight marginBottom marginLeft marginRight marginTop maxHeight '+
    'maxWidth minHeight minWidth opacity outlineColor outlineOffset outlineWidth paddingBottom paddingLeft '+
    'paddingRight paddingTop right textIndent top width wordSpacing zIndex').split(' ');

  function parse(value){
    var v = parseFloat(value), u = value.replace(/^[\d\.]+/,'');
    return { value: isNaN(v) ? u : v, unit: isNaN(v) ? 'color' : u };
  }

  function normalize(style){
    var css, rules = {}, i = props.length, v;
    parseEl.innerHTML = '<div style="'+style+'"></div>';
    css = parseEl.childNodes[0].style;
    while(i--) if(v = css[props[i]]) rules[props[i]] = parse(v);
    return rules;
  }

  function color(source,target,pos){
    var i = 2, j, c, v = [], r = [];
    while(i--)
      if(arguments[i][0]=='r'){
        c = arguments[i].match(/\d+/g); j=3; while(j--) v.push(parseInt(c[j]));
      } else {
        c = arguments[i].substr(1); j=3; while(j--) v.push(parseInt(c.substr(j*2,2), 16));
      }
    j=3; while(j--) { tmp = ~~(v[j+3]+(v[j]-v[j+3])*pos); r.push(tmp<0?0:tmp>255?255:tmp); }
    return 'rgb('+r.join(',')+')';
  }

  (object||window)[emile] = function(el, style, opts){
    el = typeof el == 'string' ? document.getElementById(el) : el;
    opts = opts || {};
    var target = normalize(style), comp = el.currentStyle ? el.currentStyle : document.defaultView.getComputedStyle(el, null),
      prop, current = {}, start = (new Date).getTime(), dur = opts.duration||200, finish = start+dur, interval;
    for(prop in target) current[prop] = parse(comp[prop]);
    interval = setInterval(function(){
      var time = (new Date).getTime(), delta = time>finish ? 1 : (time-start)/dur;
      for(prop in target)
        el.style[prop] = target[prop].unit == 'color' ?
          color(current[prop].value,target[prop].value,delta) :
          (current[prop].value+(target[prop].value-current[prop].value)*delta).toFixed(3) + target[prop].unit;
      if(time>finish) { clearInterval(interval); opts.after && opts.after(); }
    },10);
  }
})('emile');
```

Wait, hold it!
Why write something new from scratch?

JavaScript frameworks

- “Best thing since sliced bread”
- Help you get stuff done more easily
- “Make JavaScript enjoyable”
- Fix cross-browser issues

JavaScript frameworks (BUT)

- Cover too much or too little
- Component and plugin hell
- Lead to uniformity
- Keep JavaScript away from you

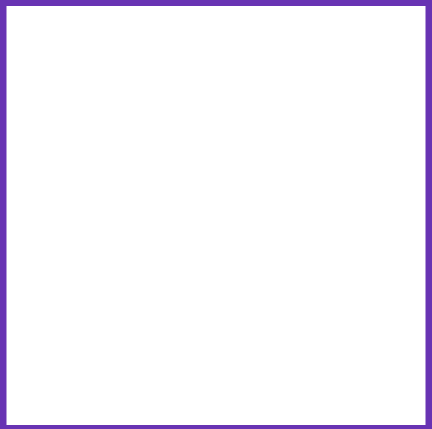
JavaScript frameworks (BUT BUT)

- Learn from them for your own code
- Pick parts you need
- Extend them for good or evil
- Be a JavaScript god/ninja/cowboy etc.

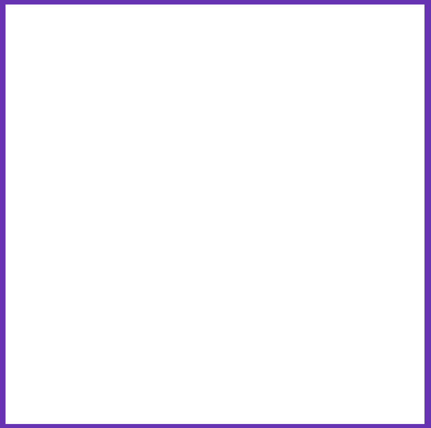
Animation! (what you came for)

- What to use for timing
- How to conquer CSS
- Performance?
- And how to make it really nice

Move a block from left to right and back



Move a block from left to right and back



Using a for loop

```
for (var i = 0; i < 1000; i++)
  element.style.left = i + 'px';

for (var j = 1000; j > 0; j--)
  element.style.left = j + 'px';
```

Using a for loop

moves block to right

```
for (var i = 0; i < 1000; i++)
  element.style.left = i + 'px';

for (var j = 1000; j > 0; j--)
  element.style.left = j + 'px';
```

moves block back to left

Using a for loop

```
for (var i = 0; i < 1000; i++)
  element.style.left = i + 'px';

for (var j = 1000; j > 0; j--)
  element.style.left = j + 'px';
```

surprise, this does nothing at all!

JavaScript and the browser rendering engine share a single thread of execution.

While the code is running, no rendering will happen.

setInterval

```
var direction = 1, i = 0,
  interval = setInterval(function(){
    i += direction;
    if(i == 1000) direction = -1;
    element.style.left = i + 'px';
    if(i < 0) clearInterval(interval);
  },10);
```

setInterval

1 = positive, -1 = negative

```
var direction = 1, i = 0,
  interval = setInterval(function(){
    i += direction;
    if(i == 1000) direction = -1;
    element.style.left = i + 'px';
    if(i < 0) clearInterval(interval);
  },10);
```

setInterval

```
var direction = 1, i = 0,
  interval = setInterval(function(){
    i += direction;
    if(i == 1000) direction = -1;
    element.style.left = i + 'px';
    if(i < 0) clearInterval(interval);
  }, 10);
```

call this function every 10ms

setInterval

```
var direction = 1, i = 0,
  interval = setInterval(function(){
    i += direction;
    if(i == 1000) direction = -1;
    element.style.left = i + 'px';
    if(i < 0) clearInterval(interval);
  },10);
```

increase or decrease the index

setInterval

```
var direction = 1, i = 0,
  interval = setInterval(function(){
    i += direction;
    if(i == 1000) direction = -1;
    element.style.left = i + 'px';
    if(i < 0) clearInterval(interval);
  },10);
```

reverse direction once we reach 1000

setInterval

```
var direction = 1, i = 0,
  interval = setInterval(function(){
    i += direction;
    if(i == 1000) direction = -1;
    element.style.left = i + 'px';
    if(i < 0) clearInterval(interval);
  },10);
```

set the style

setInterval

```
var direction = 1, i = 0,
  interval = setInterval(function(){
    i += direction;
    if(i == 1000) direction = -1;
    element.style.left = i + 'px';
    if(i < 0) clearInterval(interval);
  },10);
```

stop doing the animation when the index drops below 0

Much better, as in, there’s actually some animation going on.

But, there’s a problem:
it’s hardly exact timing to use the
10ms interval.

Not all users have the super-fast
laptops you all have,
or maybe they’re looking at it on a
mobile browser.

```
(new Date).getTime()
1257185326039
```

milliseconds since epoch
(January 1, 1970 00:00:00 UTC)

Epoch FTW

```
<div id="test" style="position:absolute">test</div>

<script type="text/javascript" charset="utf-8">
var element = document.getElementById('test');

var start = (new Date).getTime(), duration = 1000,
  finish = start+duration;

var interval = setInterval(function(){
  var time = (new Date).getTime(),
    pos = time>finish ? 1 : (time-start)/duration;
  element.style.left = (1000*pos) + 'px';
  if(time>finish) clearInterval(interval);
},10);
</script>
```

Epoch FTW

```
<div id="test" style="position:absolute">test</div>

<script type="text/javascript" charset="utf-8">
var element = document.getElementById('test');

var start = (new Date).getTime(), duration = 1000,
  finish = start+duration;

var interval = setInterval(function(){
  var time = (new Date).getTime(),
    pos = time>finish ? 1 : (time-start)/duration;
  element.style.left = (1000*pos) + 'px';
  if(time>finish) clearInterval(interval);
},10);
</script>
```

starts now, calculate finish time from duration (for now one second)

Epoch FTW

