Se familiariser avec scriptaculous Partie I

par Didier Mouronval (Site perso)

Date de publication : 21 octobre 2008

Dernière mise à jour :

Cet article a pour but de vous familiariser avec le framework scriptaculous.

Avant-propos

Avec l'avènement de ce que l'on appelle (avec du reste beaucoup de mal à le définir !) le Web 2.0, les développeurs Web se tournent de plus en plus vers les frameworks JavaScript pour concevoir et dynamiser leurs pages.

Parmi ces bibliothèques, **Prototype** est particulièrement appréciée. Cependant, Prototype est essentiellement axé vers la simplification de la syntaxe, la navigation dans le DOM, les appels AJAX et surtout la portabilité du code sur différents navigateurs.

Afin d'ajouter du dynamisme et de l'ergonomie aux pages, **script.aculo.us** a été créée, se basant sur la puissance de Prototype et en y ajoutant de nombreuses fonctionnalités que nous détaillerons par la suite.

Tous les exemples de cet articles sont valides xhtml 1.0 strict et css 2.1 et ont été testés sous : Internet Explorer 6 et 7, Firefox 2 et 3, Safari 3, Opera 9.6, Netscape 9 et Chrome.

I - Intégrer script.aculo.us à sa page

La première des choses à faire est bien entendu de télécharger l'archive du framework **disponible ici** puis de la décompresser !

I-Préambule - Pourquoi utiliser scriptaculous ?

Tout d'abord, avant d'essayer d'aller plus loin, la première question à se poser est la suivante : "Ai-je réellement besoin de cette librairie pour mes pages ?"
En effet, différentes motivations peuvent inciter à utiliser un framework, des bonnes et des mauvaises. Or ces bibliothèques imposent de télécharger (parfois à chaque page en fonction des réglages du serveur et/ou de la configuration du cache du navigateur) jusqu'à 300ko de fichiers, ce qui peut ralentir la navigation sur votre site.

Parmi les bonnes raisons :

- Utiliser beaucoup de fonctionnalités complexes du framework.
- Mutualiser des fonctions génériques sur plusieurs pages.
- Etre à l'aise avec JavaScript, mais vouloir optimiser son code dans un aspect "cross-browser".
- Savoir faire un ratio facilité de coder / difficulté de reproduire les fonctionnalités.

Parmi les mauvaises raisons :

- Apprécier un raccourci de code ou une fonctionnalité !
- Méconnaissance de JavaScript.
- Croire que le framework vous permettra de surmonter les difficultés à coder.

A titre d'exemple :

```
Event.observe(window, 'load', function(){
    $('une_image').setOpacity(0.5);
    $('une_image').observe('mouseover',function(){
        new Effect.Opacity('une_image', {from: 0.5, to: 1});
    });
    $('une_image').observe('mouseout', function(){
        new Effect.Opacity('une_image', {from: 1, to: 0.5});
    });
});
```

Voir l'exemple

aura le même comportement que

```
var IsIE = !!document.all;
function fadePic(elt, debut, fin){
    if(debut == fin){ return false }
    IsIE ? elt.filters[0].opacity=debut : elt.style.opacity=debut/100;
    debut > fin ? debut -= 2 : debut += 2;
    setTimeout(function(){ fadePic(elt, debut, fin); }, 10);
}
window.onload=function(){
    var elt = document.getElementById('une_image');
    if(IsIE){
        elt.style.filter = 'alpha(opacity = 50)';
        elt.filters[0].opacity = 50;
    }
    else{ elt.style.opacity = '0.5'; }
    elt.onmouseover = function(){ fadePic(elt, 50, 100) }
    elt.onmouseout = function(){ fadePic(elt, 100, 50) }
}
```

Voir l'exemple

Nous voyons bien ici qu'intégrer une librairie comme scriptaculous (presque 250ko et 8000 lignes de code) n'est pas forcément optimisé pour n'utiliser que quelques fonctions réalisables par soi-même pour peu que l'on maîtrise un minimum JavaScript et la compatibilité des navigateurs !

En revanche, dans de nombreux cas, les fonctionnalités vous aideront à concevoir et coder vos pages avec énormément d'efficacité.

I-A - Présentation de l'archive

L'archive fournie dans le lien est supposée contenir les fichiers et dossiers suivants :

Contenu de l'archive

- 3 fichiers (CHANGELOG MIT-LICENCE et README)
- Répertoire 'lib' ;
- Répertoire 'src' ;
- Répertoire 'test'.

Les fichiers, bien que sans extension, sont des fichiers texte dont je vous invite à prendre connaissance.
Le répertoire 'test' contient des pages html permettant de faire tourner des tests unitaires ou fonctionnels.
Le répertoire 'lib' contient la bibliothèque Prototype, elle est indispensable pour pouvoir utiliser scriptaculous, cependant, la version fournie n'est pas à jour, la dernière version de Prototype (1.6.0.3 sortie le 29/09/08) est **disponible ici**. Pour cet article, c'est la version 1.6.0.2 qui a été prise comme référence.
Le répertoire 'src' contient les fichiers source de scriptaculous, organisés par modules.

I-B - Les différents modules

Le fonctionnement de scriptaculous se fait par un système de modules. Chaque module regroupant un ensemble de fonctions d'un même type. L'intérêt de cette structure étant de pouvoir ne charger que les modules dont on a réellement besoin comme nous le verrons au chapitre suivant.
Les différents modules sont les suivants :

Modules scriptaculous

- effects.js : gère les fonctions relatives aux effets ;
- controls.js : gère essentiellement les fonctions d'auto complétion et de champs éditables ;
- dragdrop.js : gère les fonctions de glisser déplacer ;
- slider.js : gère les fonctions de barres de mesure ;
- builder.js : gère la création d'éléments du DOM ;
- sound.js : gère l'intégration de sons.

Le fichier unittest.js ne faisant pas à proprement parler partie du framework, il ne sera pas abordé dans cet article.

Le fichier de base du framework, 'scriptaculous.js' ne contient aucune fonction importante. Il n'est en fait qu'un loader utilisé pour vérifier la compatibilité avec la version de Prototype chargée et charger les modules désirés (par défaut, tous les modules). L'originalité de la librairie étant de pouvoir indiquer en paramètre du fichier les modules à installer.

