CSS Sprites2 - C'est l'heure du JavaScript

par Dave Shea (Auteur) Joris Crozier (Traducteur)

Date de publication : 26 février 2009

Dernière mise à jour :

Cet article est la traduction de **CSS Sprites2 - It's JavaScript Time** (disponible **ici**).

I - Introduction

Le sens du mouvement est souvent le différentiateur entre les sites en Flash lourds et les sites basés sur les standards du Web. Les interfaces flash ont toujours semblé beaucoup plus vivantes - répondant aux interactions avec l'utilisateur de manière dynamique que les sites basés sur les standards du web n'ont jamais pu reproduire.

Plus tard, tout a changé, bien sûr, avec une réapparition des effets dynamiques dans les interfaces, aidés par les librairies JavaScript qui le font facilement ? librairies comme **Prototype**, **Scriptaculous**, **Moo**,**YUI**, **MochiKit** (et je pourrais en citer d?autres).Il est vraiment temps de revisiter la technique des Sprites CSS datant d'il y a quatre ans, et de voir si on peut exclamer un peu de mouvement de notre part.

L'exemple suivant démontre les CSS Sptite2 en ligne, la technique sera couverte par cet article : {exemple}

II - Plus loin avec jQuery

Donc, voici le premier avertissement : nous allons compter sur jQuery pour que cela se réalise. **jQuery** est une librairie JavaScript mature qui fait les mêmes choses ordonnées que les autres librairies JavaScript, et a un avantage qui fait qu?il se prête particulièrement bien a l?extension des Sprites CSS : jQuery nous permet de sélectionner des éléments sur une page en utilisant une syntaxe CSS que nous connaissons déjà.

Nous devons noter les Kb non triviaux que la librairie va ajouter au chargement initial de nos pages. Un fichier externe JavaScript peut être mis en cache, bien sûr, c'est donc une seule fois seulement que ce chargement impactera l'utilisateur qui vient sur notre site. La version la plus compacte de jQuery pèse 15 Kb, ce sont des frais généraux inévitables et ce peut être un sujet d'inquiétude. Si vous utilisez déjà jQuery sur votre site pour d'autres raisons, les frais généraux sont un cul de sac. SI vous êtes intéressés pour l'ajouter seulement pour cette technique, considérez le poids du fichier et demandez-vous si l'effet le vaut. (Depuis que Google héberge **jQuery**, vous pouvez pointer sur ses versions de la librairie, comme nous le faisons dans les exemples suivants, et espérer que vos utilisateurs aient déjà cette URL en cache dans leurs navigateurs)

Comme pour les autres librairies JavaScript ? Il n'y a absolument aucune raison pour que vous ne deviez ou ne pussiez pas les utiliser, considérez cet article comme une invitation ouverte à porter cette technique sur la librairie de votre choix.

III - HTML et configuration CSS basique

La première chose que nous voulons faire est un état sans script JavaScript par défaut pour les utilisateurs. (Nous avons lu l'article de **Jeremy Keith** il y a un an et nous sommes maintenant de gros fans du scripting DOM non intrusif, naturellement)

Nous avions déjà une méthode purement CSS pour les rollovers , donc commençons par construire notre navigation pour qu'elle fonctionne avec une configuration de Sprite CSS basique. Et parce que nous sommes paresseux, nous ne reconstruirons pas le rollover une deuxième fois, nous allons juste réutiliser les fondations plus tard et ajouter jQuery par-dessus. Nous y arriverons dans un instant.

Je vais laisser le pourquoi du comment des Sprites CSS dans **l'article original**, mais il y a quelques choses à clarifier avant. Commençons par le HTML. Prêtez une grande attention à la structure - nous y référerons plusieurs fois.

```
<ul class="nav current-about">
    <li class="home"><a href="#">Home</a></li>
    <li class="about"><a href="#">About</a></li>

    <li class="services"><a href="#">Services</a></li>
    <li class="contact"><a href="#">Contact</a></li>
```

```
</ul>
```

Chaque classe atteint un objectif, le conteneur UL a une classe nav qui nous permette de le cibler dans notre CSS (et plus tard dans le JavaScript), tout comme une classe .current-about que nous allons utiliser pour indiquer, au sein de la navigation, quelle page ou section du site nous sommes en train de regarder. Chaque élément LI a sa propre classe unique, que nous utiliserons aussi pour faire du ciblage.