```
<div id="test" style="position:absolute">test</div>

<script type="text/javascript" charset="utf-8">
var element = document.getElementById('test');

var start = (new Date).getTime(), duration = 1000,
  finish = start+duration;

var interval = setInterval(function(){
  var time = (new Date).getTime(),
    pos = time>finish ? 1 : (time-start)/duration;
  element.style.left = (1000*pos) + "px";
  if(time>finish) clearInterval(interval);
},10);
</script>
```

calculate a position between 0 and 1
(0 = start of effect, 1 = end of effect)

"pos" is 0 at the animation's start, 1 at the animation's end

```
var time = (new Date).getTime(),
     pos = time>finish ?
       1 : (time-start)/duration;
```

```
var time = (new Date).getTime(),
     pos = time>finish ?
       1 : (time-start)/duration;
```

```
var time = (new Date).getTime(),
      pos = time>finish ?
        1 : (time-start)/duration;
```

start = 6039

duration = 1000 (1 second)

finish = start + duration = 7039

current time = 6539

```
var time = (new Date).getTime(),
      pos = time>finish ?
          1 : (time-start)/duration;
```

start = 6039

duration = 1000 (1 second)

finish = start + duration = 7039

current time = 6539

what's pos at 6539?

$$6039 \xrightarrow[t]{} 7039$$

```
var time = (new Date).getTime(),
     pos = time>finish ?
       1 : (time-start)/duration;
```

start = 6039

duration = 1000 (1 second)

finish = start + duration = 7039

current time = 6539

what's pos at 6539?

$$6039 \xrightarrow[t]{} 7039$$

(time-start)/duration =

(6539-6039)/1000 =

500/1000

= 0.5

Epoch FTW

```
<div id="test" style="position:absolute">test</div>

<script type="text/javascript" charset="utf-8">
var element = document.getElementById('test');

var start = (new Date).getTime(), duration = 1000,
  finish = start+duration;

var interval = setInterval(function(){
  var time = (new Date).getTime(),
    pos = time>finish ? 1 : (time-start)/duration;
  element.style.left = (1000*pos) + 'px';
  if(time>finish) clearInterval(interval);
},10);
</script>
```

use the position to calculate the style

The core loop is complete, but supporting only the CSS “left” property is boring.

So how do we query/set more CSS properties?

“It depends.”

Reading CSS properties

IE

```
computedStyle =
  element.currentStyle ? element.currentStyle :
    document.defaultView.getComputedStyle(element, null);
```

DOM

My thinking is, IE's currentStyle property is more elegant.

However:

```
> element.style.border = "2px solid green";
2px solid green

> document.defaultView.getComputedStyle(element,null).border
```

nothing returned?

However:

```
> element.style.border = "2px solid green";
2px solid green

> document.defaultView.getComputedStyle(element,null).border


> document.defaultView.getComputedStyle(element,null).borderLeftWidth
2px

> document.defaultView.getComputedStyle(element,null).borderLeftColor
rgb(0, 128, 0)
```

colors are normalized

shorthand properties are expanded

This means, to transform from

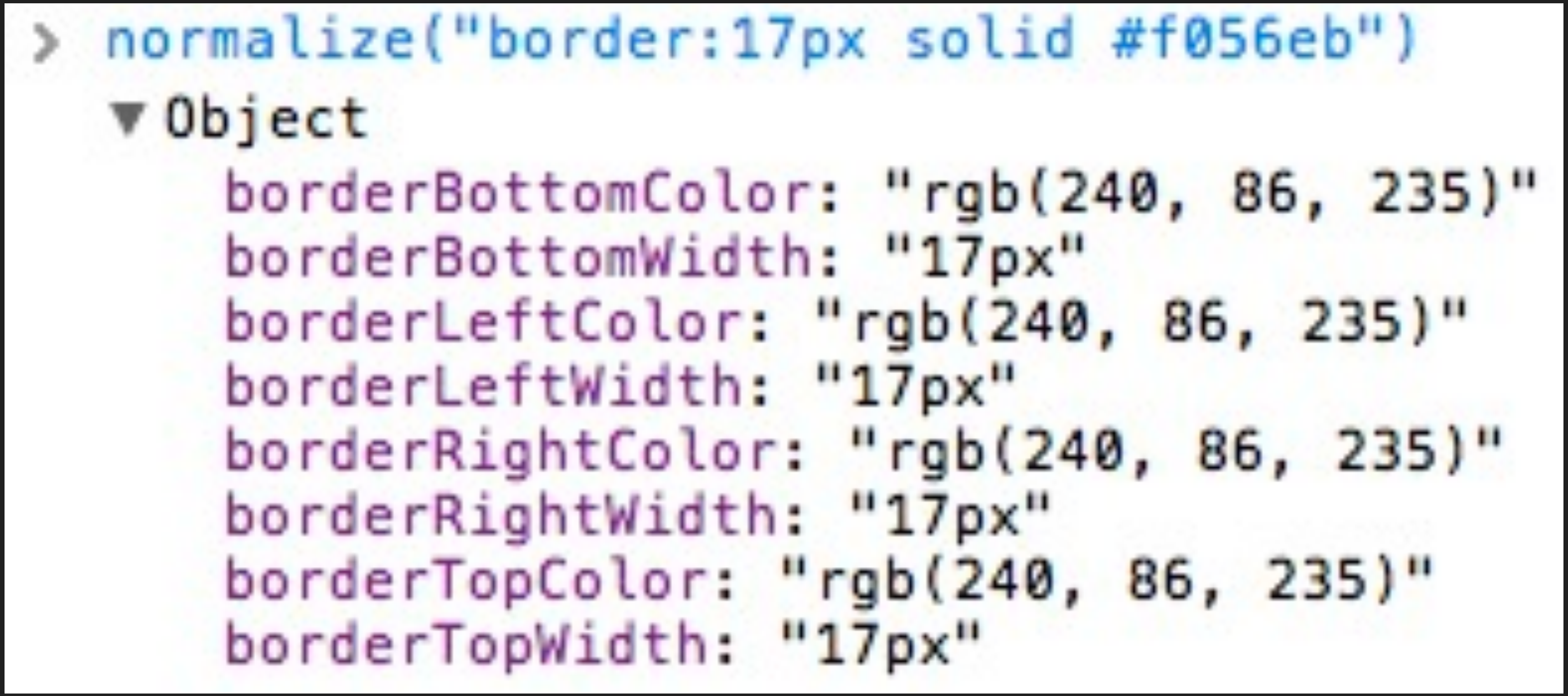
```
border:2px solid green;
```

to

```
border:17px solid #f056eb;
```

We need to expand/normalize the target properties.

Normalizing CSS properties



```
> normalize("border:17px solid #f056eb")
  ▼Object
     borderBottomColor: "rgb(240, 86, 235)"
     borderBottomWidth: "17px"
     borderLeftColor: "rgb(240, 86, 235)"
     borderLeftWidth: "17px"
     borderRightColor: "rgb(240, 86, 235)"
     borderRightWidth: "17px"
     borderTopColor: "rgb(240, 86, 235)"
     borderTopWidth: "17px"
```

Normalizing CSS properties

```
var parseEl = document.createElement('div'),
  props = ('backgroundColor borderBottomColor '+
   // imagine more lines with more CSS properties here
   'width wordSpacing zIndex').split(' ');

function normalize(style){
  var css, rules = {}, i = props.length, v;
  parseEl.innerHTML = '<div style="'+style+'"></div>';
  css = parseEl.childNodes[0].style;
  while(i--) if(v = css[props[i]]) rules[props[i]] = v;
  return rules;
}
```

Normalizing CSS properties

create a DIV, to give the browser the hard work

```
var parseEl = document.createElement('div'),
  props = ('backgroundColor borderBottomColor '+
   // imagine more lines with more CSS properties here
   'width wordSpacing zIndex').split(' ');

function normalize(style){
  var css, rules = {}, i = props.length, v;
  parseEl.innerHTML = '<div style="'+style+'"></div>';
  css = parseEl.childNodes[0].style;
  while(i--) if(v = css[props[i]]) rules[props[i]] = v;
  return rules;
}
```

Normalizing CSS properties

define a list of possible properties

