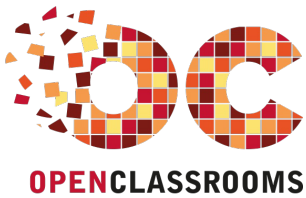
Animez votre site avec CSS3 !

Par Adrien Guéret (KorHosik)



www.openclassrooms.com

Dernière mise à jour le 17/09/2011

Sommaire

Animez votre site avec CSS3 !

Par  Adrien Guéret (KorHosik)

Mise à jour : 17/09/2011
Difficulté : Facile Durée d'étude : 45 minutes

Avec l'arrivée de CSS3, les développeurs web ont vu leurs possibilités en *webdesign* se multiplier : que ce soient les coins arrondis, les fonds en dégradé ou encore les rotations, les nouvelles fonctionnalités utilisables laissent rêveur.

L'une des nouveautés principales est l'animation des éléments, appelée également transition. Et c'est ce que je vous propose de découvrir dans ce tutoriel !

Sommaire du tutoriel :

Le principe des transitions

Avant de commencer, il est bon de savoir *quand* a lieu une transition en CSS. Vous n'êtes pas sans savoir qu'il est possible de modifier le style d'un élément d'une page lorsqu'un certain événement se déclenche. L'exemple le plus basique est celui des liens qui changent de couleur lors du passage de la souris.

La transition a lieu entre l'état initial de l'élément (un lien bleu, par exemple) et son état final après l'événement (admettons que le lien devienne rouge). Grâce à la transition, notre lien ne passera pas du bleu au rouge directement, mais il changera progressivement de couleur.

Bien entendu, il est possible de faire appel aux transitions sur d'autres propriétés que la couleur comme les dimensions des éléments ou leur position. Je vous invite à consulter cette page pour prendre connaissance des propriétés pouvant être animées. Notez cependant que la propriété `transform` n'y est pas répertoriée alors qu'elle peut pourtant bel et bien être utilisée : en effet, il est tout à fait possible d'effectuer une transition lors d'une rotation, par exemple.

Pour conclure avec cette première partie, retenez bien ceci : **une transition, c'est tout simplement le changement d'état progressif d'une propriété CSS d'un état initial vers un état final voulu**.

Notre première transition

Finalement, comment faire des transitions ? C'est bien plus simple qu'il n'y paraît puisqu'il s'agit en fait de simples propriétés CSS.

Nous avons à notre disposition cinq propriétés pour effectuer des transitions et nous commencerons par étudier les deux principales :

- transition-property ;
- transition-duration.

Les normes CSS3 n'étant pas encore finalisées, les navigateurs n'interprètent pas tous ces propriétés, ou alors les interprètent à leur manière !
Si ce cher Internet Explorer les ignore complètement (même dans sa version 9), il est nécessaire de mettre des préfixes

pour Firefox, Google Chrome, Safari et Opera. Ainsi, pour Firefox, il est préférable d'écrire **-moz-transition-property**, pour Google Chrome et Safari **-webkit-transition-property** et pour Opera **-o-transition-property**.
Pour plus de clarté dans ce tutoriel, les propriétés seront utilisées sans leur préfixe.

La propriété **transition-property** permet d'indiquer les propriétés de notre élément qui doivent être animées. Par défaut, sa valeur est fixée à « **all** », ce qui signifie que toutes les propriétés seront sujettes à la transition.
Si nous souhaitons appliquer des transitions sur plusieurs propriétés, il suffit de les séparer par des virgules :

Code : CSS

```
p
{
    /* Seules la couleur du texte et les dimensions
de nos paragraphes auront une transition. */
    transition-property: color, width, height;
}
```

Il nous faut maintenant utiliser la propriété **transition-duration** pour indiquer le temps que durera la transition entre l'état initial de la propriété et son état final. Par défaut, cette propriété vaut **0s**, il n'y a donc pas de transition.
Il est possible d'indiquer un temps en secondes ou en millisecondes. Pour cela, il suffit d'ajouter respectivement **s** ou **ms** à la suite de notre valeur.

Code : CSS

```
p
{
    /* Seules la couleur du texte et les dimensions
de nos paragraphes auront une transition. */
    transition-property: color, width, height;

    /* La transition durera trois secondes. */
    transition-duration: 3s;
}
```

Nos transitions sont définies, il ne nous reste plus qu'à indiquer *quand* les utiliser ! Et c'est là que c'est magique : les transitions auront lieu automatiquement dès que les propriétés concernées seront modifiées !

