

AWA1

ANIMATION WEB

Gsap

[GreenSock Animation Platform]

Contenu

1. Introduction
2. TweenLite
3. TweenLite Transformations
4. TweenMax
5. TimelineLite
6. TimelineMax
7. DrawSVGPlugin
8. MorphSVGPlugin
9. SplitText
10. Draggable

1. Introduction Gsap

Qu'est-ce que Gsap ?

- Gsap, acronyme de 'GreenSock Animation Platform'.
- TweenLite: Noyau de Gsap, léger, peut être complété par des plugins.
Permet d'animer les attributs des objets.
- TweenMax = TweenLite + plusieurs plugins, complet mais plus lourd.
<http://cdnjs.cloudflare.com/ajax/libs/gsap/latest/TweenMax.min.js>
- TimelineLite : Outil de séquencage pour les tweens, léger, sans plugins
- TimelineMax : TimelineLite + fonctions
- Plugins payants : morphSVGplugin, drawSVGplugin, throwPropsplugin, ...

2. TweenLite : exemple

```
<script src="js/TweenLite.min.js"></script>
<script src="js/CSSPlugin.min.js"></script>
<script src="js/EasePack.min.js"></script>
<script>
window.onload = function() {
var photo = document.getElementById("photo");

TweenLite.to(photo, 2, { width:"100px"});

}

// TweenLite.to( élément a changer, durée, {propriétés} );
```

2. TweenLite : éléments

```
// Tween l'element avec l'id "myID" inclure jQuery
```

```
TweenLite.to("#myID", 2, {backgroundColor:"#ff0000", width:"50%", top:"100px"});
```

```
// Tween les éléments de la class "myClass" inclure jQuery
```

```
TweenLite.to(".myClass", 2, {boxShadow:"0px 0px 20px red", color:"#FC0"});
```

```
// Tween un array d'objets
```

```
TweenLite.to([obj1, obj2, obj3], 1, {opacity:0.5, rotation:45});
```

2. TweenLite : set

// Définit, ou redéfinit les propriétés d'un élément

```
TweenLite.set("#letterN", {  
  scale:0.8,  
  opacity:0,  
  transformOrigin:'50% 50%'  
});
```

```
TweenLite.to("#letterN",1, { scale:1, opacity:1 });
```

2. TweenLite : ease (accélération)

Exemple pour easeOut:

```
var photo = document.getElementById("photo");  
TweenLite.to(photo, 1, {width:100, ease:Bounce.easeOut});
```

Ease les plus utilisés:

Power1..4, Back, Bounce, Elastic, ...

[voir GSAP ease visualiser ->](#)

2. TweenLite + jQuery

```
var $box = $('.box');  
  
$box.hover(  
    function() {  
        TweenLite.to($(this), 0.3, {scale:1.2});  
    },  
    function() {  
        TweenLite.to($(this), 0.15, {scale:1});  
    }  
);
```

// Agrandit l'élément courant quand on le survole, puis revient a la taille originale sur mouseout

2. TweenLite + jQuery

```
var $box = $('.box');  
  
$box.on('click', function(e){  
    TweenLite.to($(this), 0.3, {x:'+=100px'});  
});
```

// Quand on clique l'élément courant, il se déplace de 100px de plus à gauche

2. TweenLite : from, to, fromTo

```
var photo = $("#photo");
```

```
TweenLite.to(photo, 2, {rotation:30, scaleX:0.8});
```

// Anime de 0 aux valeurs actuelles de scaleX/scaleY

```
TweenLite.from(photo, 1.5, {scaleX:0, scaleY:0});
```

```
TweenLite.from(photo, 2, {opacity:0, top:"100px"});
```

// Anime de width 0 a 100 et largeur 0 a 200

```
TweenLite.fromTo(photo, 1.5, {width:0, height:0}, {width:100, height:200});
```

2. TweenLite : référence + actions

```
var myTween = TweenLite.to(element, 2, {width:200, height:150}, paused:true);
```

```
btnStartTween.onclick = function() { myTween.start(); };
```

Actions :

```
myTween.play()
```

```
myTween.pause()
```

```
myTween.resume() // reprendre depuis la pause
```

```
myTween.restart() // recommencer au début
```

```
myTween.reverse()
```

```
myTween.seek(0.5) // commence depuis 0.5s
```

```
tween.timeScale(2); // ralenti 2x
```