```
var parseEl = document.createElement('div'),
  props = ('backgroundColor borderBottomColor '+
   // imagine more lines with more CSS properties here
   'width wordSpacing zIndex').split(' ');

function normalize(style){
  var css, rules = {}, i = props.length, v;
  parseEl.innerHTML = '<div style="'+style+'"></div>';
  css = parseEl.childNodes[0].style;
  while(i--) if(v = css[props[i]]) rules[props[i]] = v;
  return rules;
}
```

Normalizing CSS properties

```
var parseEl = document.createElement('div'),
  props = ('backgroundColor borderBottomColor '+
   // imagine more lines with more CSS properties here
   'width wordSpacing zIndex').split(' ');

function normalize(style){
  var css, rules = {}, i = props.length, v;
  parseEl.innerHTML = '<div style="'+style+'"></div>';
  css = parseEl.childNodes[0].style;
  while(i--) if(v = css[props[i]]) rules[props[i]] = v;
  return rules;
}
```

create a new element with
the CSS properties we want to have normalized

Normalizing CSS properties

```
var parseEl = document.createElement('div'),
  props = ('backgroundColor borderBottomColor '+
   // imagine more lines with more CSS properties here
   'width wordSpacing zIndex').split(' ');

function normalize(style){
  var css, rules = {}, i = props.length, v;
  parseEl.innerHTML = '<div style="'+style+'"></div>';
  css = parseEl.childNodes[0].style;
  while(i--) if(v = css[props[i]]) rules[props[i]] = v;
  return rules;
}
```

like getComputedStyle(), the style property of an element contains normalized CSS properties

Normalizing CSS properties

```
var parseEl = document.createElement('div'),
  props = ('backgroundColor borderBottomColor '+
   // imagine more lines with more CSS properties here
   'width wordSpacing zIndex').split(' ');

function normalize(style){
  var css, rules = {}, i = props.length, v;
  parseEl.innerHTML = '<div style="'+style+'"></div>';
  css = parseEl.childNodes[0].style;
  while(i--) if(v = css[props[i]]) rules[props[i]] = v;
  return rules;
}
```

slightly optimized way of “for all properties on our list, check if it’s defined, and if yes, add it to the rules object”

Interpolating values and colors from A to B

Interpolating between two CSS values

```
origin + difference × position
```

Interpolating between two CSS values

origin + difference × position

```
origin = ‘12px’
```

Interpolating between two CSS values

origin + difference × position

```
origin = ‘12px’
target = ‘20px’
```

Interpolating between two CSS values

origin + difference × position

```
origin = ‘12px’
target = ‘20px’
position = 0.5
```

Interpolating between two CSS values

origin + difference × position

```
origin = ‘12px’
target = ‘20px’
position = 0.5

12 + (20-12) × 0.5 =
```

Interpolating between two CSS values

origin + difference × position

```
origin = '12px'
target = '20px'
position = 0.5

12 + (20-12) × 0.5 =
12 + 8 × 0.5 =
```

Interpolating between two CSS values

origin + difference × position

```
origin = ‘12px’
target = ‘20px’
position = 0.5

12 + (20-12) × 0.5 =
12 + 8 × 0.5 =
12 + 4 = 16
```

Interpolating between two colors

```
function color(source,target,pos){
  var i = 2, j, c, tmp, v = [], r = [];
  while(i--)
    if(arguments[i][0]=='r'){
      c = arguments[i].match(/\d+/g); j=3; while(j--) v.push(parseInt(c[j]));
    } else {
      c = arguments[i].substr(1); j=3; while(j--) v.push(parseInt(c.substr(j*2,2), 16));
    }
  j=3; while(j--) { tmp = ~~(v[j+3]+(v[j]-v[j+3])*pos); r.push(tmp<0?0:tmp>255?255:tmp); }
  return 'rgb('+r.join(',')+')';
}
```

looks complicated, but it really only is interpolating for each color component (red, green, blue) individually.

Also…

```
function color(source,target,pos){
  var i = 2, j, c, tmp, v = [], r = [];
  while(i--)
    if(arguments[i][0]=='r'){
      c = arguments[i].match(/\d+/g); j=3; while(j--) v.push(parseInt(c[j]));
    } else {
      c = arguments[i].substr(1); j=3; while(j--) v.push(parseInt(c.substr(j*2,2), 16));
    }
  j=3; while(j--) { tmp = ~~(v[j+3]+(v[j]-v[j+3])*pos); r.push(tmp<0?0:tmp>255?255:tmp); }
  return 'rgb('+r.join(',')+')';
}
```

This JavaScript snippet is optimized for code size, not for readability. It could be expressed much more elegantly.

JavaScript numbers

```
> 0.1
0.1
> 0.0001
0.0001
> 0.0000001
1e-7
```

string representation

JavaScript numbers