I-C - Intégrer le script à la page

Il ne reste donc plus qu'à insérer les scripts dans la page html.
Pour cela, nous allons insérer les balises script dans le head de notre page comme ceci :

Contrairement à d'autres frameworks, les différents modules sont indépendants entre eux, ainsi, aucune fonction de scriptaculous ne nécessite la présence d'un module autre que celui qui la contient.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>scripaculous</title>
    <script type="text/javascript" src="js/prototype.js"></script>
    <script type="text/javascript" src="js/scriptaculous.js"></script>
</head>
```

Nous avons évoqué la possibilité de ne charger que certains modules, dans ce cas, si vous souhaitez par exemple ne charger que les fichiers effects et dragdrop, il suffit de remplacer la seconde ligne par :

```
<script type="text/javascript" src="js/scriptaculous.js?load=effects,dragdrop"</script>
```

et le script se charge du reste !

Il est bien évidemment possible de charger directement les scripts, mais d'une part, pourquoi se priver d'une fonctionnalité créée spécialement pour cela et d'autre part, encore une fois, la vérification de la compatibilité Prototype / scriptaculous est bien pensée pour les "étourdis" !

Il existe une archive très pratique, nommée protopack, qui contient beaucoup de choses utiles :
- Les dernières versions des fichiers ;
- Des versions avec différents niveaux de compression des deux framework (appelées protoculous) ;
- Différentes versions de protoculous intégrant uniquement le fichier effects ;
- Une version faiblement compressée (shrinkvar) de protoculous, facilement éditable pour se créer sa librairie personnalisée.
*Cette archive est **disponible ici***

II - La syntaxe sommaire des fonctions scriptaculous

Maintenant que scriptaculous est chargé dans votre page, reste à utiliser ses fonctions !
Nous partons ici du principe que vous connaissez les bases de Prototype, si ce n'est pas le cas, je vous invite à prendre connaissance de **cet article**
Nous allons donc détailler les différentes fonctions par module.

II-A - Les effets visuels (fichier effects.js)

Les effets visuels de scriptaculous sont répartis en deux catégories, les effets noyau et les effets combinés.
Les effets combinés sont en fait une combinaison d'effets de base préformatés.
Bien comprendre cette notion vous sera très utile lorsque vous voudrez créer vos propres effets personnalisés.

II-A-1 - Les effets noyau

Les effets noyau correspondent à des effets de base, certains utiles, d'autres moins, mais qui seront indispensables aux effets combinés qui constituent une vraie richesse pour un langage comme JavaScript aussi peu porté sur le graphisme !

Il existe 7 effets noyau, Opacity, Scale, Morph, Move, Highlight, ScrollTo et Tween que nous allons détailler ici.
Il est à noter qu'il existe d'autres effets faisant partie des "core effects" de la bibliothèque, mais j'ai préféré les détailler à part dans les effets de synchronisation car ils n'agissent pas de façon visuelle.

La syntaxe générale pour appeler un effet est la suivante :

```
new Effect.NomEffet('element', parametres, [{options}]);
```

Explicatif :

- NomEffet : le nom de l'effet. (Les choses qui vont sans dire vont toujours mieux en les disant !)
- 'element' : l'id de l'élément sur lequel va s'appliquer l'effet. Vous noterez qu'il n'est pas nécessaire d'utiliser la syntaxe d'élément étendu de Prototype ($('element')), scriptaculous se charge de cela lui-même !
- parametres : les paramètres requis pour l'effet.
- {option} : au format JSON (JavaScript Object Notation), les paramètres facultatifs de l'effet.

II-A-1-a - Les options communes

Attention, il s'agit bien des options et non des paramètres !

Option	Description	Défaut
duration	La durée en secondes de l'effet	1.0
fps	Le nombres de rafraichissements par secondes	25
transition	L'option transition est en fait un effet à part entière que nous aborderont plus loin	sinoidal
from	Valeur initiale de la transition, entre 0.0 et 1.0	0.0
to	Valeur finale de la transition, entre 0.0 et 1.0	1.0
sync	Booléen utilisé par l'effet Parallel. Synchronise ou non les effets	true
queue	Position dans une file d'attente. Là encore, un effet à part entière que nous verrons plus loin.	
delay	Le délai en secondes avant le lancement de l'effet.	0.0

Il existe de même des événements spécifiques et des variables accessibles pendant le déroulement de l'effet :

Evénement	Description
beforeStart	Appelé avant de commencer l'effet.
beforeUpdate	Appelé avant la mise à jour du rafraichissement.
afterUpdate	Appelé après la mise à jour du rafraichissement.
afterFinish	Appelé quand l'effet est terminé.

Variable	Dexcription
effect.element	L'élément sur lequel est appliqué l'effet.
effect.options	Les options de l'effet.
effect.currentFrame	Le numéro de l'image actuelle.
effect.startOn, effect.finishOn	Le moment (en ms) de début et de fin de l'effet.
effect.effects[]	Pour les effets Parallel, le tableau des effets invoqués.
effect.cancel()	Stopper l'effet.

II-A-1-b - Présentation et options spécifiques des effets noyau

L'effet Opacity

Cet effet fait varier l'opacité d'un élément du paramètre from au paramètre to.

Syntaxe

```
new Effect.Opacity('id_element'[, options]);
new Effect.Opacity(element[, options]);
```

Avec :

- 'id_element' ou element faisant référence à l'élément sur lequel on souhaite appliquer l'effet.
- Paramètres obligatoires : from et to.

Les objets Prototype (créés avec la syntaxe $('id_element')) sont appelés des objets étendus dans la mesure où Prototype leur donne des propriétés internes.
Il est intéressant de noter que vous n'avez pas besoin de mettre en premier paramètre un objet Prototype étendu car cette action est effectuée automatiquement par scriptaculous pour tous les effets.

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Test effet Opacity</title>
    <script type="text/javascript" src="js/prototype.js"></script>
    <script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>
```