Modifions par exemple le style de nos paragraphes au passage de la souris. Je vous rappelle pour cela qu'il faut utiliser le pseudo-format `:hover` :

Code : CSS

```
/* Style par défaut de nos paragraphes
(= état initial). */
p
{
    /* Seules la couleur du texte et les dimensions
de nos paragraphes auront une transition. */
    transition-property: color, width, height;

    /* La transition durera trois secondes. */
    transition-duration: 3s;

    /* On stylise un peu plus nos paragraphes. */
    width: 400px;
    height: 200px;
    color: #fff;
```

```
    background-color: #a5a5a5;
    border: 2px solid #000;
    margin: auto;
    text-align: center;
}

/* Style lors du passage de la souris
(= état final). */
p:hover
{
   width: 300px;
   height: 300px;
   color: #000;
}
```

Il ne nous reste plus qu'à tester tout ça avec du code HTML5, extrêmement basique. Notez que le code CSS3 ci-dessus est enregistré dans le fichier `style.css`.

Code : HTML

```
<!DOCTYPE html>
<html>
 <head>
  <title>Ma première transition</title>
                <meta charset="UTF-8" />
  <link rel="stylesheet" type="text/css" href="style.css" />
 </head>
 <body>
  <p>
                    Passez la souris sur moi !
                </p>
 </body>
</html>
```

Essayer !

L'avantage des transitions, c'est qu'elles ne sont pas bloquantes : si votre visiteur utilise un navigateur ne pouvant pas les interpréter, le changement d'état des propriétés se fera quand même, seulement, il ne sera pas progressif.

Des transitions (un peu) plus complexes

Maintenant que vous avez saisi le principe des transitions CSS, il est temps de voir deux autres propriétés pouvant être utiles :

- transition-delay ;
- transition-timing-function.

La première, **`transition-delay`**, permet comme son nom l'indique de définir un délai avant que la transition n'ait lieu. Comme pour **`transition-duration`**, il est possible de la définir en secondes (**s**) ou en millisecondes (**ms**), et sa valeur par défaut est **0s**.

Reprenons l'exemple précédent et ajoutons-y cette propriété :

Code : CSS

```
/* Style par défaut de nos paragraphes
(= état initial). */
p
{
    /* Seules la couleur du texte et les dimensions
```

```
de nos paragraphes auront une transition. */
    transition-property: color, width, height;

    /* La transition durera trois secondes. */
    transition-duration: 3s;

/* On attend 1,5 seconde avant que
la transition ne se lance. */
transition-delay: 1500ms;

    /* On stylise un peu plus nos paragraphes. */
    width: 400px;
    height: 200px;
    color: #fff;
    background-color: #a5a5a5;
    border: 2px solid #000;
    margin: auto;
    text-align: center;
}
```

Essayer !

Bien plus intéressante, **transition-timing-function** permet de spécifier l'accélération de notre transition : celle-ci doit-elle être rapide au début puis lente à la fin ? Doit-elle s'exécuter à vitesse constante ?
Cette propriété peut prendre les valeurs suivantes :

- **linear** : la vitesse de transition est la même du début à la fin de la transition ;
- **ease** : la transition commence doucement, s'accélère, puis se termine doucement. Il s'agit de la valeur par défaut ;
- **ease-in** : la transition s'accélère peu à peu ;
- **ease-out** : la transition ralentit peu à peu ;
- **ease-in-out** : la transition commence doucement, s'accélère, puis se termine doucement. Il s'agit en fait de la même chose que la valeur **ease**, sauf que l'effet est plus prononcé ;
- **cubic-bezier(x,x,x,x)** : la plus compliquée ! Cette valeur emploie la courbe de Bézier pour définir l'accélération de la transition. Remplacez « simplement » les **x** par des valeurs comprises entre 0 et 1.

Je vous conseille fortement de consulter ce site. Ce dernier vous permet de définir et de choisir plus facilement les valeurs qui correspondent le mieux à vos attentes pour la propriété **transition-timing-function**. Sélectionnez la valeur souhaitée dans la liste déroulante et testez-la directement sur ce site ou bien modifiez la courbe d'accélération pour obtenir les valeurs adéquates sur la courbe de Bézier.

La méga propriété de transition

Comme souvent en CSS, il est possible de fusionner toutes les propriétés que nous venons de voir en une seule, générique. Celle-ci se nomme tout simplement… **transition** !
Voici la manière dont elle se présente :

Code : CSS

```
p
{
    transition: property duration timing-function delay;
}
```

Dans le cas où nous souhaitons faire des transitions pour plusieurs propriétés, il suffit de les ajouter à la suite, chacune séparée par une virgule. Pour notre exemple, voici ce que ça donne :

Code : CSS

```
p
{
    /* Il est possible de spécifier des valeurs différentes pour
chaque propriété. */
    transition: color 3s ease 1500ms, width 1s ease-in 0s, height 2s
ease-out 0.2s;
}
```

Essayer !

Un menu sympa !

Les transitions permettent de donner un côté vraiment très sympathique à nos pages web. En guise d'exercice, je vous propose de réaliser un petit menu dynamique, et ce uniquement en CSS : aucune ligne de JavaScript ne sera nécessaire !