```
font-size: 1e-7px
```

doesn’t work in CSS

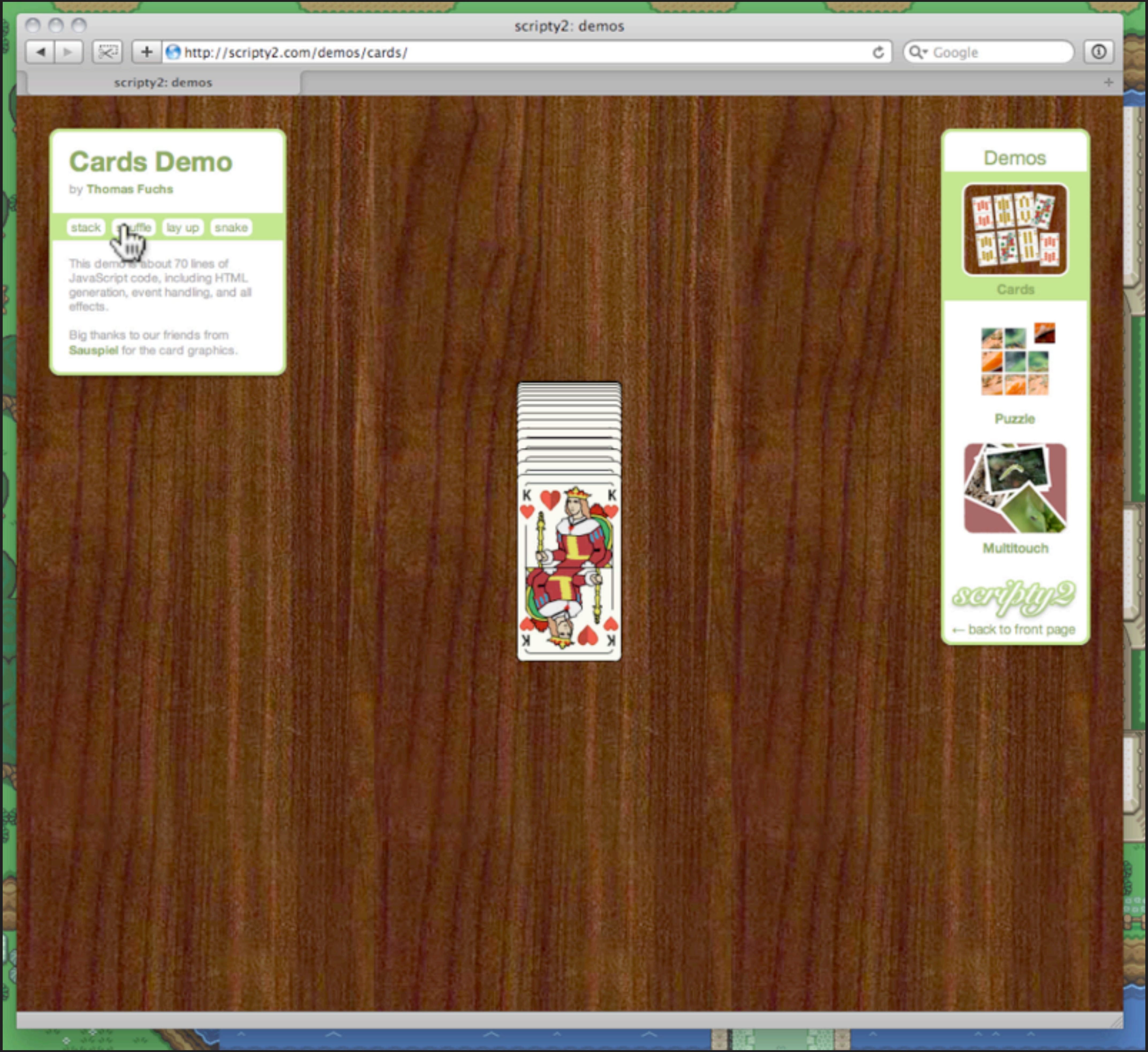
```
number.toFixed(3)
```

toFixed(3) round the number to 3 decimal places and and prevents an error

Optimizing rendering speed

Reduce the amount of nodes (HTML elements and text nodes) and avoid using the “opacity” CSS property.

And finally… easing.



Cards Demo

by **Thomas Fuchs**

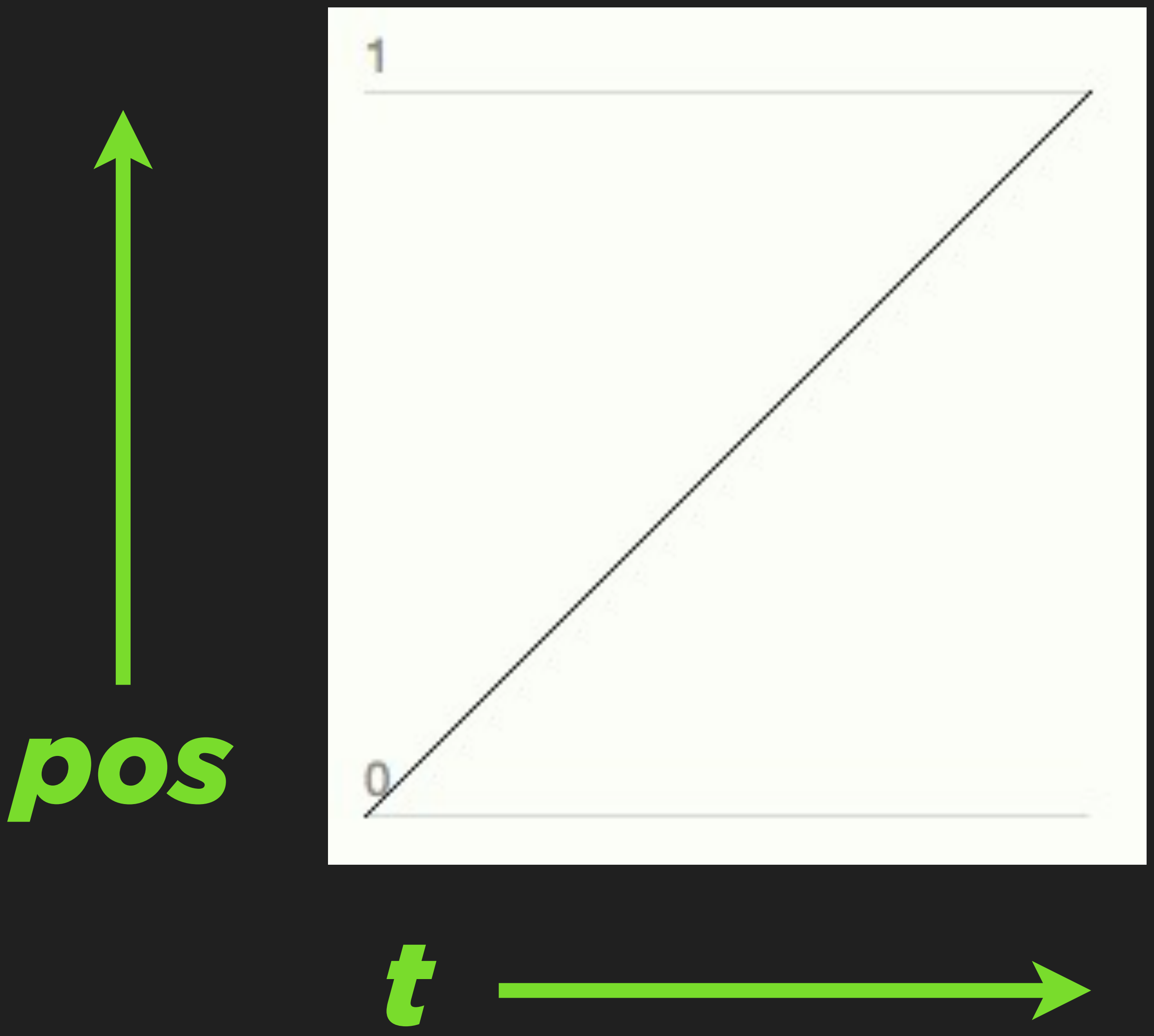
This demo is about 70 lines of JavaScript code, including HTML generation, event handling, and all effects.

Big thanks to our friends from **Sauspiel** for the card graphics.

“pos” is 0 at the animation’s start, 1 at the animation’s end

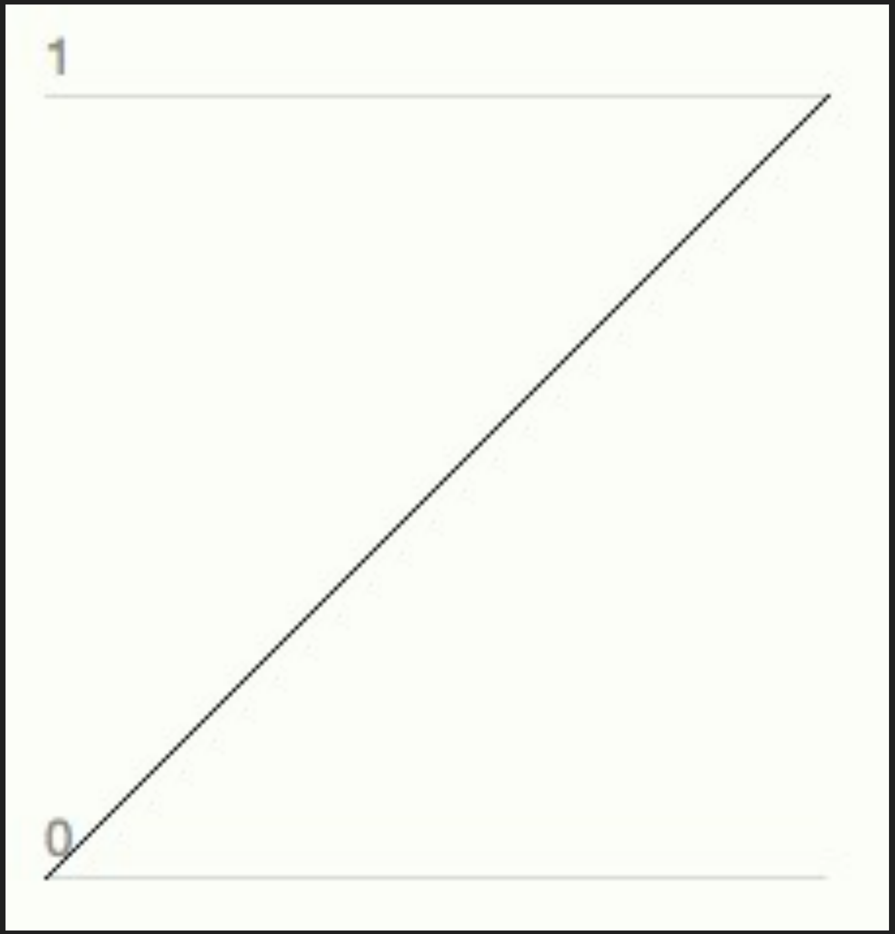
```
var time = (new Date).getTime(),
  pos = time>finish ?
    1 : (time-start)/duration;
```

No easing



No easing

sudden change
in velocity
at end



sudden change
in velocity at start

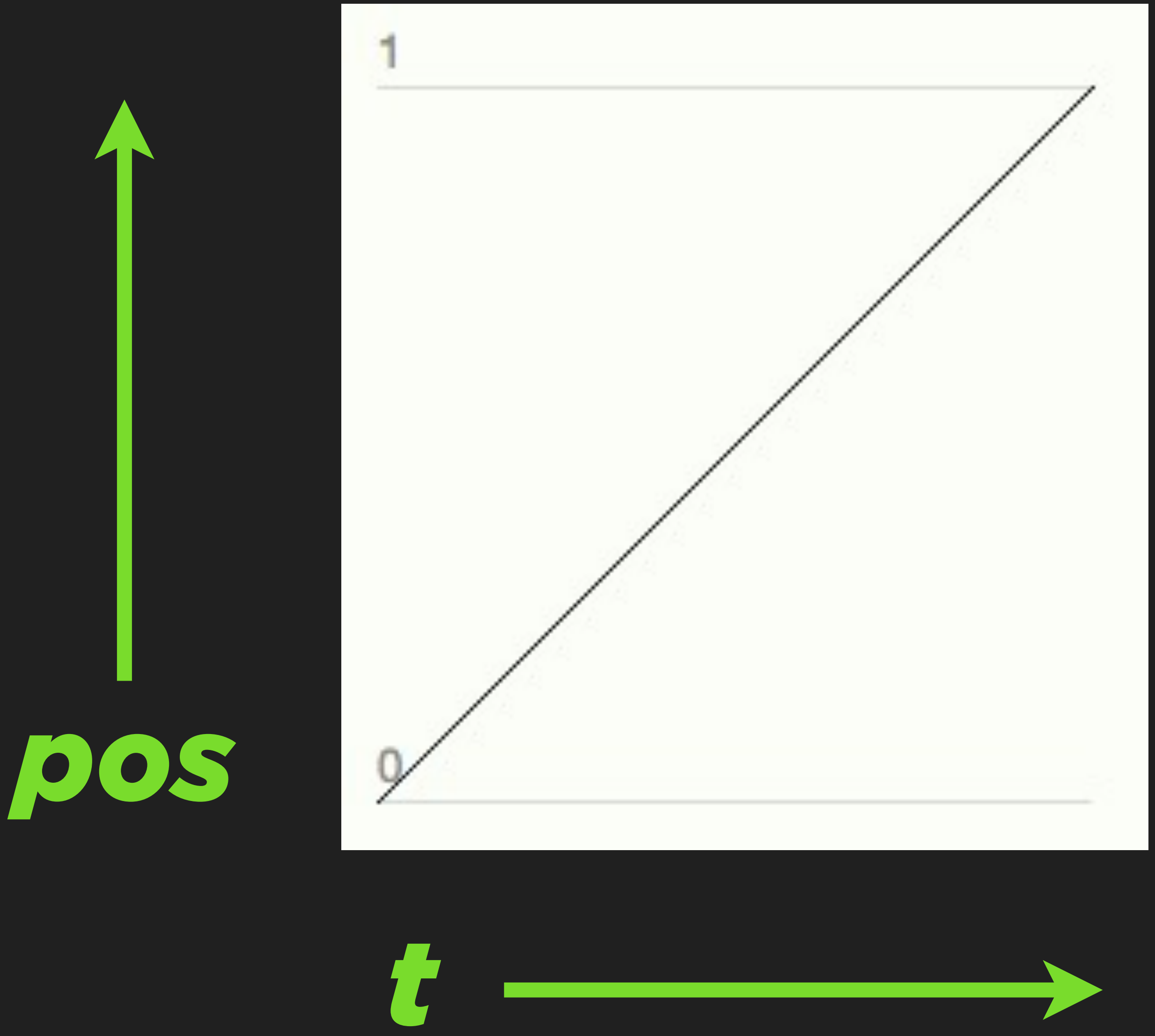
Easing is nothing more than messing with “pos”