Code de l'exemple
```
<body>

  <div id="conteneur" style="margin: 50px 0 0 50px;width: 100px;height:100px; background-color:gray;">
    </div>
    <div>
        <input type="button" value="Afficher (1 seconde)"
        onclick="new Effect.Opacity('conteneur', { from: 0, to: 1 });" />
        <input type="button" value="Effacer (2 secondes)"
        onclick="new Effect.Opacity('conteneur', { from: 1, to: 0, duration: 2 });" />
        <br /><input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Attention ! Dans les versions antérieures de scriptaculous, l'élément auquel s'applique l'effet devait avoir un layout pour pouvoir fonctionner sous IE. Même si ce n'est plus le cas, pensez-y si vous utilisez une version ancienne du script.

Source de effects.js
```
if (Prototype.Browser.IE && (!this.element.currentStyle.hasLayout))
      this.element.setStyle({zoom: 1});
```

Affecter un 'layout' à un élément sous IE.

- Attribuer des propriétés CSS "height" ou "width" (ne fonctionne pas sur les éléments de type inline si IE n'est pas en *quircksmode*).
- Mettre la propriété "display" à *inline-block*.
- Mettre la propriété "position" à *absolute*.
- Mettre la propriété "writingMode" à *tb-rl*.
- Mettre la propriété "contentEditable" à *true*.

L'effet Scale

Adapte progressivement la taille d'un élément d'un pourcentage donné.

Syntaxe
```
new Effect.Scale('id_element', pourcentage, [options]);
new Effect.Scale(element, pourcentage, [options]);
```

Avec :

- 'id_element' ou element faisant référence à l'élément sur lequel on souhaite appliquer l'effet.
- pourcentage : valeur numérique correspondant à l'agrandissement / rétrécissement de l'élément cible.

Cet effet possède des options spécifiques :

Options spécifiques	Description	Défaut
scaleX	L'effet doit être appliqué horizontalement	true
scaleY	L'effet doit être appliqué verticalement	true
scaleContent	L'effet doit être aussi appliqué au contenu	true
scaleFromCenter	L'effet doit être appliqué depuis le centre de l'élément	false
scaleMode	'box' n'applique l'effet qu'à la partie visible de l'élément, 'contents' applique l'effet à	'box'

	l'ensemble de l'élément, y compris les parties cachées, ou un hash (format JSON) précisant la taille de départ (exemple : scaleMode: {originalHeight: y, originalWidth: x } (originalHeight et originalWidth sont des valeurs numériques)	
scaleFrom	Pourcentage de début de l'effet	100.0

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Test effet Scale</title>
    <script type="text/javascript" src="js/prototype.js"></script>
    <script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div style="margin: 50px 10px 0 50px;"><span id="conteneur">Div à agrandir / réduire</span>
    </div>
    <div>
        <input type="button" value="Agrandir (x2)" onclick="new Effect.Scale('conteneur', 200);" />
        <input type="button" value="Diminuer (x0.75)" onclick="new Effect.Scale('conteneur', 75);" />
        <br /><input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

L'effet Morph

Change les propriétés CSS d'un élément.

Syntaxe chaîne

```
new Effect.Morph('id_element',{style: 'propriete-css1: valeur1; propriete-css: valeur2;'[, options]});
```

Syntaxe hash

```
new Effect.Morph('id_element',{style: {proprieteJavascript1:'valeur1',
 proprieteJavascript2: 'valeur2'[, options]});
```

Avec :

- 'id_element' ou element faisant référence à l'élément sur lequel on souhaite appliquer l'effet.
- style a pour valeur une chaîne construite à la manière de l'attribut HTML "style" ou un hash à la manière de l'objet style de JavaScript. 'valeur' correspondant à la valeur finale souhaitée.

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
```

Code de l'exemple

```
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Test effet Morph</title>
    <style type="text/css">
<!--
#conteneur{
 margin: 50px 0 10px 50px;
 width: 100px;
 height: 100px;
 color:#000;
 background-color:#FFF;
 border: 1px solid silver;
}
-->
    </style>
    <script type="text/javascript" src="js/prototype.js"></script>
    <script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div id="conteneur">Une div</div>
    <div>
        <input type="button" value="Blanc -> Noir"
        onclick="new Effect.Morph('conteneur', {style: 'background-color:#000; color:#FFF; width: 300px'});" />
        // Notation chaîne / CSS
        <input type="button" value="Noir -> Blanc"
        onclick="new Effect.Morph('conteneur', {style: {backgroundColor: '#FFF', color: '#000', width: '100px'},
        // Notation hash / JavaScript
        <br /><input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Cet effet ne s'applique qu'aux propriétés CSS dont la valeur est de type dimension ou couleur.
Cet effet est utilisable quelle que soit la façon dont la règle CSS a été définie (CSS externe, interne ou attribut style).
Seul Internet Explorer n'accepte pas les couleurs définies par leur nom. Il est donc préférable d'utiliser la notation hexadécimale pour la compatibilité !

L'effet Move

Cet effet déplace un élément dans la page.

Syntaxe mode absolute

```
new Effect.Move('id_element', { x: 0, y: 0, mode: 'absolute' });
```

Syntaxe mode relative