2. TweenLite : callbacks (exemple)

```
TweenLite.to(photo, 2, {left:"300px",  
    onUpdate:updateHandler,  
    onComplete:completeHandler,  
    onCompleteParams:["animation complete!"]});
```

```
function updateHandler() {  
    updateCount++;  
    updateOutput.innerHTML = "update : " + updateCount;  
}
```

```
function completeHandler(message) {  
    completeOutput.innerHTML = "fin : " + message;  
}
```

2. TweenLite : autoAlpha

AutoAlpha combine opacity et visibility. Utile pour qu'élément soit pas cliquable et améliore performances de rendu.

// Fade out et set visibility:hidden

```
TweenLite.to(element, 2, {autoAlpha:0});
```

// En 2 seconds, fade back et visibility:visible

```
TweenLite.to(element, 2, {autoAlpha:1, delay:2});
```

3. TweenLite : transformations

// Exemple transform 2D

```
TweenLite.to(element, 2, {rotation:+=30, scaleX:0.8});
```

// Exemple transform 3D (rotationX, rotationY, rotationZ , z, perspective,
et transformPerspective

```
TweenLite.to(element, 2, {rotationX:45, scaleX:0.8, z:-300});
```

3. TweenLite : transformations

CSS3 3D Transforms - WD

Method of transforming an element in the third dimension using the **transform** property. Includes support for the **perspective** property to set the perspective in z-space and the **backface-visibility** property to toggle display of the reverse side of a 3D-transformed element.

Global 81.81% + 8.06% = 89.87%
unprefixed: 63.33% + 8.06% = 71.39%

Current aligned Usage relative Show all

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Chrome for Android
								4.1	
8			43					4.3	
9		40	44					4.4	
¹ 10	12	41	45	8	32	8.4		4.4.4	
¹ 11	13	42	46	² 9	33	9.1	8	46	46
	14	43	47		34				
		44	48		35				
		45	49						

Notes Known issues (5) Resources (10) Feedback

¹ Partial support in IE refers to not supporting the **transform-style: preserve-3d** property. This prevents nesting 3D transformed elements.

² Safari 9 is reported to still require a prefix for the related **backface-visibility** property.

3. TweenLite : transformations

transformPerspective



no transformPerspective

No visual distortion at all. Impossible to distinguish vanishing point or depth. DOM elements by default have no transformPerspective.



transformPerspective:200

The lower the transformPerspective, the more extreme the distortion.



transformPerspective:600

With a higher value the 3D effect is less pronounced.

3. TweenLite : transformations

transformPerspective Vs perspective

transformPerspective is applied to each box causing each box to have its own vanishing point



A single **perspective** is applied to the parent div of all the boxes causing each box to share the same vanishing point



3. TweenLite : transformations

// Appliquer la perspective a l'élément PARENT (container) pour que la perspective s'applique aux éléments enfant.

```
TweenLite.set(container, {perspective:500});
```

// Appliquer la perspective a tous les éléments qu'on tween.

```
CSSPlugin.defaultTransformPerspective = 500;
```

// Appliquer la perspective a 1 seul élément avec "transformPerspective"

```
TweenLite.set(element, {transformPerspective:500});
```

3. TweenLite : transformations

// css:

```
.myClass { transform: scale(1.5, 1.5) rotateY(45deg) translate3d(10px, 0px, -200px) }
```

// transformation GSAP equivalente (animée sur 2s):

```
TweenLite.to(element, 2, {scale:1.5, rotationY:45, x:10, y:0, z:-200});
```

// css avec perspective():

```
.myClass { transform: perspective(500px) rotate(120deg) translateY(50px) }
```

// transformation GSAP equivalente (settée, pas animée):

```
TweenLite.set(element, {transformPerspective:500, rotation:120, y:50});
```

3. TweenLite : transformations

transformOrigin

The negative z-index (-200) set in the transformOrigin properties of the second animation changes the effect drastically.

```
TweenMax.to(box1, 3, {rotationY:360, transformOrigin:"left top"})
```



```
TweenMax.to(box2, 3, {rotationY:360, transformOrigin:"left 50% -200"})
```



<http://codepen.io/GreenSock/pen/Ltfdb>

4. TweenMax

TweenMax = TweenLite + de nombreux plugins (CSSPlugin, BezierPlugin, RoundPropsPlugin, TimelineLite et TimelineMax).

