TP : le carrousel - Correction

Votre carrousel fonctionne bien ? Ou en est-il toujours à la phase théorique ? Quoi qu'il en soit, nous vous proposons comme de juste une correction. Suivez-la bien pas à pas, et vous saurez quelles ont été vos erreurs si vous en avez commis.

Comme pour le premier TP, nous vous avions fourni le code HTML minimal permettant de constituer une base de travail :

```
<div id="carrousel">
    <ul>
        <li><img src="http://lorempixel.com/700/200/" /></li>
        <li><img src="http://lorempixel.com/g/700/200/" /></li>
        <li><img src="http://lorempixel.com/700/200/abstract/" /
></li>
    </ul>
</div>
```

Notre carrousel possède donc une structure bien simple : une liste non-ordonnée d'images. Il est difficile d'être plus minimaliste. La première chose à faire était évidemment d'inclure jQuery, si vous avez oublié de faire cela, alors nous vous conseillons vivement de revoir certains chapitres.

```
<div id="carrousel">
    <ul>
        <li><img src="http://lorempixel.com/700/200/" /></li>
    <li><img src="http://lorempixel.com/g/700/200/" /></li>
    <li><img src="http://lorempixel.com/700/200/abstract/" /></li>
    </ul>
</div>

<!-- on inclut la bibliothèque depuis les serveurs de Google -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/
jquery.min.js"></script>
```

Il fallait également inclure correctement le style CSS minimal du carrousel, qui consistait à définir les positions absolues des images et la position relative du carrousel. C'est un point très important, sans cela, vous auriez des soucis de disposition très embêtants !

Explications du code jQuery

De même que pour l'inclusion de la librairie, il fallait tout d'abord s'assurer que le DOM était correctement chargé, grâce à l'évènement `ready()` lancé sur le document :

```
$(document).ready(function(){

    // notre code ici

});
```

Plusieurs variables devaient ensuite être définies. Une bonne chose à faire avant de se lancer dans un code jQuery est de lister tous les objets que l'on va devoir utiliser, ainsi que toutes les informations dont on a besoin. Ici, il suffisait de cibler seulement deux éléments : le bloc du carrousel, et les images contenues par celui-ci. Ce genre de réflexion était plutôt facile, contrairement à la suite : qu'a-t-on besoin de connaître pour avancer dans notre système ?

Nous vous avions mis sur une petite piste : les index des images. Grâce à un système de compteur, vous pouviez contrôler l'affichage des images par leur index. Il était donc d'usage de définir le nombre d'images qu'il y avait dans la liste, puis de soustraire 1 pour obtenir l'index de la dernière diapositive (rappelez-vous que l'on commence de compter les index à partir de 0, et non de 1).

Une fois ceci fait, il fallait obtenir l'image courante, c'est-à-dire la première image sur laquelle le slider devait commencer. Soyons classique, et partons de la toute première image, celle qui possède l'index 0. Grâce à la fonction `eq()`, vous pouviez la cibler très facilement.

```
var $carrousel = $('#carrousel'), // on cible le bloc du carrousel
    $img = $('#carrousel img'), // on cible les images contenues
dans le carrousel
    indexImg = $img.length - 1, // on définit l'index du dernier
élément
    i = 0, // on initialise un compteur
    $currentImg = $img.eq(i); // enfin, on cible l'image courante,
qui possède l'index i (0 pour l'instant)
```

Une fois les variables créées, il fallait s'assurer que toutes les images sauf une étaient invisibles. La propriété CSS `display : none` vous permettait de faire cela : on cache toutes les images, puis on affiche seulement l'image courante.

```
$img.css('display', 'none'); // on cache les images
$currentImg.css('display', 'block'); // on affiche seulement
l'image courante
```

Créer les contrôles : image précédente et image suivante

La base de notre système posée, employons nous à présent à travailler sur les contrôles des images, c'est-à-dire les fonctions d'affichage des images précédentes ou suivantes. La structure HTML ne comprenait pas les deux éléments permettant d'interagir avec la page, il fallait donc les créer au moyen d'une méthode de manipulation du DOM. Nous vous suggérons `append()`, bien que vous pouviez tout aussi bien employer d'autre fonctions.

Nous avons choisi de créer un bloc `div` contenant deux `span`, ayant respectivement la classe `.prev` et `.next`.

```
$carrousel.append('<div class="controls"> <span
class="prev">Precedent</span> <span class="next">Suivant</span> </
div>');
```

En écoutant le clic sur ces éléments, il était possible de déterminer s'il fallait passer à l'image suivante, ou l'image précédente. Dans les deux cas, il suffisait de jouer avec l'index des diapositives : si l'on va à l'image suivante, l'index s'incrémente de 1, si on va à l'image précédente, il se décrémente de 1.

```
$('.next').click(function(){ // image suivante
    i++; // on incrémente le compteur
    $img.css('display', 'none'); // on cache les images
    $currentImg = $img.eq(i); // on définit la nouvelle image
    $currentImg.css('display', 'block'); // puis on l'affiche
});
```

```
$('.prev').click(function(){ // image précédente
    i--; // on décrémente le compteur, puis on réalise la même
chose que pour la fonction "suivante"
    $img.css('display', 'none');
    $currentImg = $img.eq(i);
    $currentImg.css('display', 'block');
});
```