Voici le code HTML de notre menu :

Code : HTML

```
<!DOCTYPE html>
<html>
 <head>
  <title>Un menu dynamique en CSS</title>
  <meta charset="UTF-8" />
  <link rel="stylesheet" type="text/css" href="style.css" />
 </head>
 <body>
  <ul>
   <li id="item1">
    <a href="#lien1">
     <span>
      Description
     </span>
    </a>
   </li>
   <li id="item2">
    <a href="#lien2">
     <span>
      Description
     </span>
    </a>
   </li>
   <li id="item3">
    <a href="#lien3">
     <span>
      Description
     </span>
    </a>
   </li>
   <li id="item4">
    <a href="#lien4">
     <span>
      Description
     </span>
    </a>
   </li>
   <li id="item5">
    <a href="#lien5">
     <span>
      Description
     </span>
    </a>
   </li>
  </ul>
```

```
 </body>
</html>
```

Nous avons donc une liste contenant cinq éléments. Chacun de ces éléments possède un lien, possédant lui-même une description.
Je vous propose d'afficher cette liste de liens sous forme d'images cliquables : au passage de la souris sur une image, la description associée s'affichera dynamiquement sur l'image.

Commençons tout d'abord par définir le style de base de notre liste :

Code : CSS

```
ul
{
 /* On supprime les puces, on précise une largeur
et on centre la liste par rapport à la page. */
        list-style: none;
 width: 60%;
 margin: auto;
}

li
{
 /* On fixe les dimensions
des éléments de notre liste. */
        width: 150px;
 height: 150px;

        /* On met en forme le texte. */
 text-align: center;
 font-weight: bold;

        /* On modifie l'affichage pour que
les éléments soient sur la même ligne. */
 display: inline-block;

        /* On ajoute une petite bordure
et on définit les marges. */
 border: 1px solid #000;
 margin: 10px;
 padding: 0px;

        /* Quitte à faire du CSS3, autant
avoir des coins arrondis :p. */
        border-radius: 10px;

        /* On centre l'arrière-plan. */
        background-position: center;

        /* On modifie la position pour pouvoir ensuite
placer comme on le souhaite la description des liens. */
        position: relative;

        /* On n'affiche pas ce qui dépasse de nos éléments ! */
 overflow: hidden;
}

/* On indique les images à afficher
en arrière-plan pour chaque élément. */
#item1{background-image: url('bgd1.jpg')}
#item2{background-image: url('bgd2.jpg')}
#item3{background-image: url('bgd3.jpg')}
#item4{background-image: url('bgd4.jpg')}
#item5{background-image: url('bgd5.jpg')}
```

Essayer !

Notre menu commence à avoir de l'allure, mais nous avons encore du boulot ! Il serait par exemple mieux de pouvoir cliquer sur toute l'image plutôt que de devoir cliquer sur le texte de description. Pour cela, il suffit d'indiquer dans la feuille de style que le lien doit prendre les mêmes dimensions que son parent.

Code : CSS

```
li a
{
 /* Pour fixer les dimensions d'un lien,
il est nécessaire de l'afficher en tant
qu'élément de type block. */
        display: block;
 width: 100%;
 height: 100%;
        text-decoration: none;
}
```

Voilà qui est mieux.
Il ne nous reste plus qu'à styliser notre description. Je vous rappelle l'objectif : nous ne voulons pas la voir initialement, mais elle doit s'afficher au survol de la souris.
Les descriptions étant contenues dans un ****, il sera nécessaire, comme pour les liens, de les afficher en tant que « **block** » pour fixer leurs dimensions.

Code : CSS

```
li span
{
 display: block;
        height: 50px;
        width: 100%;
        color: #fff;
        margin: 0px;

        /* La taille des lignes est égale à
la hauteur de l'élément. Cela permet
de centrer verticalement le texte. */
        line-height: 50px;

        /* Abusons du CSS3 en affichant
un arrière-plan semi-transparent ! */
        background-color: rgba(0,0,0,0.3);

        /*Mettons les coins inférieurs arrondis
pour que ce soit plus propre
(merci à wibix pour la remarque)*/
        border-bottom-left-radius: 10px;
 border-bottom-right-radius: 10px;

        /* Position absolue pour afficher
la description avec précision. */
        position: absolute;

        /* L'ordonnée de l'élément est égale
à la hauteur de son parent (150px) :
la description sera donc située sous
son parent. */
 left: 0px;
```

```
 top: 150px;
}
```

Voici donc la technique pour rendre « invisible » notre description : on l'affiche sous l'élément qui la contient. Rappelez-vous, nous avons spécifié aux balises **<li>** que leur propriété **overflow** devait avoir pour valeur « **hidden** » : nos descriptions ne s'afficheront pas.
Mais nous avons oublié un point important. Eh oui, nous n'avons pas indiqué la manière dont les transitions doivent réagir !

