

Une galerie d'images utilisant la propriété CSS z-index et jQuery

par [Sebastian Senf \(Auteur\)](#) [Didier Mouronval \(Traducteur\)](#)

Date de publication : 26 avril 2009

Dernière mise à jour :

Cet article est la traduction de :  [Create a unique Gallery by using z-index and jQuery.](#)

Dans cet article, nous allons combiner la propriété CSS *z-index* et la puissance de jQuery pour réaliser une galerie d'images avec l'apparence d'une pile de photos. Vous pouvez [voir un exemple](#) ou [télécharger l'archive](#).

Introduction.....	3
Le XHTML, le CSS et les images.....	3
Le XHTML.....	3
Le CSS.....	3
Les images.....	4
Le principe de changement de z-index.....	4
Le code.....	5
Conclusion.....	7
Remerciements.....	7

Introduction

Dans cet article, nous voulons créer une galerie de photos en utilisant la propriété CSS *z-index*. Dans notre exemple, nous avons une pile de photos. En cliquant sur **Suivant**, nous mettons la première photo en dernière position et à l'inverse, en cliquant sur **Précédent**, la dernière photo se mettra en première position. Tout cela est réalisé uniquement en modifiant la valeur de *z-index* en rajoutant bien sûr une animation pour laisser croire qu'il s'agit bien d'une pile de photos.

Vous pouvez [voir un exemple](#) ou [télécharger l'archive](#).

Le XHTML, le CSS et les images

Le XHTML

```
<!-- début de la partie concernant le tutoriel -->
<div class="grid_6 prefix_1 suffix_1" id="gallery">
  <div id="pictures">
    
    
    
    
    
  </div>

  <div class="grid_3 alpha" id="prev">
    <a href="#previous"><< Image précédente</a>
  </div>
  <div class="grid_3 omega" id="next">
    <a href="#next">Image suivante >></a>
  </div>
</div>
<!-- fin de la partie concernant le tutoriel -->
```

Nous avons le conteneur global **gallery**, un bloc **pictures** pour les images et deux boutons **prev** et **next** pour changer d'image.

Nous pouvons mettre autant d'images que nous voulons car le script les traite automatiquement.

Si les noms de classes des *div* ne vous disent rien, ils viennent de  **960 Grid System** et ne servent qu'à positionner les blocs. Ils ne sont pas utilisés par le code jQuery et n'interviennent pas dans le fonctionnement de la galerie.

Le CSS

```
/* début de la partie concernant le tutoriel */
#gallery { position: relative; }
#pictures { position: relative; height: 408px; }
#pictures img { position: absolute; top: 0; left: 0; }

#prev, #next { margin-top: 30px; text-align: center; font-size: 2.0em; }
/* fin de la partie concernant le tutoriel */
```



Le bloc **pictures** est en position *relative* (sous l'en-tête) et a la hauteur d'une image. Les blocs **img** dans le conteneur **pictures** sont en position *absolute*. Comme ils ont tous les propriétés *top* et *left* à 0, ils s'empilent de la dernière image (au-dessus) à la première (en-dessous).

#pictures container



Conteneur des images, position absolue, top et left 0

Les images

Dans l'exemple, nous utilisons des images au format **PNG** qui ont toutes un fond transparent et les mêmes dimensions. Pour créer l'effet de pile, nous avons ajouté une légère ombre portée et une petite rotation dans différentes directions. Je l'ai fait avec Photoshop, mais si vous voulez automatiser cela, vous pouvez utiliser  **Image Magick** ou regarder du côté des  **canvas**.

Le principe de changement de z-index

La propriété *z-index* va représenter la position de l'image, 1 correspond à la dernière position et 5 (il n'y a que 5 images dans l'exemple) à la première.

En cliquant sur **Image suivante**, nous voulons mettre la première image en dernière position, c'est-à-dire affecter son *z-index* à 1. Tous les autres *z-index* devront être incrémentés de 1. Ainsi, l'image qui était en deuxième position (4) se retrouvera en première place et ainsi de suite.

En cliquant sur **Image précédente**, nous voulons mettre la dernière image en première position, c'est-à-dire affecter son *z-index* à 5. Tous les autres *z-index* devront être décrémentés de 1. Ainsi, l'image qui était en première position (5) se retrouvera en deuxième place et ainsi de suite.

Le code

```
$(document).ready(function() { //attend que le DOM soit prêt pour commencer
    var z = 0; //initialise les z-index
    var inAnimation = false; //flag pour tester si nous sommes en cours d'animation

    $('#pictures img').each(function() { //attribue les z-index de départ
        z++; //à la fin, le plus haut z-index sera dans cette variable
        $(this).css('z-index', z); //attribue le z-index à la balise <img>
    });

    function swapFirstLast(isFirst) {
        if(inAnimation) return false; //si une animation est en cours, on ne fait rien
        else inAnimation = true; //modifie le flag pour dire que l'animation est en cours

        var processZindex, direction, newZindex, inDeCrease; //prévoit le cas précédent ou suivant

        if(isFirst) { processZindex = z; direction = '-'; newZindex = 1;
        inDeCrease = 1; } //attribue les variables dans le cas 'suivant'
        else { processZindex = 1; direction = ''; newZindex = z;
        inDeCrease = -1; } //attribue les variables dans le cas 'précédent'

        $('#pictures img').each(function() { //actualise chaque image
            if($(this).css('z-index') == processZindex) { //s'il s'agit de l'image à animer
                $(this).animate({ 'top' : direction +
                $(this).height() + 'px' }, 'slow', function() { //anime l'image au-dessus/en-
                dessous (les images doivent avoir la même hauteur)
                    $(this).css('z-index', newZindex) //attribue le nouveau z-index
                    .animate({ 'top' : '0' }, 'slow', function() { //ramène l'image à sa position initiale
                        inAnimation = false; //modifie le flag pour dire que l'animation est finie
                    });
                });
            } else { //ce n'est pas l'image à animer, on modifie juste le z-index

            $(this).animate({ 'top' : '0' }, 'slow', function() { //il faut attendre la fin de l'animation pour modifier le
            index
                $(this).css('z-index', parseInt($(this).css('z-index')) + inDeCrease); //ajuste le z-index
            });
            }
        });

        return false; //on ne suit pas le href du lien
    }

    $('#next a').click(function() {
        return swapFirstLast(true); //met la première image en dernière position
    });

    $('#prev a').click(function() {
        return swapFirstLast(false); //met la dernière image en première position
    });
});
```