Jusqu'ici c'est bon, notre balise de navigation est une liste HTML simple et accessible, et nous avons assez de classes pour faire marcher nos Sprites.

```
.nav {
    width: 401px;
    height: 48px;
    background: url(../i/blue-nav.gif) no-repeat;
    position: absolute;
    top: 100px;
    left: 100px;
}
```

Nous avons réglé la valeur de la position sur absolute pour changer l'offset de positionnement du LI, à la place de l'élément body. Nous aurions pu utiliser relative à la place, pour accomplir la même chose tout en laissant l'élément .nav dans le flux des documents. Il y a des raisons pour le faire de cette manière, mais pour l'heure nous resterons avec la valeur absolute.

Pour lus d'informations sur le couple absolute/relative, voyez ***l'article*** *de Douglas Bowman sur le sujet.*

Le coeur de notre technique Sprites se fait en appliquant une image de fond à tous les éléments nav, et de les positionner absolument dans les parents UL.

```
.nav li a:link, .nav li a:visited {
    position: absolute;
    top: 0;
    height: 48px;
    text-indent: -9000px;
    overflow: hidden;
}
.nav .home a:link, .nav .home a:visited {
    left: 23px;
    width: 76px;
}
.nav .home a:hover, .nav .home a:focus {
    background: url(../i/blue-nav.gif) no-repeat -23px -49px;
}
.nav .home a:active {
    background: url(../i/blue-nav.gif) no-repeat -23px -98px;
}
```

Nous allons un peu plus loin que l'article original en appliquant les états :focus et :ative . L'ancien est une addition mineure pour déclencher l'image quand une ancre est sujette aux états :focus ou :hover. Le dernier ajoute un nouvel état quand l'utilisateur click sur un élément. Ni l'un ni l'autre n'est essentiel, bien que ce soit une bonne idée de les définir tous les deux. La règle overflow :hidden est nouvelle aussi - elle est juste la pour prévenir les navigateurs contre le prolongement d'une ligne pointillée depuis la positon de l'élément de toute manière hors de la partie gauche de l'écran jusqu'au texte indenté négativement.

Exemple 1: Configuration des Sprites CSS basique //TODO LIEN

Cela nous amène à notre vrai point de départ - un menu Sprite qui fonctionne, complet avec les éléments de navigation sélectionnés. Maintenant, nous allons l'étendre.

IV - Initialisation avec jQuery

Notez que tout ce qui va suivre devra être placé à l'intérieur d'une fonction jQuery ce qui nous assure que le code sera exécuté une fois que la page sera complètement chargée. Les extraits de code que vous voyez au-dessous supposent qu'ils fonctionnent à l'intérieur de cette fonction, donc si vous rencontrez des erreurs, rappelez-vous de vérifier ou vous les avez placés.

```
$(document).ready(function(){
    // Tout doit être placé ici
});
```

Puisque le menu Sprite est notre état de chute quand le JavaScript est désactivé (si nous l'avons fait correctement), nous devrions nous débarrasser de ces fonds d'image CSS appliqués sur le survol des éléments, car nous allons créer le nôtre dans le script suivant :

```
$(".nav").children("li").each(function() {
    $(this).children("a").css({backgroundImage:"none"});
});
```

Sur la première ligne, nous requêtons tous les éléments de la classe .nav et attachons une nouvelle fonction à tous les éléments enfants LI qu'elles contiennent. Cette fonction se trouve sur la seconde ligne et requête l'objet THIS pour tous les éléments enfants A. Si elle les trouve, elle règle la propriété CSS background-image sur none. Dans le contexte donné, THIS veut dire les éléments LI sur lesquels la fonction agit.