Code : CSS

```
li span
{
 display: block;
        height: 50px;
        width: 100%;
        color: #fff;
        margin: 0px;

        /* La taille des lignes est égale à
la hauteur de l'élément. Cela permet
de centrer verticalement le texte. */
        line-height: 50px;

        /* Abusons du CSS3 en affichant
un arrière-plan semi-transparent. */
        background-color: rgba(0,0,0,0.3);

        /* Position absolue pour afficher
la description avec précision. */
        position: absolute;

        /* L'ordonnée de l'élément est égale
à la hauteur de son parent (150px) :
la description sera donc située sous
son parent. */
 left: 0px;
 top: 150px;

/* N'oublions pas les transitions ! */
transition-property: top;
transition-duration: 1s;
}
```

Pour afficher notre description, il suffit de jouer avec sa propriété **top** : c'est donc sur elle que nous voulons appliquer nos transitions. J'ai choisi pour ma part une durée d'une seconde, ça me semble bien. Je n'ai pas voulu m'embêter avec des délais et des accélérations : les valeurs par défaut me conviennent très bien.

Il ne nous reste plus qu'à indiquer quand modifier cette propriété **top**. Petite difficulté ici : nous voulons modifier la position de notre description uniquement lorsque la souris survole le parent de cette description.

Code : CSS

```
/* On modifie l'ordonnée des descriptions contenues
dans un élément d'une liste survolée par la souris. */
li:hover span
{
 top: 100px;
}
```

Essayer !

Et voilà !
Nous avons obtenu un menu fort sympathique assez rapidement et sans véritable difficulté puisqu'il n'y a que du code CSS !

Il est cependant possible d'utiliser JavaScript pour jouer avec les transitions CSS, c'est d'ailleurs ce que nous allons voir dès maintenant.

Plus de liberté avec JavaScript

L'intérêt des transitions CSS est de s'affranchir de JavaScript pour dynamiser nos pages Web. Cependant, il peut s'avérer nécessaire d'utiliser ce langage pour effectuer certaines manipulations. Par exemple, comment indiquer au navigateur qu'il doit lancer la transition de tel élément lorsque l'on clique sur tel bouton ? En CSS, ce n'est tout simplement pas possible…

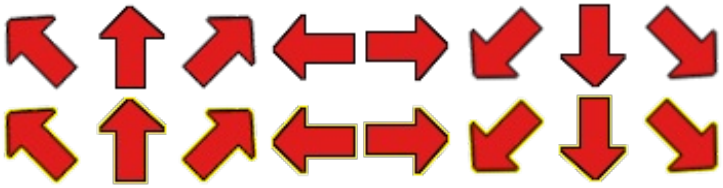
JavaScript va donc nous offrir plus de possibilités avec les transitions, tout ceci avec une simplicité déconcertante !

Vingt mille lieues sous les mers

Pour vous donner un exemple des possibilités qui nous sont offertes, je vous propose de vous prendre pour le capitaine Nemo ! Nous simulerons une sorte de sous-marin pour voir l'océan à travers un hublot…

Pour cela, nous allons avoir besoin de trois images :

Le hublot, petit montage de ma conception



Les flèches pour le tableau de bord, de ma conception également

Et le fond marin (en lien parce que trop grand !).
Venant de Linternaute.com, mais vous l'aviez deviné

Voici donc ce que je vous propose de faire. Notre page contiendra deux ** **. La première aura pour fond notre océan et contiendra une image : le hublot.
La seconde ** ** en contiendra huit autres : une pour chaque flèche de notre tableau de bord.
Ce dernier se comportera de cette façon : quand la souris passera sur une flèche, la vue que nous aurons de l'océan à travers le hublot se déplacera dans la direction indiquée.

Ça peut vous paraître flou pour l'instant, mais je vous assure que c'est très simple.

Les codes HTML et CSS

Vous ne devriez pas avoir de mal à comprendre le code HTML de ce mini projet :

Code : HTML

```
<!DOCTYPE html>
<html>
 <head>
  <title>Vingt mille lieues sous les mers</title>
  <meta charset="UTF-8" />
  <link rel="stylesheet" type="text/css" href="style.css" />
  <script type="text/javascript">
   window.onload=function()
   {
    //On placera notre code JS ici.
   };
  </script>
 </head>
 <body>
  <div id="ocean">
   <img src="hublot.png" alt="Hublot"/>
  </div>
  <div id="controls">
   <div id="fleche_hg"></div>
   <div id="fleche_h"></div>
   <div id="fleche_hd"></div>
   <div id="fleche_g"></div>
   <div id="fleche_d"></div>
   <div id="fleche_bg"></div>
   <div id="fleche_b"></div>
   <div id="fleche_bd"></div>
  </div>
  <h1>Passez la souris sur les flèches !</h1>
 </body>
</html>
```

Tous les éléments de notre page sont mis en place, il ne nous reste qu'à styliser tout ça.
Nous ne nous occuperons du code JavaScript qu'à la fin, une fois que tout sera bien fixé dans notre code CSS.