```
emile('test2', 'left:300px;padding:10px;border:50px solid #ff0000', {
  duration: 500,
  after: function(){
    emile('test1', 'background:#0f0;left:100px;padding-bottom:100px;opacity:1', {
      duration: 4000, easing: bounce
    });
  }
});
```

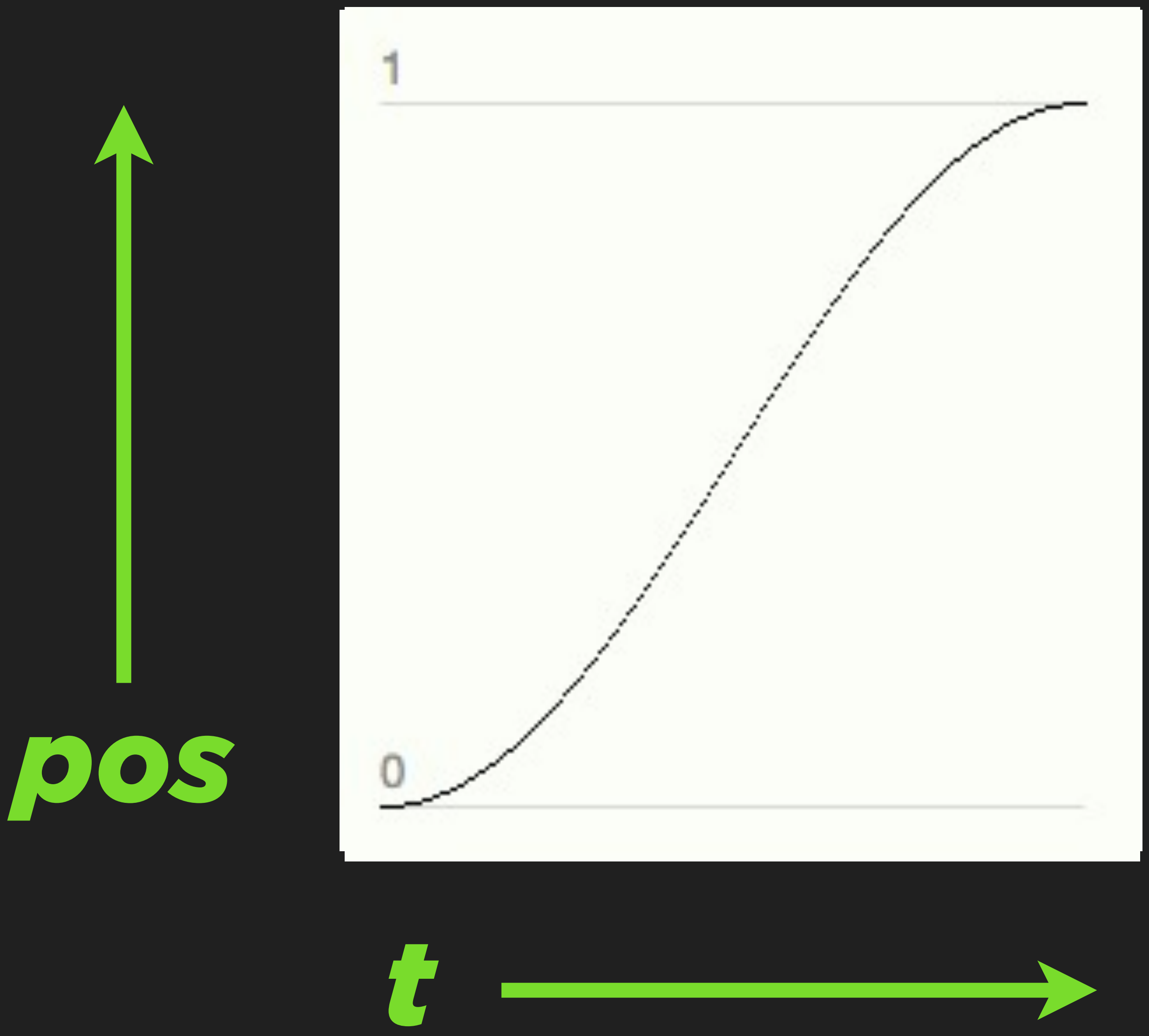
No easing looks unnatural.

Things move by accelerating and stop by decelerating.

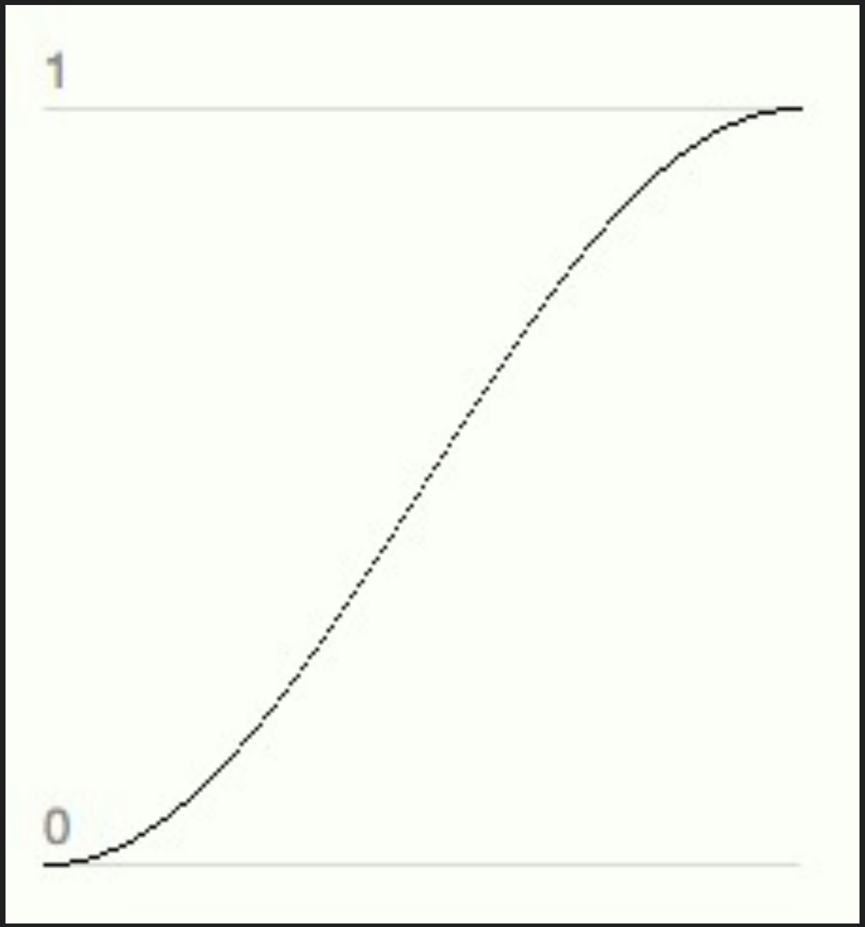
(-Math.cos(pos*Math.PI)/2) + 0.5



(-Math.cos(pos*Math.PI)/2) + 0.5



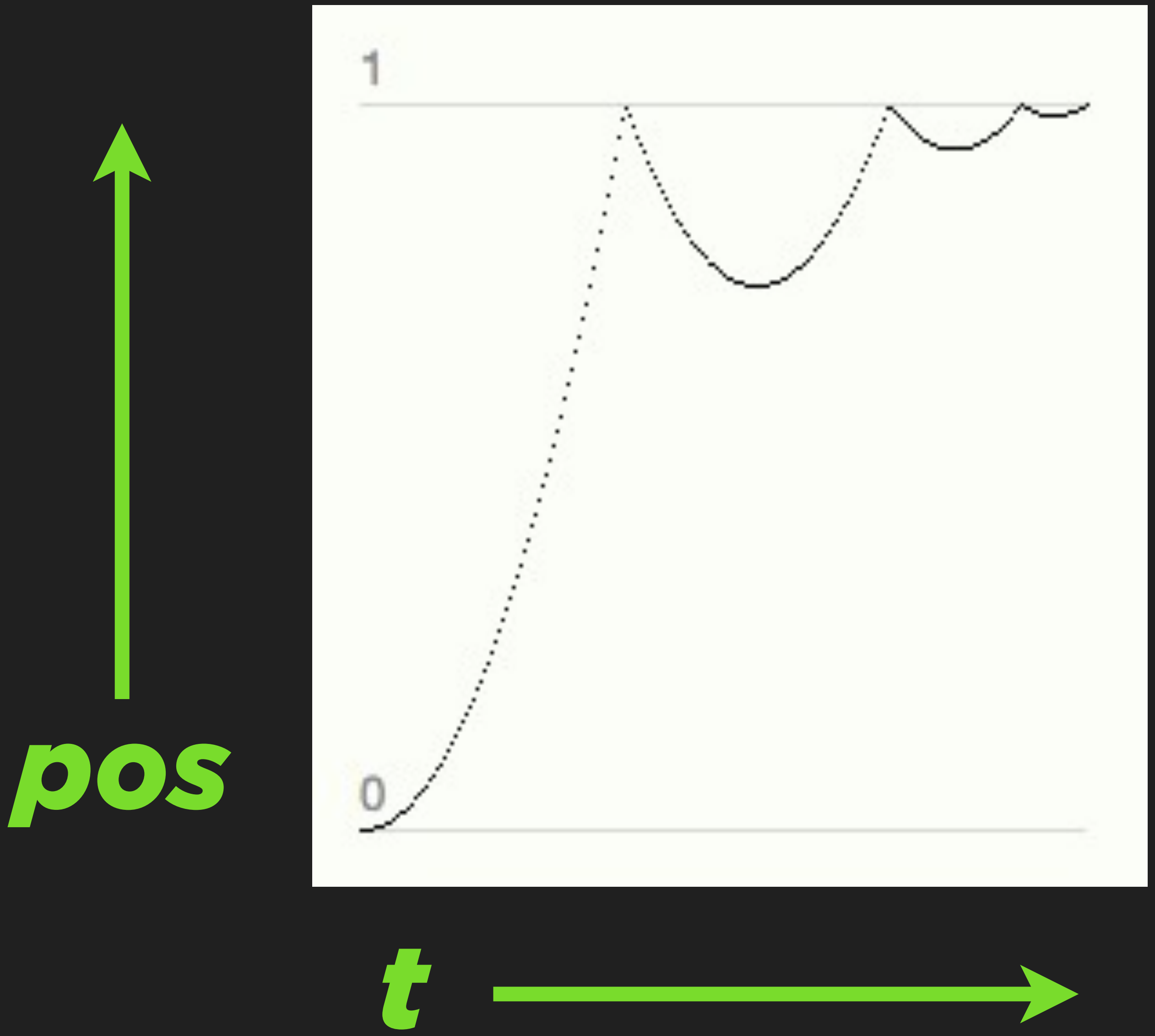
(-Math.cos(pos*Math.PI)/2) + 0.5

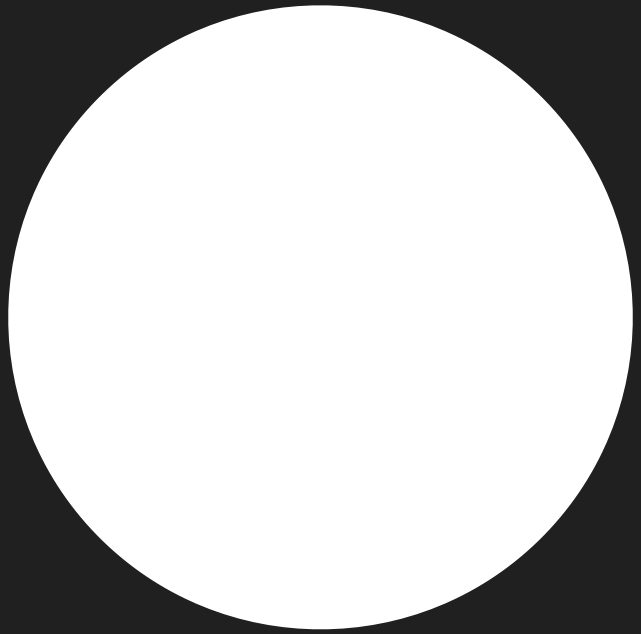


deceleration