Avantage : Pratique, un seul fichier à charger.

Inconvénient : Plus lourd que TweenLite.

4. TweenMax : staggerTo

```
var $box = $('.box');
```

```
TweenMax.staggerTo(".box", 1, {rotation:360, y:100}, 0.5);
```

// Le paramètre stagger (0.5) définit le temps de départ entre chaque animation.

<http://codepen.io/MAW/pen/azOogg>

// stagger divs

<http://codepen.io/GreenSock/pen/yelzn>

// staggerFrom drop menu

<http://codepen.io/grayghostvisuals/pen/RNLgJP>

// polyLion

4. TweenMax : repeat + yoyo

```
// Répétition du déplacement 4x en 'yoyo' avec 1s entre les déplacements  
TweenMax.to(logo, 1, {left:"300px", repeat:4, repeatDelay:1, yoyo:true});
```

<http://codepen.io/natewiley/pen/GJVOVw>

<http://codepen.io/MAW/pen/jEmpoJ>

// particules

// menu accordéon horiz

5. TimelineLight : base

```
var tl = new TimelineLite({paused:true});  
tl.add( TweenLite.to(element, 1, {left:100}) );  
tl.add( TweenLite.to(element, 1, {top:50}) );  
tl.add( TweenLite.to(element, 1, {opacity:0}) );
```

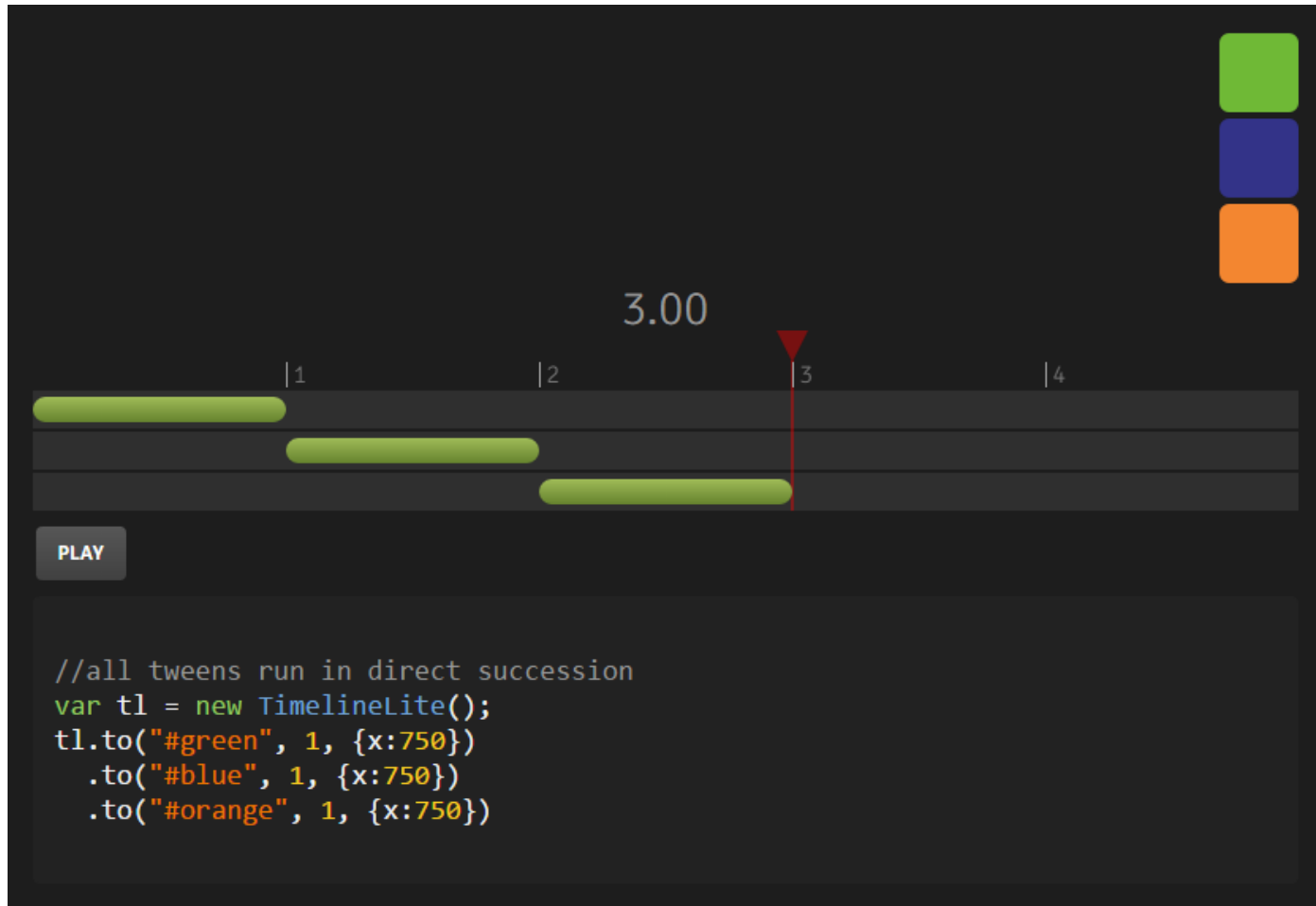
ou avec la syntaxe courte

```
var tl = new TimelineLite({paused:true});  
tl.to(element, 1, {left:100}).to(element, 1, {top:50}).to(element, 1, {opacity:0});
```