Commençons donc par traiter ce qu'il y a de plus facile : je propose déjà de colorer le fond de notre page en gris, parce que le blanc est un peu agressif. Centrons également l'instruction « *Passez la souris sur les flèches !* » pour faire plus propre.

Code : CSS

```
body
{
 background-color: #a5a5a5;
}
```

```css
h1
{
 text-align: center;
}
```

Bon, passons aux choses sérieuses avec l'océan à proprement parler.
Nous devons le mettre aux mêmes dimensions que l'image du hublot. Pour que ce soit plus joli, nous pouvons également le centrer et lui ajouter une petite bordure. N'oublions également pas de lui appliquer notre image de fond que nous allons d'ailleurs centrer.
Rappelons-nous le but de l'exercice tout de même : utiliser les transitions. Pour simuler l'exploration des fonds marins, nous modifierons la position de l'arrière-plan, tout simplement !

Code : CSS

```css
#ocean
{
 margin: auto;
 margin-bottom: 10px;
 width: 640px;
 height: 400px;
 border: 2px solid #000;
 background-image: url('background.jpg');
 background-position: center;
        transition-property: background-position;
        transition-duration: 10s;
}
```

Essayer !

Nous avançons bien !
Attaquons-nous maintenant au tableau de bord. Il contiendra les flèches de contrôle, à raison de trois par ligne : chaque flèche étant large et haute de 45 pixels, le tableau de bord doit avoir pour dimensions 135 pixels de haut et de large. Je préfère également mettre sa `position` en « **relative** » afin de placer au pixel près les flèches qu'il contiendra.

Code : CSS

```css
#controls
{
 width: 135px;
 height: 135px;
 margin: auto;
 position: relative;
}
```

Les flèches ont toutes les mêmes dimensions et la même image de fond, ce n'est donc pas trop compliqué :

Code : CSS

```css
#controls div
{
 width: 45px;
 height: 45px;
 position: absolute;
 background-image: url('fleches.png');
 background-repeat: no-repeat;
}
```

Pour les flèches, j'ai fait le choix d'employer la technique des *sprites* CSS : cela ne permet d'avoir qu'une seule image à gérer pour toutes les flèches au lieu d'une vingtaine.
En contrepartie, le code CSS est bien plus pénible à écrire, mais il n'est pas plus complexe pour autant.

Code : CSS

```
#fleche_hg{left: 0px; top: 0px; background-position: 0px top;}
#fleche_hg:hover{background-position: 0px bottom;}

#fleche_h{left: 45px; top: 0px; background-position: -45px top;}
#fleche_h:hover{background-position: -45px bottom;}

#fleche_hd{left: 90px; top: 0px; background-position: -90px top;}
#fleche_hd:hover{background-position: -90px bottom;}

#fleche_g{left: 0px; top: 45px; background-position: -135px top;}
#fleche_g:hover{background-position: -135px bottom;}

#fleche_d{left: 90px; top: 45px; background-position: -180px top;}
#fleche_d:hover{background-position: -180px bottom;}

#fleche_bg{left: 0px; top: 90px; background-position: -225px top;}
#fleche_bg:hover{background-position: -225px bottom;}

#fleche_b{left: 45px; top: 90px; background-position: -270px top;}
#fleche_b:hover{background-position: -270px bottom;}

#fleche_bd{left: 90px; top: 90px; background-position: -315px top;}
#fleche_bd:hover{background-position: -315px bottom;}
```

Essayer !

Tout est fin prêt ! Notre code CSS est en effet achevé, notre navigateur sait, grâce aux transitions, qu'il devra animer le changement de position de l'arrière-plan de l'océan… Il ne nous reste plus qu'à utiliser JavaScript pour ce faire.

La finalisation avec JavaScript

Nous pouvons désormais fermer notre fichier CSS pour revenir à notre fichier HTML et écrire le code JavaScript.
Nous avons déjà préparé notre fichier de façon à ce qu'il accueille le JavaScript correctement : je vous rappelle quand même que tout se passe dans la fonction **window.onload** afin que le code ne s'exécute qu'une fois la page complètement chargée.

Bref, que voulons-nous faire exactement ?
Au passage de la souris sur une flèche, nous souhaitons modifier la propriété **background-position** de l'océan.
Nous devrons donc écrire la fonction **onmouseover** pour chaque flèche.
Pour déplacer le fond vers le coin supérieur gauche, il suffit de spécifier la propriété CSS **background-position** à « **left top** », pour le mettre en bas à droite à « **right bottom** », etc.