```
new Effect.Move('id_element', { x: -20, y: 30, mode: 'relative' });
```

Avec

- 'id_element' ou objet element faisant référence à l'élément sur lequel on souhaite appliquer l'effet.
- x et y : la position à laquelle finira l'effet (mode absolute) ou le nombre de pixels dont doit être déplacé l'élément (mode relative).
- mode : 'absolute' ou 'relative' (valeur par défaut), la position finale, calculée par rapport à la page ou à la position initiale de l'élément.

Voir un exemple

Code de l'exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Move</title>
<style type="text/css">
<!--
#conteneur{
    margin: 50px 0 10px 50px;
    width: 200px;
    height: 100px;
    color: #000;
    background-color: silver;
    cursor: pointer;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
<script type="text/javascript">
<!--// [CDATA[
var posX = 200;
var posY = 100;
// ]] -->
</script>
</head>

<body>

  <div id="conteneur" onclick="new Effect.Move('conteneur', { x: 0, y: 0, mode: 'absolute', duration: 0.5 });">
        Div à déplacer.<br />Cliquer dessus pour revenir à la position initiale
    </div>
    <div>
        <label for="posX">Déplacement en x : </label>
        <select id="posX" onchange="posX=parseInt(this.value,10)">
            <option value="-100">-100</option>
            <option value="-50">-50</option>
            <option value="0">0</option>
            <option value="50">50</option>
            <option value="100">100</option>
            <option value="200" selected="selected">200</option>
        </select>
        <label for="posY">Déplacement en y : </label>
        <select id="posY" onchange="posY=parseInt(this.value,10)">
            <option value="-100">-100</option>
            <option value="-50">-50</option>
            <option value="0">0</option>
            <option value="50">50</option>
            <option value="100" selected="selected">100</option>
            <option value="200">200</option>
        </select><br />

  <input type="button" value="Déplacer" onclick="new Effect.Move('conteneur', { x: posX, y: posY });" />

  <input type="button" value="Revenir" onclick="new Effect.Move('conteneur', { x: 0, y: 0, mode: 'absolute', dura
        <br /><input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Dans des versions précédentes de scriptaculous, l'effet de déplacement (Move ou MoveBy selon les versions) nécessitait de positionner l'élément. On trouve toujours cette mention dans certaines documentations, mais ce n'est plus le cas. Pour preuve, dans l'exemple donné, la div 'conteneur' n'est pas positionnée, mais l'effet fonctionne tout de même sous IE !

L'effet Highlight

Cet effet permet de modifier la couleur de fond d'un élément.
Il est particulièrement utile pour attirer l'attention sur un changement de contenu réalisé en AJAX.

Syntaxe

```
new Effect.Highlight('id_element'[, options]);
```

Options spécifiques	Description	Défaut
startcolor	La couleur de fond sur laquelle débutera l'effet	'#FFFF99'
endcolor	La couleur de fond sur laquelle finira l'effet	Propriété *background-color* ou '#FFFFFF'
restorecolor	La couleur de fond affectée à l'élément après la fin de l'effet	Indéfinie

Attention !

- Si restorecolor n'est pas définie, l'effet tentera de lui donner la valeur de la propriété *background-color*. Ce qui n'est garanti sur tous les navigateurs que si celle-ci est affectée en utilisant la syntaxe rgb (ex : rgb(255, 255, 255)).
- Pour les options startcolor et endcolor, préférer les valeurs hexadécimales complètes (#FFFFFF) plutôt que courtes (#FFF).
- Le *new* est obligatoire pour cet effet, sinon il ne fonctionnera pas !

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Highlight</title>
<style type="text/css">
<!--
div{
    margin: 40px;
    float: left;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
 <div>
  <h3 id="conteneur1">Valeurs de l'effet par défaut</h3>
  <input type="button" value="Tester" onclick="new Effect.Highlight('conteneur1');" />
 </div>
 <div>
  <h3 id="conteneur2">startcolor : '#FF0000'</h3>
  <input type="button" value="Tester" onclick="new Effect.Highlight('conteneur2', {startcolor: '#FF0000'});" />
 </div>
 <div>
  <h3 id="conteneur3">endcolor : '#C0C0C0'</h3>
  <input type="button" value="Tester" onclick="new Effect.Highlight('conteneur3', {startcolor: '#FFFFFF', endcolo
 </div>
 <div>
  <h3 id="conteneur4">restorecolor : 'rgb(192, 192, 192)'</h3>
  <input type="button" value="Tester" onclick="new Effect.Highlight('conteneur4', {restorecolor: 'rgb(192, 192, 1
```

Code de l'exemple
```
   <input type="button" value="Restaurer" onclick="$('conteneur4').style.backgroundColor=''" />
  </div>
  <div style="clear: both">
   <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
  </div>
 </body>
 </html>
```

L'effet ScrollTo

Cet effet permet d'effectuer un d'atteindre un élément.

Syntaxe
```
new Effect.ScrollTo('id_element'[, options]);
```

'id_element' est l'élément vers lequel on veut aller.

Voir un exemple

Code de l'exemple
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet ScrollTo</title>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
<script>
<!--
function compte(){
    for(var i=0;i<200;i++){
        $('conteneur').innerHTML+='<br />'+i;
    }
}
-->
</script>
</head>

<body onload="compte()">
    <div id="conteneur">
        <span id="haut">Haut de la page</span>
        <input type="button" value="Descendre" onclick="new Effect.ScrollTo('bas');" />
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
    <div>