at end

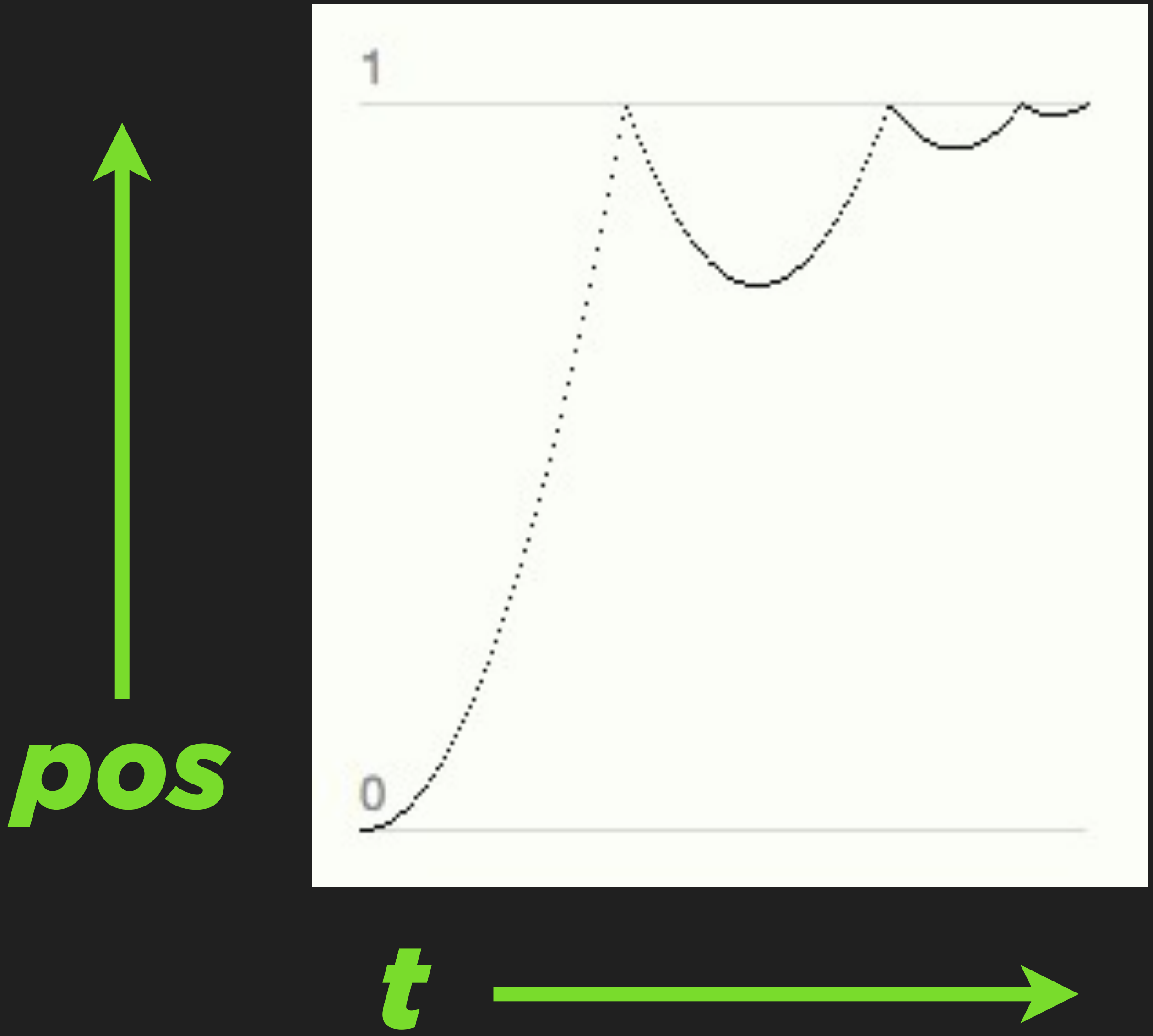
acceleration
at start

A “bounce” easing

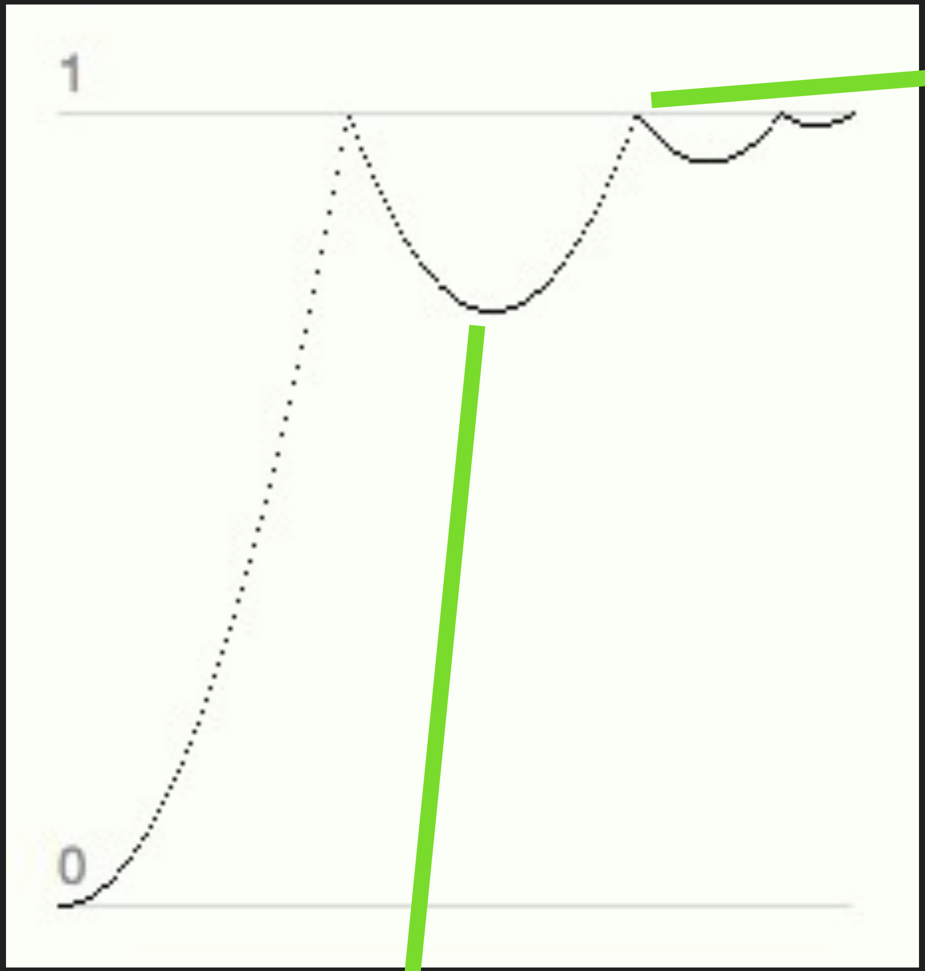




A “bounce” easing



A “bounce” easing



hard velocity changes

quadratic “gravity”

A “bounce” easing

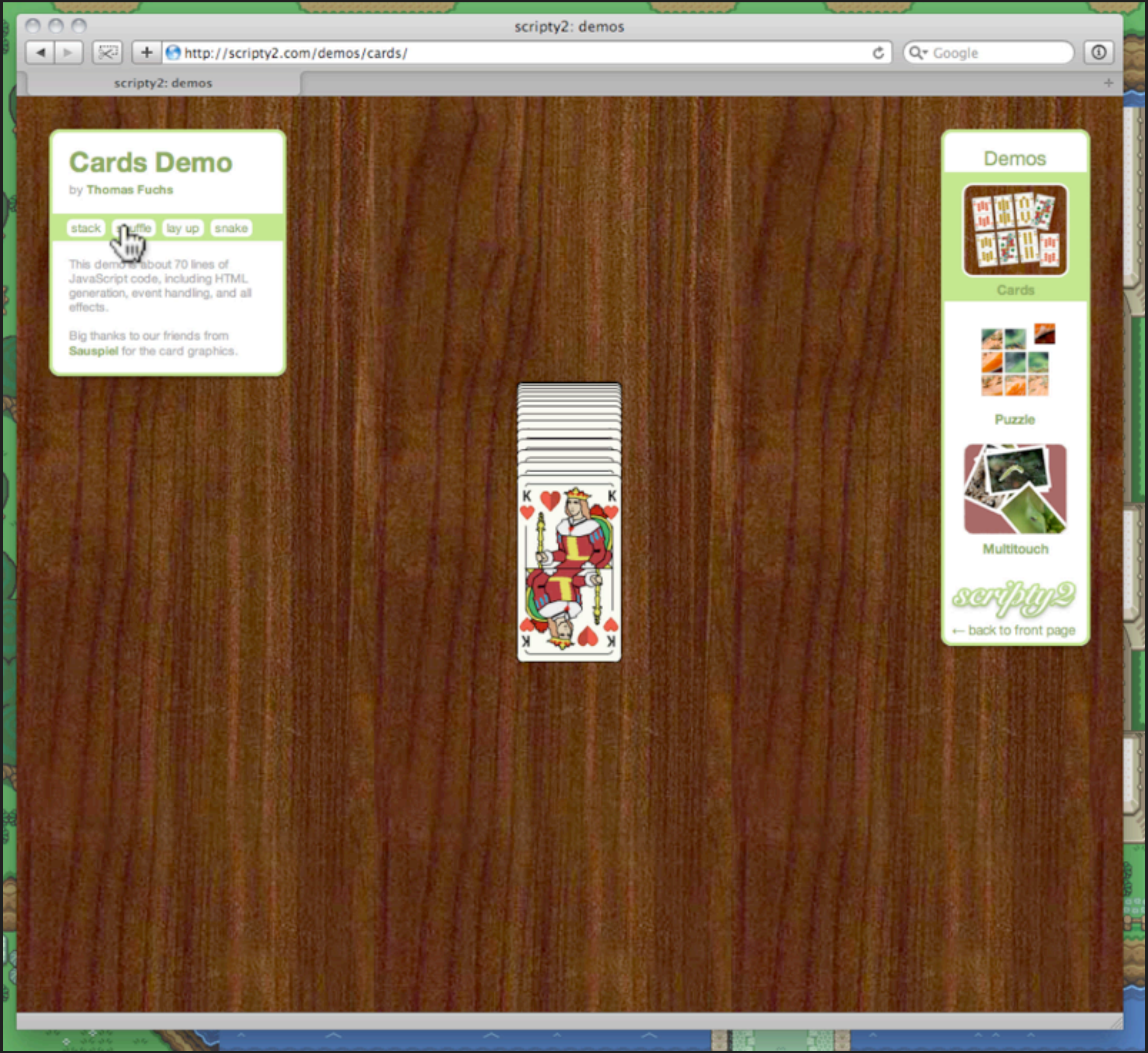
```
function bounce(pos) {
   if (pos < (1/2.75)) {
       return (7.5625*pos*pos);
   } else if (pos < (2/2.75)) {
       return (7.5625*(pos-=(1.5/2.75))*pos + .75);
   } else if (pos < (2.5/2.75)) {
       return (7.5625*(pos-=(2.25/2.75))*pos + .9375);
   } else {
       return (7.5625*(pos-=(2.625/2.75))*pos + .984375);
   }
 }
```