Notre code peut donc être le suivant :

Code : JavaScript

```
//On stocke notre océan pour plus de clarté.
var ocean=document.getElementById('ocean');
document.getElementById('fleche_hg').onmouseover=function()
{
 ocean.style.backgroundPosition='left top';
```

```
};
document.getElementById('fleche_h').onmouseover=function()
{
 ocean.style.backgroundPosition='center top';
};
document.getElementById('fleche_hd').onmouseover=function()
{
 ocean.style.backgroundPosition='right top';
};
document.getElementById('fleche_g').onmouseover=function()
{
 ocean.style.backgroundPosition='left center';
};
document.getElementById('fleche_d').onmouseover=function()
{
 ocean.style.backgroundPosition='right center';
};
document.getElementById('fleche_bg').onmouseover=function()
{
 ocean.style.backgroundPosition='left bottom';
};
document.getElementById('fleche_b').onmouseover=function()
{
 ocean.style.backgroundPosition='center bottom';
};
document.getElementById('fleche_bd').onmouseover=function()
{
 ocean.style.backgroundPosition='right bottom';
};
```

Oui, ça fait long comme code, mais avouez qu'on a connu plus difficile en JavaScript.

Il serait également bon de recentrer l'arrière-plan lorsque nous ne sommes plus sur une flèche. Pour cela, il faut définir la méthode **onmouseout** de chaque flèche. Cette fois, cette méthode doit faire la même chose quelle que soit la flèche, une boucle pourra donc nous tirer d'affaire.

Code : JavaScript

```
//On récupère tous les éléments enfants du tableau de bord.
var fleches=document.getElementById('controls').childNodes;

//Pour chaque enfant de notre tableau de bord...
for(var i in fleches)
{
 /* Malheureusement, « fleches » ne contient pas que nos flèches,
il contient aussi des éléments « Text » : il nous faut donc vérifier
que
l'enfant parcouru possède l'attribut « style ». Si oui, il s'agit
d'une
de nos flèches, on peut donc continuer. */
        if(fleches[i].style)
 {
  //On peut donc définir notre fonction.
                fleches[i].onmouseout=function()
  {
              //Fonction qui recentre l'arrière-plan de l'océan.
                    ocean.style.backgroundPosition='center';
  }
 }
}
```

Essayer !

Malheureusement, cet exemple ne fonctionne pas toujours avec le navigateur Opera et pour une raison bien étrange : ce dernier ne prend pas encore correctement en charge les transitions de la propriété **background-position**... Gageons que cette erreur soit corrigée lors de la prochaine mise à jour du navigateur ! En attendant, il est toujours possible de ruser en changeant la façon de penser le système : on peut par exemple ajouter une balise ** ** contenant notre fond, la placer en position « absolue » et la déplacer en jouant avec ses propriétés **top** et **left**.

Notre JavaScript modifie donc la position de l'arrière-plan selon les flèches survolées. Grâce à notre code CSS, le changement se fera progressivement, donnant ainsi l'illusion de mouvement ! Notez bien qu'aucune transition ne se fait dans le code JavaScript, tout se déroule dans le code CSS.

N'est-ce pas merveilleux ? 

Allons plus loin avec *transitionend*

Détectons la fin de notre transition

Vous n'êtes pas sans savoir que JavaScript permet de lancer des fonctions lors de l'exécution d'événements. Ainsi, dans notre expérience sous-marine, nous bougions l'arrière-plan lors de l'événement « passage de la souris sur les flèches du tableau de bord ».

Les transitions CSS ont fait naître un nouvel événement : **transitionend**. Comme les anglophones parmi vous l'auront deviné, cet événement se produit lorsque la transition s'est achevée.

Sur le papier, ça peut sembler pratique, notamment pour créer un système d'animations enchaînées. Malheureusement, tout comme la propriété CSS, l'événement JavaScript change de nom selon le navigateur du visiteur...

Ainsi, si Firefox utilise bien **transitionend**, Opera préfère **OTransitionEnd** tandis que Google Chrome a opté pour **webkitTransitionEnd**. Bien entendu, Internet Explorer s'en contre-fiche, vu qu'il ne se préoccupe déjà pas des transitions en CSS. 

Ce dernier point est un véritable problème. Si le navigateur de votre visiteur n'interprète pas les transitions CSS, l'événement **transitionend** ne se déclenchera jamais ! Évitez donc d'y utiliser des fonctions indispensables à la bonne navigation de votre site.

Tout ceci ne simplifie pas nos affaires et je vous propose donc un petit exemple pour comprendre le fonctionnement.