  <input type="button" value="Remonter rapidement" onclick="new Effect.ScrollTo('haut', {duration: 0.5});" />
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    <div id="bas">Tout en bas !</div>
</body>
</html>
```

L'effet Tween

Cet effet permet de faire varier la valeur numérique d'une propriété ou du paramètre d'une fonction.

Syntaxe
```
new Effect.Tween(element, debut, fin[, options], Nom_de_propriete);
new Effect.Tween('id_element', debut, fin[, options], fonction);
```

Avec :

- element ou 'id_element' : l'élément sur lequel va s'appliquer l'effet.
- debut : la valeur initiale de la propriété ou du compteur.
- fin : la valeur finale de la propriété ou du compteur.
- options : les options habituelles communes aux effets.
- Nom_de_propriete : la propriété de l'élément que l'on souhaite faire varier.
- fonction : une fonction appelée à chaque changement de valeur.

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Tween</title>
<style type="text/css">
<!--
#conteneur2{
    background-color : blue;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effet dans la barre de titre.</h3>

  <input type="button" value="Tester" onclick="new Effect.Tween(document, 0, 100, {duration: 5.0}, 'title')" />
    </div>
    <div class="exemple">
        <h3>Apparition d'une image.</h3>
        <div>Propriétés width et height de l'objet image.<br />
            <img id="conteneur1" width="0" height="0" src="./imgs/exemple.jpg" alt="Fond" />
        </div>

  <input type="button" value="Apparaitre" onclick="new Effect.Tween('conteneur1', 0, 400, 'width');new Effect.Twe

  <input type="button" value="Disparaitre" onclick="new Effect.Tween('conteneur1', 400, 0, 'width');new Effect.Tw
    </div>
    <div class="exemple">
        <h3>Apparition d'une div.</h3>
        <div id="conteneur2"></div><br />

  <input type="button" value="Apparaitre" onclick="new Effect.Tween('conteneur2', 0, 300, function(p){this.style.

  <input type="button" value="Disparaitre" onclick="new Effect.Tween('conteneur2', 300, 0, function(p){this.style
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

II-A-2 - Les effets de synchronisation

Il n'existe pas de référence à des effets de synchronisation dans la librairie scriptaculous ni dans la documentation officielle. Cependant, j'ai choisi de réunir dans cette rubrique une série d'effets provenant soit des effets noyaux soit de la classe *Effect.Base*.
Il s'agit d'une collection d'effets dont le rôle est de coordonner les autres effets entre eux.
Les effets que nous allons voir dans cette rubrique sont les suivants : Transitions, Parallel, multiple, Queue / Queues / ScopedQueue (que j'ai réunis car ils participent à la même fonctionnalité) et Methods.

L'effet Transitions

L'effet Transitions n'est en réalité qu'une collection de formules mathématiques qui vont permettre de modifier le comportement d'un effet dans le temps.
En interne, un effet n'est pas une simple incrémentation linéaire de valeurs. Dans les exemples proposés précédemment, la transition par défaut (sinoidal) a été appliquée et vous pouvez constater que l'effet accélère au début et ralenti à la fin.

Il existe neuf types de transitions :

Transition	Description	Modélisation mathématique
linear	Incrémentation linéaire des valeurs de l'effet.	i
sinoidal	Accélération progressive de l'animation.	-cos(i∏/2)+0.5
reverse	Effet inversé.	1-i
flicker	Effet de tremblement accentué sur la fin de l'animation.	max(((-cos(i∏)/4)+0.75)+rand()/4, 1)
wobble	Effet de secousses.	-cos(10i∏)/2+0.5
pulse	L'effet est accéléré 5 fois et répété 5 fois.	(i*pulses*2)-floor((i*pulses*2)) ou 1-(i*pulses*2-floor(i*pulses*2)) pulses est le nombre de pulsations (5 par défaut)
spring	Effet de stabilisation sur la position finale.	1-cos(4.5i*∏)*e(-6i)
none	L'effet reste à sa position initiale.	0
full	L'effet va à sa position finale immédiatement.	1

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Transitions</title><style type="text/css">
<!--
#conteneur1{
   background-color : blue;
   width: 150px;
   height: 100px;
```

Code de l'exemple

```
    margin: 20px;
}
input{
    margin: 10px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <h2>Test de l'effet Scale</h2>
    <h3 style="margin: 10px 0 10px 50px;"><span id="conteneur">Elément à agrandir / réduire</span></h3>
    <div>

  <input type="button" value="Agrandir" onclick="new Effect.Scale('conteneur', 200, {transition: Effect.Transitio

  <input type="button" value="Diminuer" onclick="new Effect.Scale('conteneur', 50, {transition: Effect.Transition
    </div>
    <h2>Test de l'effet Move</h2>
    <div id="conteneur1"></div>
    <div>

  <input type="button" value="Tester" onclick="new Effect.Move('conteneur1', {x: 200, y: 0, mode: 'relative', tra
        <input type="button" value="Retour" onclick="$('conteneur1').style.left=''" />
    </div>
    <div>
        Choisissez une transition :
        <select id="transition">
            <option value="linear">linear</option>
            <option value="sinoidal">sinoidal</option>
            <option value="reverse">reverse</option>
            <option value="flicker">flicker</option>
            <option value="wobble">wobble</option>
            <option value="pulse">pulse</option>
            <option value="spring">spring</option>
            <option value="none">none</option>
            <option value="full">full</option>
        </select>
        puis testez un effet.
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

L'effet Parallel

Cet effet est utilisé pour lancer plusieurs effets.

Syntaxe

```
new Effect.Parallel([tableau effets][, options]);
```

Avec :

- tableau d'effets : un tableau contenant les différents effets à lancer, dans leur syntaxe usuelle.

Option spécifique	Description	Défaut
sync	Valeur booléenne, permet de lancer les effets simultanément. A indiquer dans les options de chaque effet.	true

Si l'option sync est false, l'effet en question sera lancé dès son invocation, ce qui peut causer un léger décalage.

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Parallel</title><style type="text/css">
<!--
#conteneur{
    background-color : blue;
    width: 150px;
    height: 100px;
    margin: 20px;
}
input{
    margin: 10px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <h2>Test de l'effet Parallel</h2>
    <div id="conteneur"></div>
    <div>
        <input type="button" value="Tester" onclick="new Effect.Parallel([
            new Effect.Move('conteneur', {sync: true, x: 400, y: 0, mode: 'relative' }),
            new Effect.Opacity('conteneur', {sync: true, from: 1, to: 0 }),
            new Effect.Morph('conteneur', {style: {width: '0px', height: '0px'}, sync: true})
        ], {duration: 1.5});" />

  <input type="button" value="Retour" onclick="$('conteneur').setStyle({top: '', left: '', opacity: 1, width: '',
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

L'effet multiple

Cet effet permet d'appliquer un effet à une collection d'éléments.

Syntaxe