// pour controler la timeline :

```
tl.play(); tl.pause(); tl.resume(); tl.seek(1.5); tl.reverse(); tl.timeScale(.2)
```


5. TimelineLight : base



The screenshot displays the TimelineLight interface. At the top right, there are three colored squares: green, blue, and orange. The main timeline area shows a horizontal axis with four markers labeled |1, |2, |3, and |4. A red vertical line with a downward-pointing triangle is positioned at marker |3, with the value 3.00 displayed above it. Three green horizontal bars represent tweens: the first starts at |1 and ends at |2, the second starts at |2 and ends at |3, and the third starts at |3 and ends at |4. Below the timeline is a 'PLAY' button. At the bottom, a code editor shows the following JavaScript code:

```
//all tweens run in direct succession
var tl = new TimelineLite();
tl.to("#green", 1, {x:750})
  .to("#blue", 1, {x:750})
  .to("#orange", 1, {x:750})
```

5. TimelineLight : offset

```
var tl = new TimelineLite({paused:true});  
tl.add( TweenLite.to(element, 1, {left:200, top:100}) );  
  
// Ajout de tween avec offset de 0.5 secondes  
tl.add( TweenLite.to(element, 0.5, {opacity:0}, "+=0.5");  
  
tl.play();
```

5. TimelineLight : offset +



The diagram illustrates a timeline with three tweens. A vertical red line on the left marks the start. The timeline is divided into four segments by markers labeled |1, |2, |3, and |4. The first segment (from start to |1) contains a green bar. The second segment (from |1 to |2) is empty. The third segment (from |2 to |3) contains a blue bar. The fourth segment (from |3 to |4) contains an orange bar. A '1.0' label is positioned above the timeline, indicating the duration of the gap between the first and second tweens. A 'PLAY' button is located below the timeline.

```
//there is a 1 second gap between tweens
var tl = new TimelineLite();
tl.to("#green", 1, {x:750})
  // insert 1 second after end of timeline
  .to("#blue", 1, {x:750}, "+=1")
  // insert 1 second after end of timeline
  .to("#orange", 1, {x:750}, "+=1")
```

5. TimelineLight : offset -



1.0

1 2 3 4

PLAY

```
//tweens overlap
var tl = new TimelineLite();
tl.to("#green", 2, {x:750})
    // insert 1 second before end of timeline
    .to("#blue", 2, {x:750}, "-=1")
    // insert 1 second before end of timeline
    .to("#orange", 2, {x:750}, "-=1");
```