Exemple 2: Désactiver les hover CSS avec jQuery. //TODO LIEN

Cela fonctionne?.Mais nous avons aussi perdu notre élément sélectionné courant dans le process. Donc, nous devons faire une petite vérification pour voir quel élément nous avons identifié avec la classe current-(ce qu'on veut) que nous avons appliqué au parent UL, et passer celle-ci à notre suppresseur d'image de fond. Le code précédent a besoin d'être un peu étendu :

```
$(".nav").children("li").each(function() {
    var current = "nav current-" + ($(this).attr("class"));
    var parentClass = $(".nav").attr("class");
    if (parentClass != current) {
        $(this).children("a").css({backgroundImage:"none"});
    }
});
```

La seconde ligne créé maintenant une variable nommée current qui utilise chaque classe LI séquentiellement pour créer une chaîne de caractères qui correspondra à la classe parente UL , si le LI particulier correspond a l'élément actuellement sélectionné. La troisième ligne crée une deuxième variable qui lit la valeur actuelle directement depuis l'élément UL. Finalement, les quatre lignes comparent les deux variables. Se elles ne correspondent pas, seulement après nous réglons la propriété CSS background-image de l'élément A. Cela passe les changements de l'image de fond pour l'élément actuellement sélectionné, ce qui est exactement ce que nous voulions.

V - Attacher les événements

Maintenant, nous devons attacher une fonction à tous les éléments LI et pour tous les événements d'interaction que nous voulons styliser. Créons une fonction pour ceci, appelée attachNavEvents :

```
function attachNavEvents(parent, myClass) {
    $(parent + " ." + myClass).mouseover(function() {
        // do things here
```

```
    }).mouseout(function() {
        // do things here
    }).mousedown(function() {
        // do things here
    }).mouseup(function() {
        // do things here
    });
}
```

Cette fonction prend 2 arguments. Le premier est une chaîne de caractères contenant le littéral de la classe de l'élément parent, le complète avec la période précédente, comme vous pourrez le voir quand nous l'appellerons plus tard. Le second est une chaîne de caractères contenant la classe du LI particulier auquel nous voulons attacher l'événement. Nous allons combiner les 2 sur la première ligne de la fonction pour créer un sélecteur jQuery pour cibler le même élément en tant que sélecteur CSS descendant de, par exemple, l'élément .nav, .home (lequel dépend de l'élément passé en paramètre à la fonction, évidemment).

Parce que jQuery nous permet de chaîne de multiples fonctions sur un seul objet, nous sommes capable de créer toutes les liaisons aux événements en même temps. Le chaînage est un concept unique de jQuery. C'est très rusé pour lier votre esprit autour (?), ce n'est pas essentiel de comprendre comment cela fonctionne, donc si vous êtes perdus, prenez simplement le fait que ça fonctionne comme acquis, pour l'instant.

Maintenant, nous allons attacher ces fonctions à tous nos objets de navigation. Ce qui suit est une façon verbeuse - nous l'optimiserons plus tard- mais pour l'instant appliquons la fonction à tous les éléments LI. Pour argument, nous passerons le parent de chaque élément LI, ainsi que la classe du LI elle-même.

```
attachNavEvents(".nav", "home");
attachNavEvents(".nav", "about");
attachNavEvents(".nav", "services");
attachNavEvents(".nav", "contact");
```

Cela ne marche pas pour l'instant, mais nous allons réparer ça.

Exemple 3: Script basique de configuration des événements.//TODO LIEN

VI - La théorie

Je vais vous expliquer ce qu'il va se passer ensuite - restez avec moi, il est important de comprendre ce qui se passe ici, car vous devrez styliser les éléments que nous manipulons ici.

Pour chacun des liens, nous créons un nouvel élément div pour chaque élément LI que nous ciblons, qui sera utilisé pour notre effet jQuery. Nous appliquerons l'image nav à cette div en utilisant la même règle background-image que nous avons utilisée pour l'élément a dans notre configuration CSS du Sprite. À travers des essais et des erreurs, j'ai trouvé que cette création de nouvelle div est moins gourmande qu'appliquer l'effet jQuery directement sur l'élément existant- c'est donc une étape nécessaire.