```
Effect.multiple('id_element', Effet[, options]); // Voir note ci-dessous.
Effect.multiple(['id_1', 'id_2', ...], Effet[, options]);
Effect.multiple(fonction, Effet[, options]);
```

Avec :

- 'id_element' : Si vous précisez un id d'élément, l'effet s'appliquera à l'ensemble des childNodes cet élément.
- ['id_1', 'id_2', ...] : Si vous précisez un tableau d'id, l'effet s'appliquera à l'ensemble des éléments correspondants.
- fonction : Si vous précisez une fonction (typiquement $$ de Prototype), l'effet s'appliquera à l'ensemble des éléments retournés par cette fonction.
- Effet : l'effet à appliquer aux éléments sélectionnés.

***Attention** : lorsque vous précisez juste un id, multiple récupèrera une collection d'éléments via la méthode DOM childNodes. Du fait de la différence d'interprétation du DOM par les navigateurs, cette syntaxe est particulièrement déconseillée car elle retournera quasiment systématiquement des erreurs liées à la présence de n□uds vides dans la collection.*

Une astuce pour parer à ce désagrément : dans le code source du fichier effects.js, vous pouvez repérer la fonction multiple et modifier l'appel à la méthode childNodes par childElements() de Prototype. Toutefois, avec cette astuce, vous ne récupèrerez pas les n□uds texte !

Option spécifique	Description	Défaut
speed	Le délai avant le démarrage de chaque effet.	0.1

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet multiple</title><style type="text/css">
<!--
.conteneur{
    background-color : blue;
    width: 100px;
    height: 20px;
    margin: 20px;
}
input{
    margin: 10px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <h2>Test de l'effet multiple</h2>
    <h3>Test sur l'effet Move :</h3>
    <div class="conteneur"></div>
    <div class="conteneur"></div>
    <div class="conteneur"></div>
    <div class="conteneur"></div>
    <div>
        <input type="button" value="Tester" onclick="new Effect.multiple($$('.conteneur'),
            Effect.Move, {x: 400, y: 0, mode: 'relative' });" />
        <input type="button" value="Retour" onclick="new Effect.multiple($$('.conteneur'),
            Effect.Move, {x: 0, mode: 'absolute', duration: 0.5, speed: 0.0  });" />
    </div>
    <div>
        <h3>Test sur l'effet Opacity :</h3>
        <ul id="groupe">
            <li>Premier</li>
            <li>Deuxième</li>
            <li>Troisième</li>
            <li>Quatrième</li>
        </ul>
        <input type="button" value="Tester" onclick="new Effect.multiple($('groupe').select('li'),
            Effect.Opacity, {from: 1.0, to: 0.0 });" />
        <input type="button" value="Retour" onclick="new Effect.multiple($('groupe').select('li'),
            Effect.Opacity, {from: 0.0, to: 1.0, duration: 0.5, speed: 0.0 });" />
    </div>
    <div>
```

Code de l'exemple
```
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Le paramètre queue

Le paramètre queue englobe en fait trois effets distincts : Queue, Queues et ScopedQueue.
Ces effets ne sont pas très utiles en soi, nous allons donc voir en détail leur comportement dans l'utilisation du paramètre queue. Celui-ci permet de gérer des piles d'effets de façon relativement complexe.

Syntaxe
```
queue: {position: 'position', scope: 'file', limit: nombre}
```

A placer dans le hash paramètres de l'effet.

Paramètre	Description
position	*front* : place l'effet au début de la file. *end* : place l'effet à la fin de la file. *with-last* : fera démarrer l'effet en même temps que le dernier de la file.
scope	Par défaut 'global', permet de définir différentes files d'effets gérées indépendamment.
limit	Le nombre maximal d'effets dans la file donnée. Ceux en surnombre seront ignorés.

Voir un exemple

Code de l'exemple
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Queue</title>
<style type="text/css">
<!--
#conteneur{
    background-color : blue;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
<script type="text/javascript">
<!--
function deplacement(){
    new Effect.Move('conteneur', {x: 400, mode: 'relative', queue: {position: 'front',
 scope: 'deplace'}});
    new Effect.Move('conteneur', {y: 400, mode: 'relative', duration: 2, queue: {position: 'end',
 scope: 'deplace'}});
    new Effect.Move('conteneur', {x: -400, y: -400, mode: 'relative', queue: {position: 'end',
 scope: 'deplace'}});
```

Code de l'exemple

```
    new Effect.Morph('conteneur', {style: {width: '300px', backgroundColor: '#000000'},
 queue: {position: 'front', scope: 'change'}});
    new Effect.Parallel([
        new Effect.Morph('conteneur', {style: {width: '200px'}, sync: true}),
        new Effect.Opacity('conteneur', {from: 1, to: 0.5, sync: true})],
        {duration: 2, queue: {position: 'end', scope: 'change'}});
    new Effect.Morph('conteneur', {style: {backgroundColor: '#0000FF'}, queue: {position: 'end',
 scope: 'change'}});
    new Effect.Opacity('conteneur', {from: 0.5, to: 1.0, queue: {position: 'with-last',
 scope: 'change'}});
}
-->
</script>
</head>

<body>
    <div class="exemple">
        <h3>Apparition d'une div.</h3>
        <div id="conteneur"></div>
        <input type="button" value="Animer" onclick="deplacement();" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Raccourcis

scriptaculous propose aussi un ensemble de raccourcis pour la plupart des effets.

Syntaxe

```
$('element').effet({options});
```

Avec :

- $('element') : l'élément étendu (au sens de Prototype) sur lequel appliquer l'effet.
- effet : nom de l'effet à appliquer.
- {options} : le hash des options obligatoires et facultatives à appliquer.

Effet	Exemple
visualEffect	$('element').visualEffect(Effet, {options});
morph	$('element').morph({propriete1: valeur1, propriete2: valeur2, ...}, {options});
fade	$('element').fade({options});
appear	$('element').appear({options});
grow	$('element').grow({options});
shrink	$('element').shrink({options});
fold	$('element').fold({options});
blindUp	$('element').blindUp({options});
blindDown	$('element').blindDown({options});
slideUp	$('element').slideUp({options});
slideDown	$('element').slideDown({options});
pulsate	$('element').pulsate({options});
shake	$('element').shake({options});
puff	$('element').puff({options});
squish	$('element').squish({options});
switchOff	$('element').switchOff({options});
dropOut	$('element').dropOut({options});
highlight	$('element').highlight({options});

Vous remarquerez que les noms des effets ne prennent dans ce cas pas de majuscule initiale.

Etant donnée la nature de cette syntaxe dont le but est de simplifier les appels des effets, nous nous abstiendront de donner un exemple.

II-A-3 - Les effets combinés

Les effets noyaux sont particulièrement utiles pour permettre de créer des effets plus complexes appelés effets combinés.
Il en existe un certain nombre prédéfinis, cependant, ceux-ci ne sont pas à considérer comme des finalités, mais plus comme des bases pour vous aider à créer vos propres effets en fonction de vos besoins.

Vous avez probablement remarqué que les effets noyaux possèdent des paramètres et des options complexes.
Rassurez-vous, cette complexité permet d'avoir une syntaxe particulièrement allégée pour les effets combinés qui sont souvent ceux que l'on souhaite utiliser dans nos applications.
Cependant, la connaissance des effets noyau vous permettra par la suite de créer vos propres effets combinés, éventuellement en vous inspirant des exemples proposés par scriptaculous.

Effet Appear / Fade

Les effets Appear et Fade permettent de faire apparaitre ou disparaitre un élément en modifiant son opacité et en ajustant sa propriété display.

Pour que cet effet fonctionne, la propriété CSS display ne doit pas être fixée dans une feuille de style externe (fichier .css) ou interne (balise <style>) mais si nécessaire, via l'attribut style de la balise, comme dans l'exemple donné.

Option	Description	Défaut
from	Valeur de l'opacité au début de l'effet (0.0 à 1.0)	Appear : 0.0 Fade : 1.0
to	Valeur de l'opacité à la fin de l'effet.	Appear : 1.0 Fade : 0.0

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Appear / Fade</title>
<style type="text/css">
<!--
#conteneur{
    background-color : blue;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Apparition / Disparition d'une div.</h3>
        <div id="conteneur" style="display:none"></div>
        <input type="button" value="Apparaitre" onclick="new Effect.Appear('conteneur');" />
        <input type="button" value="Disparaitre" onclick="$('conteneur').fade({duration: 2});" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Effet Puff

Effet "bulle de savon" sur un élément : agrandissement et transparence.

Syntaxe