```
emile('test2', 'left:300px;padding:10px;border:50px solid #ff0000', {
  duration: 500,
  after: function(){
    emile('test1',
      'background:#0f0;left:100px;padding-bottom:100px;opacity:1', {
      duration: 4000, easing: bounce
    });
  }
});
```

st

```
emile('test2', 'left:300px;padding:10px;border:50px solid #ff0000', {
  duration: 500,
  after: function(){
    emile('test1',
      'background:#0f0;left:100px;padding-bottom:100px;opacity:1', {
      duration: 4000, easing: bounce
    });
  }
});
```

Easing animated CSS properties individually



Cards Demo

by Thomas Fuchs

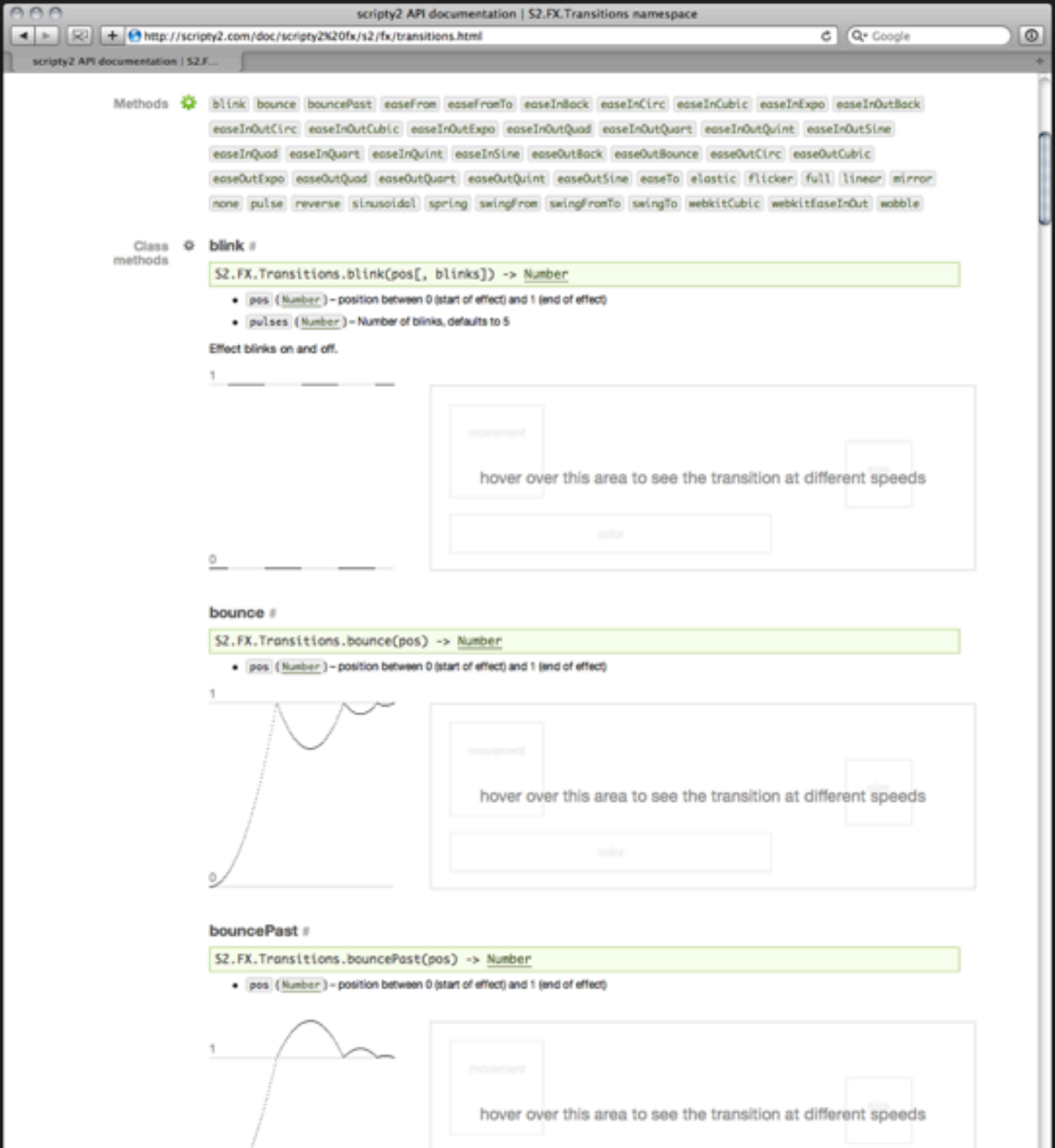
This demo is about 70 lines of JavaScript code, including HTML generation, event handling, and all effects.

Big thanks to our friends from Sauspiel for the card graphics.