On se heurte alors à un nouveau problème : si l'on clique trop de fois sur une des deux fonctions, alors le compteur ne suit plus les index des images. On peut alors se retrouver avec une image courante qui n'existe pas, et qui ne peut donc pas s'afficher. Pour remédier à ce problème, il suffit de s'assurer grâce à une condition que le compteur ne dépasse pas le dernier index, et ne puisse pas aller en dessous de 0 :

```
$('.next').click(function(){ // image suivante
    i++; // on incrémente le compteur
    if( i <= indexImg ){
        $img.css('display', 'none'); // on cache les images
        $currentImg = $img.eq(i); // on définit la nouvelle image
        $currentImg.css('display', 'block'); // puis on l'affiche
    }
    else{
        i = indexImg;
    }
});
```

```
$('.prev').click(function(){ // image précédente
    i--; // on décrémente le compteur, puis on réalise la même
chose que pour la fonction "suivante"
    if( i >= 0 ){
        $img.css('display', 'none');
        $currentImg = $img.eq(i);
        $currentImg.css('display', 'block');
    }
    else{
        i = 0;
    }
});
```

Créer le défilement d'images automatique

Enfin, pour terminer, vous deviez créer un défilement d'images automatique. Nous vous conseillions pour cela d'utiliser la fonction `setTimeout()` couplée à une autre fonction pour créer une boucle infinie, et répéter le défilement. Celui-ci ne va que dans un sens, ce qui est logique : il défile vers la droite, c'est-à-dire qu'il affiche à chaque fois l'image suivante. Rien de bien compliqué : il suffit d'incrémenter le compteur. Seulement, il convenait de faire attention à ce qu'il ne dépasse pas l'index de la dernière image, en le remettant à 0 le cas échéant.

```
function slideImg(){
    setTimeout(function(){ // on utilise une fonction anonyme
        if(i < indexImg){ // si le compteur est inférieur au
dernier index
        i++; // on l'incrémente
    }
    else{ // sinon, on le remet à 0 (première image)
        i = 0;
    }
    $img.css('display', 'none');
    $currentImg = $img.eq(i);
    $currentImg.css('display', 'block');
    slideImg(); // on oublie pas de relancer la fonction à la fin
    }, 7000); // on définit l'intervalle à 7000 millisecondes (7s)
}
slideImg(); // enfin, on lance la fonction une première fois
```

Et voilà, le carrousel est terminé !

```
$(document).ready(function(){

    $carrousel = $('#carrousel'), // on cible le bloc du carrousel
    $img = $('#carrousel img'), // on cible les images contenues dans le carrousel
    indexImg = $img.length - 1, // on définit l'index du dernier élément
    i = 0, // on initialise un compteur
    $currentImg = $img.eq(i); // enfin, on cible l'image courante, qui possède l'index i (0 pour l'instant)

$img.css('display', 'none'); // on cache les images
$currentImg.css('display', 'block'); // on affiche seulement l'image courante

$carrousel.append('<div class="controls"> <span class="prev">Precedent</span> <span class="next">Suivant</span> </div>');

$('.next').click(function(){ // image suivante

    i++; // on incrémente le compteur

    if( i <= indexImg ){
        $img.css('display', 'none'); // on cache les images
        $currentImg = $img.eq(i); // on définit la nouvelle image
        $currentImg.css('display', 'block'); // puis on l'affiche
    }
    else{
        i = indexImg;
    }

});

$('.prev').click(function(){ // image précédente

    i--; // on décrémente le compteur, puis on réalise la même chose que pour la fonction "suivante"

    if( i >= 0 ){
        $img.css('display', 'none');
        $currentImg = $img.eq(i);
        $currentImg.css('display', 'block');
    }
    else{
        i = 0;
    }

});

function slideImg(){
    setTimeout(function(){ // on utilise une fonction anonyme

        if(i < indexImg){ // si le compteur est inférieur au dernier index
        i++; // on l'incrémente
    }
    else{ // sinon, on le remet à 0 (première image)
        i = 0;
    }

    $img.css('display', 'none');

    $currentImg = $img.eq(i);
    $currentImg.css('display', 'block');

    slideImg(); // on oublie pas de relancer la fonction à la fin

    }, 7000); // on définit l'intervalle à 7000 millisecondes (7s)
}

slideImg(); // enfin, on lance la fonction une première fois

});
```

Améliorations

Nous venons de vous présenter un carrousel très basique. La plupart de ceux que vous rencontrerez sur la toile sont beaucoup plus sophistiqués, mais rappelez-vous qu'ils fonctionnent presque tous de la même manière ! Que diriez-vous donc d'améliorer le votre ? Voici quelques pistes :

- le défilement d'images n'est pas très esthétique : à la place du système de `display`, essayez de mettre en place des effets d'animation sympatiques !

- pour le moment, les contrôles sont très basiques : il n'est pas possible d'aller à une image précise en cliquant une seule fois. Que diriez-vous de réaliser une liste de boutons représentant chacun une diapositive ?

- de même, il est possible de faire des miniatures des images et de les afficher en dessous du carrousel pour faire une bien meilleure navigation !

- ...

Les possibilités sont infinies, on peut toujours trouver de nouvelles idées, qu'elles soient bonnes ou non. N'hésitez pas à regarder comment certains carrousel sont fait, vous risquez d'avoir de fabuleuses surprises ! 

Voilà un TP qui ne vous aura pas fait chômer !

[Source : https://openclassrooms.com/fr/courses/1567926-un-site-web-dynamique-avec-jquery/1569419-tp-le-carrousel]