```
new Effect.Puff('element'[, options]);
$('element').puff({options});
```

Cet effet fonctionne pour les éléments de type block sauf les tables !

Comme tous les effets gérant la transparence, l'effet Puff gère les options *from* et *to*.

Voir un exemple

Code de l'exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Puff</title>
<style type="text/css">
<!--
#conteneur{
    background-color : blue;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effet Puff.</h3>
        <div id="conteneur"></div>

  <input type="button" value="Exemple 1" onclick="new Effect.Puff('conteneur', {duration: 2, afterFinish: functio

  <input type="button" value="Exemple 2" onclick="$('conteneur').puff({transition: Effect.Transitions.spring, aft
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Effet DropOut

L'élément tombe en disparaissant.

Syntaxe

```
new Effect.DropOut('element'[, options]);
$('element').dropOut({options});
```

Cet effet fonctionne pour les éléments de type block sauf les tables !

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Dropout</title>
<style type="text/css">
<!--
#conteneur{
   background-color : blue;
   width : 200px;
   height : 150px;
}
```

Code de l'exemple

```
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effet DropOut.</h3>
        <div id="conteneur"></div>
        <input type="button" value="DropOut" onclick="new Effect.DropOut('conteneur');" />
        <input type="button" value="Rétablir" onclick="$('conteneur').style.display='';" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Effet Shake

Effet de secousses latérales.

Option	Description	Défaut
distance	L'amplitude du mouvement, en pixels.	20

Syntaxe

```
new Effect.Shake('element'[, options]);
$('element').shake({options});
```

Cet effet fonctionne pour les éléments de type block sauf les tables !

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Shake</title>
<style type="text/css">
<!--
#conteneur{
   background-color : blue;
   width : 200px;
   height : 150px;
}
.exemple{
    margin: 40px 100px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>
```

Code de l'exemple

```
<body>
    <div class="exemple">
        <h3>Effet Shake.</h3>
        <div id="conteneur"></div>
        <input type="button" value="Valeurs par défaut" onclick="new Effect.Shake('conteneur');" />
        <input type="button" value="Amplitude 60px" onclick="$('conteneur').shake({distance: 60});" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Effet SwitchOff

Léger clignotement puis disparition de l'élément par une ligne centrale.

Syntaxe

```
new Effect.SwitchOff('element'[, options]);
$('element').switchOff({options});
```

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet SwitchOff</title>
<style type="text/css">
<!--
#conteneur{
    background-color : blue;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effet SwitchOff.</h3>
        <div id="conteneur"></div>
        <input type="button" value="SwitchOff" onclick="new Effect.SwitchOff('conteneur');" />
        <input type="button" value="Rétablir" onclick="$('conteneur').style.display='';" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Cet effet fonctionne pour les éléments de type block sauf les tables !

Effets BlindUp / BlindDown

Ces effets font disparaitre ou apparaitre un élément avec un effet de glissement.

Syntaxe

```
new Effect.BlindUp('element'[, options]);
$('element').blindUp({options});
new Effect.BlindDown('element'[, options]);
$('element').blindDown({options});
```

Option	Description	Défaut
scaleX	Booléen. Détermine si l'effet s'applique de façon horizontale.	false
scaleY	Booléen. Détermine si l'effet s'applique verticalement.	true
scaleContent	Booléen. Détermine si l'effet s'applique au contenu.	false
scaleFromCenter	Booléen. Le positionnement est calé sur le centre de l'élément.	false
scaleMode	'box', 'content' ou objet {originalWidth: largeur_initiale, originalHeight: hauteur_initiale}	'box'
scaleFrom	Entier. Pourcentage définissant la taille initiale de l'élément	100
scaleTo	Entier. Pourcentage définissant la taille finale de l'élément	0

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet BlindUp / BlindDown</title>
<style type="text/css">
<!--
#conteneur{
    background-color : silver;
    text-align : center;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
```

Code de l'exemple

```
    <div class="exemple">
        <h3>Effets Blind.</h3>
        <div id="conteneur">Un peu de texte<br />pour voir l'action de l'effet<br />sur le
 contenu.</div>

  <input type="button" value="BlindDown" onclick="new Effect.BlindDown('conteneur', {scaleX: true});" />

  <input type="button" value="BlindUp" onclick="$('conteneur').blindUp({duration: 2, scaleContent: true});" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Cet effet fonctionne pour les éléments de type block sauf les tables !

Si vous désirez que l'élément soit en display: none par défaut, il faut préciser le style dans l'attribut style de la balise et non dans les feuilles CSS.

Effets SlideUp / SlideDown

Similaire aux effets BlindUp et BlindDown sauf que le contenu suit l'effet.

Syntaxe