Le style pour cette div doit déjà exister dans le CSS. Nous allons créer une nouvelle classe pour l'élément LI (.nav-home), basée sur la classe de l'élément LI ciblé (comme ça , ça ne rentrera pas en conflit avec ce que nous avons créé au début), et ajoutez le style.

```
.nav-home {
    position: absolute;
    top: 0;
    left: 23px;
    width: 76px;
    height: 48px;
    background: url(../i/blue-nav.gif) no-repeat -23px -49px;
}
```

VII - La pratique

Maintenant, il est temps d'ajouter les effets, quand l'événement mouseover est déclenché, nous allons créer l'élément div et lui donner la classe précédemment mentionnée. Nous avons besoin de ça invisible pour commencer avant de fondre, donc nous allons utiliser les fonctions CSS de jQuery pour régler la valeur css de display sur none. Finalement, nous utiliserons la fonction jQuery FadeIn pour fondre de " caché" à " visible ", et lui passer un argument de 200 à cette fonction pour spécifier la durée de l'animation en millisecondes.

```
function attachNavEvents(parent, myClass) {
    $(parent + " ." + myClass).mouseover(function() {
        $(this).before('<div class="nav-' + myClass + "
        '"></div>');
        $("div.nav-" + myClass).css({display:"none"}) "
        .fadeIn(200);
    });
}
```

Maintenant, nous faisons l'inverse sur l'événement mouseout - nous allons faire un fondu inversé sur le div. Une fois le fondu finit, nous allons nettoyer nous-mêmes en le supprimant du DOM. C'est à ça que notre fonction attachNavEvents doit ressembler.

```
function attachNavEvents(parent, myClass) {
    $(parent + " ." + myClass).mouseover(function() {
        $(this).before('<div class="nav-' + myClass + "
         '"></div>');
        $("div.nav-" + myClass).css({display:"none"}) "
        .fadeIn(200);
    }).mouseout(function() {
        // fade out & destroy pseudo-link
        $("div.nav-" + myClass).fadeOut(200, function() {
            $(this).remove();
        });
    });
}
```

Et c'est à peu près tout pour les hovers :

Exemple 4: Evénements hovers scriptés.//TODO LIEN

Nous ferions mieux de faire quelque chose pour les événements mousedown et mouseup aussi, si nous avons précédemment défini un changement pour l'état :active dans le CSS. Nous allons avoir besoin d'une classe différente pour les événements hover pour que nous puissions les cibler uniquement en CSS, donc changeons la classe sur le mousedown. Nous allons aussi l'inverser sur l'événement mouseup pour restaurer l'etat :hover , puisque l'utilisateur a pu ne pas avoir bougé sa souris hors de l'élément de navigation. Voici a quoi la fonction révisée attachNavEvents ressemble :

```
function attachNavEvents(parent, myClass) {
    $(parent + " ." + myClass).mouseover(function() {
        $(this).before('<div class="nav-' + myClass + "
        '"></div>');
        $("div.nav-" + myClass).css({display:"none"})"
        .fadeIn(200);
    }).mouseout(function() {
        $("div.nav-" + myClass).fadeOut(200, function() {
            $(this).remove();
        });
    }).mousedown(function() {
        $("div.nav-" + myClass).attr("class", "nav-" "
        + myClass + "-click");
    }).mouseup(function() {
```

```
        $("div.nav-" + myClass + "-click").attr("class", "
        "nav-" + myClass);
    });
}
```

Nous pouvons réutiliser le style div hover, en modifiant légèrement la position du fond pour ajuster les parties de notre image de Sprite principal qui est montré sur le clique :

```
.nav-home, .nav-home-click {
    position: absolute;
    top: 0;
    left: 23px;

    width: 76px;
    height: 48px;
    background: url(../i/blue-nav.gif) no-repeat -23px -49px;
}
.nav-home-click {
    background: url(../i/blue-nav.gif) no-repeat -23px -98px;
}
```

Maintenant nous avons les hovers, l'élément actuellement sélectionné, et les événements sur le clique qui fonctionnent.