Code : HTML

```
<!DOCTYPE html>
<html>
 <head>
  <title>Découvrons transitionend</title>
  <meta charset="UTF-8" />
  <link rel="stylesheet" type="text/css" href="style.css" />
  <script>
   window.onload=function()
   {
    // Nombre de fois que l'animation a été achevée
    var total=0;

    // On stocke les noms des événements selon les navigateurs
    var
navigatorsProperties=['transitionend','OTransitionEnd','webkitTransitionEnd'];

    // On parcourt chacun de ces événements
    for(var i in navigatorsProperties)
    {
     // Et on les lie à notre élément HTML
```

```
document.getElementById('boite').addEventListener(navigatorsProperties[i],functi
      {
       total++;
       document.getElementById('infos').innerHTML='Transition terminée '+total+'
!';
      },false);
     }
    };
  </script>
 </head>
 <body>
  <h1 id="infos">Passez la souris sur le carré !</h1>
  <div id="boite"></div>
 </body>
</html>
```

Essayer !

Bon, je ne vous montre pas le code CSS, j'y modifie juste la propriété **margin-left** de ma ** ** au passage de la souris, et ce avec une **transition-duration** de 2 secondes. Vous devriez être capables de le faire tout seul, maintenant.

Si vous avez testé le code, vous avez vu que l'événement **transitionend** ne se déclenche que lorsque la transition a atteint son état initial ou son état final. Ainsi, si vous sortez la souris du carré pendant son déplacement, celui-ci revient vers son point de départ : la transition vers son état final est certes terminée mais aucun événement ne s'est déclenché ! En effet, le carré fait demi-tour vers son état initial, il considère donc que sa transition n'est pas achevée !

Quelle propriété a été animée ? Combien de temps a duré la transition ?

JavaScript nous offre la possibilité de connaître la propriété CSS qui a été animée. Pour cela, il suffit de récupérer le paramètre fourni par l'événement **transitionend**. Ce paramètre contient en effet deux propriétés qui peuvent nous être utiles :

- *propertyName* contient le nom de la propriété CSS qui vient d'être animée ;
- *elapsedTime* indique le temps écoulé depuis le début de la transition, en secondes.

Code : JavaScript

```
window.onload=function()
{
 var navigatorsProperties=['transitionend','OTransitionEnd','webkitTransitionEnc

 for(var i in navigatorsProperties)
 {

document.getElementById('boite').addEventListener(navigatorsProperties[i],functi
  {
   document.getElementById('infos').innerHTML='Transition de la propriété
<em>'+e.propertyName+'</em> terminée en <b>'+e.elapsedTime+'s</b> !';
  },false);
 }
};
```

Essayer !

Il y a toutefois une particularité pour *elapsedTime* : si la transition se coupe en plein milieu, il est remis à zéro. Essayez avec l'exemple ci-dessus en sortant la souris du carré pendant son déplacement, vous verrez que le temps indiqué à la fin de l'animation est toujours inférieur à deux secondes contrairement à ce qui a été indiqué dans le CSS !

Maintenant que tout est clair sur l'événement **transitionend**, je vous propose un petit exemple d'utilisation.

La balade de Carapuce

Pour cet exercice, je vous propose de déplacer une ** ** dans une autre. Rien de bien palpitant me direz-vous, mais cela peut donner des résultats intéressants, vous verrez.

Tout d'abord, voici la structure de notre page HTML :

Code : HTML

```
<!DOCTYPE html>
<html>
 <head>
  <title>Une div se promène.</title>
  <meta charset="UTF-8" />
  <link rel="stylesheet" type="text/css" href="style.css" />
  <script type="text/javascript">
   window.onload=function()
   {
      // Plus tard =D
   };
  </script>
 </head>
 <body>
  <div id="conteneur">
   <div id="moveDiv"></div>
  </div>
 </body>
</html>
```

Et son CSS correspondant, que vous pouvez bien entendu améliorer (il est franchement très basique et très moche).

Code : CSS

```
#conteneur
{
 width: 155px;
 height: 130px;
 position: relative;
 margin: auto;
 background-color: #000;
}

#moveDiv
{
 width: 23px;
 height: 19px;
 position: absolute;
 top: 10px;
 left: 10px;
 background-color: #fff;
 transition: left 4s linear, top 5s linear;
}
```

Comme vous le voyez, j'ai décidé d'animer les propriétés **left** et **top** : le déplacement de notre ** ** sera donc fluide.

Il ne nous reste plus qu'à lancer la transition en JavaScript pour qu'elle se fasse dès le chargement de la page, sans aucune action de la part du visiteur. Ensuite, via **transitionend**, on demandera un autre déplacement de sorte que notre élément visite chacun des quatre coins de son parent.