```
propertyTransitions: {
  marginLeft: 'mirror',
  marginTop: 'bouncePast',
  left: 'swingFromTo',
  zIndex: zIndexTransition
}
```

scripty2 code – Not supported by Émile, too specialized. But easy to add.

scripty2 has tons of easings you can lift and use in your own apps



scripty2 API documentation | S2.FX.Transitions namespace

http://scripty2.com/doc/scripty2%20fx/s2/fx/transitions.html

Methods: blink bounce bouncePast easeFrom easeFromTo easeInBack easeInCirc easeInCubic easeInExpo easeInOutBack easeInOutCirc easeInOutCubic easeInOutExpo easeInOutQuad easeInOutQuart easeInOutQuint easeInOutSine easeInQuad easeInQuart easeInQuint easeInSine easeOutBack easeOutBounce easeOutCirc easeOutCubic easeOutExpo easeOutQuad easeOutQuart easeOutQuint easeOutSine easeTo elastic flicker full linear mirror none pulse reverse sinusoidal spring swingFrom swingFromTo swingTo webkitCubic webkitEaseInOut wobble

Class methods

blink

```
S2.FX.Transitions.blink(pos[, blinks]) -> Number
```

- pos (Number) – position between 0 (start of effect) and 1 (end of effect)
- pulses (Number) – Number of blinks, defaults to 5

Effect blinks on and off.

hover over this area to see the transition at different speeds

bounce

```
S2.FX.Transitions.bounce(pos) -> Number
```

- pos (Number) – position between 0 (start of effect) and 1 (end of effect)

hover over this area to see the transition at different speeds

bouncePast

```
S2.FX.Transitions.bouncePast(pos) -> Number
```

- pos (Number) – position between 0 (start of effect) and 1 (end of effect)

hover over this area to see the transition at different speeds

Demo them at

http://tr.im/E0JS

Q&A

And thanks!

http://github.com/madrobby/emile
http://scripty2.com/

Slides: http://mir.aculo.us/ (soon)