Regardons de plus près les points importants.

```
$('#pictures img').each(function() { //attribue les z-index de départ
    z++; //à la fin, le plus haut z-index sera dans cette variable
    $(this).css('z-index', z); //attribue le z-index à la balise <img>
});
```

Nous n'avons pas fixé de *z-index* dans les balises ni en CSS. Pour avoir un point de départ, nous l'attribuons avec jQuery. L'ordre est celui d'apparition des balises dans le code. A la fin de la boucle sur toutes les images du conteneur **pictures**, nous obtenons le nombre d'images et la valeur maximale de *z-index* dans la variable **z**

```
function swapFirstLast(isFirst) {
```

Nous utilisons la même fonction pour les deux animations.

```
if(inAnimation) return false; //si une animation est en cours, on ne fait rien
else inAnimation = true; //modifie le flag pour dire que l'animation est en cours
```

Nous ne voulons qu'une animation à la fois. Le *flag inAnimation* nous permet de savoir si une animation est en cours. Si c'est le cas, nous sortons directement de la fonction, sinon, on en lance une.

```
var processZindex, direction, newZindex, inDeCrease; //prévoit le cas précédent ou suivant

if(isFirst) { processZindex = z; direction = '-'; newZindex = 1;
  inDeCrease = 1; } //attribue les variables dans le cas 'suivant'
else { processZindex = 1; direction = ''; newZindex = z;
  inDeCrease = -1; } //attribue les variables dans le cas 'précédent'
```

Comme nous n'utilisons qu'une fonction pour deux actions différentes, nous devons ajuster les variables. **processIndex** est la position que nous devons animer, soit la première (suivante) soit la dernière (précédente). **direction** correspond à où l'image va devoir aller, soit au-dessus, soit en-dessous de la pile. **newZindex** sera attribué à l'image à animer, 1 pour la dernière position ou la valeur de **z** pour la première. Enfin, mais pas des moindres, **inDeCrease** permet de savoir si les *z-index* seront incrémentés ou décréments.

```
$('#pictures img').each(function() { //actualise chaque image
```

On boucle sur les images pour modifier les *z-index*.

```
if($(this).css('z-index') == processZindex) { //s'il s'agit de l'image à animer
$(this).animate({ 'top' : direction +
$(this).height() + 'px' }, 'slow', function() { //anime l'image au-dessus/en-
dessous (les images doivent avoir la même hauteur)
$(this).css('z-index', newZindex) //attribue le nouveau z-index
.animate({ 'top' : '0' }, 'slow', function() { //ramène l'image à sa position initiale
inAnimation = false; //modifie le flag pour dire que l'animation est finie
});
});
});
```

D'abord, on vérifie s'il s'agit de l'image à animer. Si c'est le cas, on lance la première animation. Lorsqu'elle est terminée, on attribue le nouveau *z-index* puis on lance la seconde animation. Une fois l'image revenue à sa place, on modifie le *flag inAnimation*.

```
else { //ce n'est pas l'image à animer, on modifie juste le z-index
$(this).animate({ 'top' : '0' }, 'slow', function() { //il faut attendre la fin de l'animation pour modifier le
index
$(this).css('z-index', parseInt($(this).css('z-index')) + inDeCrease); //ajuste le z-index
});
}
```

Si l'image n'est pas celle à animer nous avons juste à modifier le *z-index*. Nous récupérons son *z-index* et l'incrémentons / décréments selon le cas. Nous lançons une animation (qui ne fait rien) pour être sûrs de ne mettre à jour le *z-index* qu'une fois que la première animation (placer l'image au-dessus des autres) est terminée.

```
$('#next a').click(function() {  
    return swapFirstLast(true); //met la première image en dernière position  
});  
  
$('#prev a').click(function() {  
    return swapFirstLast(false); //met la dernière image en première position  
});
```

Ici, nous affectons au clic sur les boutons **prev** et **next** l'appel de la fonction. Nous retournons le résultat de la fonction (qui vaut toujours *false*) de façon à éviter de suivre le lien.

Conclusion

Nous avons créé une galerie de bonne allure avec juste quelques lignes de code. Bien sûr, on pourrait ajouter d'autres fonctionnalités (image aléatoire, mélange des images...) mais maintenant, vous savez comment combiner des propriétés CSS et la puissance de jQuery. N'hésitez pas à modifier le code selon vos besoins et de nous le faire partager. Vous pouvez aussi laisser vos commentaires.

Vous pouvez [voir un exemple](#) ou [télécharger l'archive](#).

Remerciements

Un grand merci à **Kerod** pour sa relecture avisée et ses conseils précieux !