Code : JavaScript

```
window.onload=function()
{
 // Les coordonnées de chaque coin du conteneur
 var topLeft=[10,10];
 var topRight=[122,10];
 var bottomLeft=[10,101];
 var bottomRight=[122,101];

 // On crée un objet littéral pour stocker nos valeurs
 var objet=
 {
  // Notre div à déplacer
  'div': document.getElementById('moveDiv'),

  // La position vers laquelle elle se dirige
  'position': topRight,

  // Et une fonction pour la faire bouger !
  'move': function()
  {
   this.div.style.left=this.position[0]+'px';
   this.div.style.top=this.position[1]+'px';
  }
 }

 var
navigatorsProperties=['transitionend','OTransitionEnd','webkitTransitionEnd'];
 for(var i in navigatorsProperties)
 {
  objet.div.addEventListener(navigatorsProperties[i],function(e)
  {
   /*Lorsque la div a fini son déplacement, on lui
indique un autre point à atteindre*/
   switch(objet.position)
   {
    case topRight:
     objet.position=bottomRight;
    break;

    case bottomRight:
     objet.position=bottomLeft;
    break;

    case bottomLeft:
     objet.position=topLeft;
    break;

    case topLeft:
     objet.position=topRight;
    break;
   }

   // Et on lui demande de se rendre à ce point !
   objet.move();
  },false);
 }

 // On lance le déplacement de notre div !
```

```
        objet.move();
    };
```

Essayer !

Notre ** ** parcourt bien chacun des quatre points de son conteneur. Bon, ce n'est pas très joli, mais en appliquant des images en arrière-plan, il est possible d'avoir des résultats bien plus sympathiques. Tenez, prenons ces images :

Trouvée sur Spriters Ressource.

Trouvée également sur Spriters Ressource et ré-arrangée par moi-même .

Essayer !

N'est-ce pas mieux ? Si vous n'êtes pas encore convaincu par les transitions CSS3, je ne peux plus rien pour vous !

Pour modifier l'orientation du Carapuce, il suffit de changer sa propriété **background-position** dans notre code JavaScript, au niveau du **switch**.

*Merci à Nesquik69 pour m'avoir suggéré la rédaction de cette partie sur **transitionend**.*

Q.C.M.

Le premier QCM de ce cours vous est offert en libre accès.
Pour accéder aux suivants

Connectez-vous Inscrivez-vous

Quelle propriété CSS3 permet de lier des propriétés aux transitions ?

- ◯ transition-property.
- ◯ transition-attribut.
- ◯ transition-properties.
- ◯ transition-attributes.

Que se passe-t-il si je passe ma souris sur une balise ** ** en ayant ces instructions CSS ?

Code : CSS

```
div
{
    width: 300px;
    height: 200px;
    background-color: #fff;
```

```
        transition-property: background-color;
    }

    div:hover
    {
        width: 400px;
        height: 100px;
        background-color: #000;
    }
```

- ◯ Toutes les propriétés de la balise se modifieront via une transition, excepté la propriété **background-color** qui sera modifiée directement.
- ◯ Toutes les propriétés de la balise se modifieront via une transition.
- ◯ Seule la propriété **background-color** se modifiera via une transition, toutes les autres se modifieront directement.
- ◯ Toutes les propriétés de la balise se modifieront directement.

Parmi les souhaits suivants, lequel nécessite obligatoirement JavaScript pour être réalisé ?

- ◯ Je souhaite effacer progressivement une image pour faire apparaître celle en dessous lorsque son élément parent est survolé par la souris.
- ◯ Je souhaite que la couleur de fond de mon paragraphe devienne bleue petit à petit lorsque je clique sur un bouton.
- ◯ Je souhaite modifier la taille de mon textarea de façon fluide quand l'utilisateur souhaite écrire du texte dedans.

J'utilise un navigateur n'interprétant pas les transitions CSS. Parmi les affirmations suivantes, laquelle est fausse ?

- ◯ Les propriétés CSS peuvent quand même être modifiées dynamiquement, seulement cela ne se fera pas de façon fluide.
- ◯ Les fonctions liées à l'événement JavaScript **transitionend** s'exécuteront quand même, comme si les transitions avaient lieu.
- ◯ Si mon navigateur n'interprète pas les transitions, c'est peut-être parce que le développeur a oublié de mettre les préfixes -moz, -webkit ou -o à ses propriétés.

Correction !

Statistiques de réponses au QCM

Les animations avec CSS3 sont donc une sacrée avancée dans le domaine du développement web. JavaScript n'est désormais plus relégué qu'à titre de support pour ces transitions, fini le temps des machines à gaz écrites avec ce langage !

Mais qu'on ne se leurre pas : même si nous pouvons faire des choses vraiment sympathiques rien qu'en CSS3, JavaScript reste indispensable pour des choses un peu plus avancées.

Il est temps de se demander si charger une bibliothèque telle que **jQuery** est indispensable pour nos petites animations.

Partager

Ce tutoriel a été corrigé par les zCorrecteurs.