5. TimelineLight : offset absolu



The image shows a TimelineLight interface. On the left, there are three colored squares: green, blue, and orange. The timeline itself is a horizontal bar with a red playhead at the start. It is divided into four segments labeled 1, 2, 3, and 4. A green bar represents a tween that starts at the beginning and ends at the 1.0 mark. Below it, a blue bar represents a tween that starts at the 1.0 mark and ends at the 2.0 mark. Below that, an orange bar represents a tween that starts at the 2.0 mark and ends at the 3.0 mark. A 'PLAY' button is located below the timeline.

```
//tweens are inserted at absolute position
var tl = new TimelineLite();
tl.to("#green", 4, {x:750})
    //insert tween 1 second into timeline
    .to("#blue", 2, {x:750}, 1)
    //insert tween 1 second into timeline
    .to("#orange", 2, {x:750}, 1);
```

5. TimelineLight : offset avec label



The image shows a TimelineLight interface. On the left, there are three colored squares: green, blue, and orange. The timeline itself is a horizontal bar with a red playhead at the start. It has four markers labeled 1, 2, 3, and 4. A label 'blueGreenSpin' is positioned above marker 2. Three green bars represent tweens: the first starts at marker 1 and ends at marker 2; the second starts at marker 2 and ends at marker 3; the third starts at marker 3 and ends at marker 4. A 'PLAY' button is located below the timeline. Below the timeline is a code editor with the following code:

```
//tweens are inserted at and relative to a label's position
var tl = new TimelineLite();
tl.to("#green", 1, {x:750})
//add blueGreenSpin label 1 second after end of timeline
.add("blueGreenSpin", "+=1")
//add tween at blueGreenSpin label
.to("#blue", 2, {x:750, rotation:360}, "blueGreenSpin")
//insert tween 0.5 seconds after blueGreenSpin label
.to("#orange", 2, {x:750, rotation:360}, "blueGreenSpin+=0.5");
```

5. TimelineLight : play depuis label

```
// Ajoute label "spin" label a 3-secondes dans la timeline
```

```
tl.addLabel("spin", 3);
```

```
// Ajoute une rotation de l'élément a l'instant "spin" (label)
```

```
tl.add( TweenLite.to(element, 2, {rotation:"+=360"}), "spin");
```

```
// Demarrer l'animation au label "spin"
```

```
tl.play("spin");
```

6. TimelineMax : répéter + inclusion

// Timeline principale répétée "aller-retour"

```
var tl = new TimelineMax({repeat:2, yoyo:true, onComplete:myFunction});
```

```
tl.add( TweenLite.to(element, 0.5, {left:100, top:100}) );
```

```
tl.to(element, 3, {left:300}, "+=0.5");
```

// Timeline secondaire répétée a l'infini en 'yoyo' avec 0.5s entre répétitions

```
var tl2 = new TimelineMax({repeat:-1, repeatDelay:0.5, yoyo:true});
```

```
tl2.to(elementInclus, 1, {rotation:360}));
```

// Inclusion de la tl2 dans la tl1

```
tl.add(tl2);
```


6. TimelineMax

Exemples :

<http://codepen.io/GreenSock/pen/bpezc>

// divs séquencées

<http://codepen.io/MAW/pen/XmozON>

// slides

<http://codepen.io/robbue/pen/weyrb>

// transition de page

<http://codepen.io/johnheiner/pen/EaGEWv>

// transition

7. DrawSVGplugin

```
// Dessine tous les éléments ".draw-me" en les démarrant à intervalles de 0.2 s  
TweenMax.staggerFrom(".draw-me", 2, {drawSVG:0}, 0.2);
```

```
// Animation part du centre et remplit jusqu'aux extrémités en 1s  
TweenLite.fromTo("#path", 1, {drawSVG:"50% 50%"}, {drawSVG:"0% 100%"});
```

```
// Animation de segment de 5% qui suit la ligne jusqu'à la fin en 1s  
TweenLite.fromTo("#path", 1, {drawSVG:"0 5%"}, {drawSVG:"95% 100%"});
```

7. DrawSVGplugin

Exemples:

<http://codepen.io/MAW/pen/gbMWwz>

// icones

<http://codepen.io/chrisgannon/pen/emVgMg>

// ampoule

<http://codepen.io/MAW/pen/PwBRxJ>

// schéma lignes

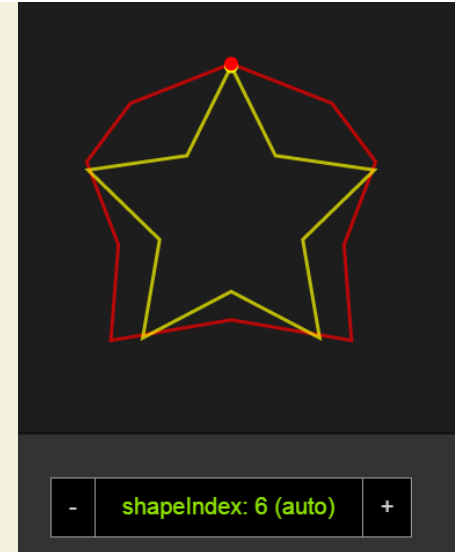
8. MorphSVGplugin

```
// Exemple pour morpher du path1 au path2  
TweenLite.to("#path1", 1, {morphSVG:"#fpath2"});
```

```
// findShapeIndex() permet d'optimiser les points  
findShapeIndex("#forme1", "#forme2");
```

```
// Une fois le shapeIndex choisi, on l'intègre dans le tween  
TweenLite.to("#square", 1, {morphSVG:{shape:"#star", shapeIndex:3}});
```

<https://s3-us-west-2.amazonaws.com/s.cdpn.io/16327/findShapeIndex.js>



8. MorphSVGplugin

// Convertit les formes en path pour pouvoir les morpher

```
MorphSVGPlugin.convertToPath("circle, rect, ellipse, line, polygon, polyline");
```

// Morphe d'une forme a l'autre chaque seconde

```
tl.to(circle, 1, {morphSVG:"#hippo"}, "+=1")
```

```
.to(circle, 1, {morphSVG:"#star"}, "+=1")
```

```
.to(circle, 1, {morphSVG:"#elephant"}, "+=1")
```

```
.to(circle, 1, {morphSVG:circle}, "+=1");
```

8. MorphSVGplugin

Exemples :

<http://codepen.io/GreenSock/pen/rOjeRq>

// formes

<http://codepen.io/chrisgannon/pen/WQGORL>

// icone morph

<http://codepen.io/rachsmith/pen/GpEJeL>

// bouton bière

<http://codepen.io/chrisgannon/pen/doNaNR>

// logo + audio

<http://codepen.io/waterfallmedia/pen/ZbOjRO>

// vagues

<http://codepen.io/chrisgannon/pen/xGOpNP>

// câbles

<http://codepen.io/jchoura/pen/yYeQdz>

// menu

9. SplitText

// Element DOM:

```
var yourElement = document.getElementById("yourID");  
var split = new SplitText(yourElement);
```

// Utilise jQuery par défaut s'il est chargé:

```
var split = new SplitText("#yourID");
```

// Utilise selecteur jQuery:

```
var split = new SplitText( $(".yourClass") );
```

// Utilise un array d'éléments du DOM:

```
var split = new SplitText([element1, element2, element3]);
```

9. SplitText

// Crée une instance SplitText avec l'élément "myID" :

```
var split = new SplitText("#myID", {type:"chars,words,lines", position:"absolute"});
```

// Anime chaque lettre a sa position depuis 100px en dessous avec fade in:

```
TweenMax.staggerFrom(split.chars, 1, {y:-100, autoAlpha:0}, 0.05);
```

// Anime chaque mot avec effet

```
TweenMax.staggerFrom(split.words, 1, {x:200, autoAlpha:0,  
ease:Elastic.easeOut}, 0.1);
```


9. SplitText

Exemples :

<http://codepen.io/GreenSock/pen/gFHza>

<http://codepen.io/GreenSock/pen/FqrEv>

10. Draggable

// Définit un élément déplaçable

```
Draggable.create("#yourID");
```

// Définit un élément déplaçable sur l'axe Y dans '#container' seulement

```
Draggable.create("#yourID", { type:"y", bounds: "#container"});
```

// Définit un élément déplaçable en rotation, avec incréments de 45deg.

```
Draggable.create("#yourID", { type:"rotation", snap: 45});
```

10. Draggable

Exemples :

<http://codepen.io/GreenSock/pen/gnoDc>

<http://codepen.io/MAW/pen/wKWVmG>

<http://codepen.io/MAW/pen/bdrBNg>

// knob + scroll

// draggable + throwprops

// bouton lacher retour

Références

<http://greensock.com/>

Ainsi que de nombreux exemples sur codepen.io