```
new Effect.SlideUp('element'[, options]);
$('element').slideUp({options});
new Effect.SlideDown('element'[, options]);
$('element').slideDown({options});
```

Attention, le contenu de l'élément doit être inséré dans une div supplémentaire pour que l'effet s'applique correctement ! (Voir le code de l'exemple).

Option	Description	Défaut
scaleX	Booléen. Détermine si l'effet s'applique de façon horizontale.	false
scaleY	Booléen. Détermine si l'effet s'applique verticalement.	true
scaleContent	Booléen. Détermine si l'effet s'applique au contenu.	false
scaleFromCenter	Booléen. Le positionnement est calé sur le centre de l'élément.	false
scaleMode	'box', 'content' ou objet {originalWidth: largeur_initiale, originalHeight: hauteur_initiale}	'box'
scaleFrom	Entier. Pourcentage définissant la taille initiale de l'élément	100
scaleTo	Entier. Pourcentage définissant la taille finale de l'élément	0

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet SlideUp / SlideDown</title>
<style type="text/css">
<!--
#conteneur{
    background-color : silver;
    text-align : center;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effets Slide.</h3>
        <div id="conteneur">
            <div>Un peu de texte<br />pour voir l'action de l'effet<br />sur le contenu.</div>
        </div>
        <input type="button" value="SlideDown" onclick="new Effect.SlideDown('conteneur');" />
        <input type="button" value="SlideUp" onclick="$('conteneur').slideUp();" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Cet effet fonctionne pour les éléments de type block sauf les tables !

Sous IE6, les éléments block flottants doivent être positionnés en relative.

Si vous désirez que l'élément soit en display: none par défaut, il faut préciser le style dans l'attribut style de la balise et non dans les feuilles CSS.

Effet Pulsate

Cet effet produit des apparitions successives de l'élément comme s'il s'agissait de pulsations.

Syntaxe

```
new Effect.Pulsate('element'[, options]);
$('element').pulsate({options});
```

Option	Description	Défaut
duration	Durée en secondes de l'effet, cette option est commune à tous les effets mais comporte	2.0

	ici une valeur par défaut différente.	
from	Valeur minimale de l'opacité lors des pulsations.	0.0
pulses	Entier, nombre de pulsations.	5

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Pulsate</title>
<style type="text/css">
<!--
#conteneur{
    background-color : blue;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effets Pulsate.</h3>
        <div id="conteneur"></div>
        <input type="button" value="Options par défaut" onclick="new Effect.Pulsate('conteneur');" />

  <input type="button" value="pulses : 8 / from : 0.5" onclick="$('conteneur').pulsate({pulses: 8, from: 0.5});"
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Fonctionne avec tous les éléments HTML à l'exception des tables.

Effets Shrink / Grow

Les effets Shrink et Grow permettent de faire disparaitre / apparaitre un élément par réduction / augmentation de sa taille par rapport à un bord de celui-ci.
Ces deux effets ne sont pas à proprement parler une paire comme nous le verrons pour l'effet toggle, cependant, ayant un comportement semblable, il m'a semblé préférable de les regrouper.

Syntaxe

```
new Effect.Shrink('element'[, options]);
$('element').shrink({options});
new Effect.Grow('element'[, options]);
$('element').grow([options]);
```

Option	Description	Défaut
direction	'center', 'top-left', 'top-right', 'bottom-left' ou 'bottom-right'. Position par rapport à laquelle l'effet se produira.	'center'

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effets Shrink / Grow</title>
<style type="text/css">
<!--
    #conteneur{
    background-color : blue;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effets Shrink / Grow.</h3>
        <div id="conteneur"></div>
        <select id="effet">
            <optgroup label="Valeur de l'option direction">
            <option value="center">center (défaut)</option>
            <option value="top-left">top-left</option>
            <option value="top-right">top-right</option>
            <option value="bottom-left">bottom-left</option>
            <option value="bottom-right">bottom-right</option>
            </optgroup>
        </select><br />

  <input type="button" value="Shrink" onclick="new Effect.Shrink('conteneur', {direction: $('effet').value});" />

  <input type="button" value="Grow" onclick="$('conteneur').grow({direction: $('effet').value});" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Ces effets fonctionnent pour les éléments de type block sauf les tables !

*Il existe un effet similaire, appelé Squish (**voir un exemple**), qui fonctionne de la même manière que Shrink doté de l'option direction : top-left, et n'acceptant pas l'option direction, nous ne reviendront pas dessus.*

Effet Fold

Cet effet fait disparaitre l'élément en réduisant sa hauteur à 5px puis sa largeur.

Syntaxe

```
new Effect.Fold('element'[, options]);
$('element').fold({options});
```

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet Fold</title>
<style type="text/css">
<!--
    #conteneur{
    background-color : blue;
    width : 200px;
    height : 150px;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effet Fold.</h3>
        <div id="conteneur"></div>
        <input type="button" value="Fold" onclick="new Effect.Fold('conteneur');" />
        <input type="button" value="Réinitialiser" onclick="$('conteneur').show();" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```

Ces effets fonctionnent pour les éléments de type block sauf les tables !

Effet toggle

Il ne s'agit pas à proprement parler d'un effet, mais plutôt d'un raccourci permettant d'appliquer un effet selon des paires prédéfinies pour faire apparaitre ou disparaitre un élément.

Syntaxe

```
new Effect.toggle('element'[, 'effet'][, options]);
```

Effet	Paire
'appear' (défaut)	Appear / Fade
'blind'	BlindDown / BlindUp
'slide'	SlideDown / SlideUp

Voir un exemple

Code de l'exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Test effet toggle</title>
<style type="text/css">
<!--
#conteneur{
    background-color : silver;
    width : 300px;
    height : 150px;
    text-align : center;
}
.exemple{
    margin: 40px;
}
-->
</style>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects"></script>
</head>

<body>
    <div class="exemple">
        <h3>Effet toggle.</h3>
        <div id="conteneur">
            <div>
            Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt
 ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
 nisi ut aliquip ex ea commodo consequat.
            </div>
        </div>
        <input type="button" value="appear" onclick="new Effect.toggle('conteneur', 'appear');" />
        <input type="button" value="blind" onclick="new Effect.toggle('conteneur', 'blind');" />
        <input type="button" value="slide" onclick="new Effect.toggle('conteneur', 'slide');" />
    </div>
    <div>
        <input type="button" value="Retour au tuto" onclick="history.go(-1)" />
    </div>
</body>
</html>
```