Exemple 5: Tout mettre ensemble. //TODO LIEN

VIII - Autres considérations

Nous ne sommes pas limites a l'effet de fondu, jQuery possède une fonction slideUp/slideDown que nous pouvons très bien utiliser (qui est montrée dans le second exemple de cet article). Ou, nous pouvons être vraiment plus fantaisistes et créer de vraie animation CSS personnalisée en utilisant les fonctions jQuery animate (comme montré dans le troisième exemple). Un mot d'attention a propos d'animate - le résultat peut être un peu erratique, comme vous avez dû le voir dans l'exemple.

La fonctionnalité inter-navigateurs est un peu une faveur, jQuery marche à travers tous les navigateurs modernes, donc tout ce que vous voyez la fonctionne sur IE6+, Firefox, Safari, Opera, etc. Nous avons aussi compté sur plusieurs scénarios de dégradation gracieux. Si un utilisateur a désactivé JavaScript, ils ont des Sprites CSS basiques. Si ils ont désactivé JavaScript et CSS, ils ont une liste HTML basique. ET nous avons les autres bénéfices des Sprites aussi, puisque nous employons toujours une image simple pour tous les divers états et effets de navigation.

Bien qu'on ne l'exige pas, on suggère fortement que vous choisissiez la subtilité ; les vitesses d'animation de plus que quelque cent millisecondes peuvent être amusantes pour commencer, mais elles taperont rapidement sur les nerfs de ceux qui emploient votre site que vous construisez après que la nouveauté soit découverte. Errez du côté des vitesses d'animation plus rapides, plutôt que plus lentes.

Un problème potentiel que vous pourriez courir est quand l'autre texte de la page " clignote " pendant les animations. C'est une solution compliquée qui doit être mise en place avec le rendu secondaire de pixel commun dans les logiciels d'exploitation modernes, et la meilleure solution semble d'appliquer une valeur légèrement moins opaque en opacité pour forcer un mode particulier de rendu des textes. SI vous ajoutez cela a votre CSS, les flash devraient être nettoyés aux dépens du texte régulier anti-aliasé au lieu pixel secondaire anti-aliasé.

```
p {
    opacity 0.9999;
}
```

Dans la démonstration, ceci est appliqué à l'élément p. Nous pouvons cette règle a n'importe quel élément de la page ainsi sélectionnons quelque chose d'inoffensif.

IX - Emballé c?est pesé

Vous ne devez pas réellement vous rappeler de tous les scripts contenus dans cet article, puisqu'il y a une fonction préconstruite vous attendant dans cet exemple final. Utilisant le JavaScript dans le fichier HTML comme référence, vous devez seulement éditer une ligne simple de JavaScript pour appliquer Sprites2 à votre site:

```
$(document).ready(function(){
    generateSprites(".nav", "current-", true, 150, "slide");
});
```

La fonction generateSprites prend cinq arguments :

1 La classe primaire de votre élément UL, incluant la période.
2 Le préfixe que vous utilisez pour les éléments sélectionne, pour une classe sélectionnée de selected-about , utilisez selected- en tant que valeur
3 Un booléen pour indiquer si vous stylisez l?état :active . Mettez-le à true si vous avez défini l?état :active et l? équivalent jQuery dans votre CSS, sinon mettez-le à false.
4 La vitesse de l?animation en millisecondes. 300 = 0.3 secondes
5 Votre style d?animation préférée, en tant que chaîne de caractère. Mettez «Slide » ou « Fade », par défaut c? est la deuxième valeur.

Exemple 6: Une ligne facile de script à modifier, merci à la fonction de pré construction. //TODO LIEN

Vous devrez aussi positionner et styliser les différents éléments scriptes dans votre CSS, donc soyez libre d'utiliser le fichier CSS d'exemple dans cet article en tant qu'exemple.

X - Note de pied de page

Durant l'écriture de cet article, une **technique similaire** a été écrite ailleurs quoique sans notre point de chute CSS Sprites. Nous avons aussi découvert un menu jQuery animé très différent que vous devriez trouver utile.

XI - Liens

Vous pouvez aussi aller voir mes autres **traductions**.