Les transformations

- Utiliser les transformations
- Les transitions
- les animations : les animations, les transitions.

Les transformations

Utiliser les transformations

CSS3 apporte les transformations en 2 dimensions à travers la propriété transform et une liste de fonctions prédéfinies.

La syntaxe est simple d'emploi.

```
transform: function(value);
```

Il est également possible d'effectuer des transformations combinées en espaçant les fonctions d'un simple caractère blanc.

```
transform : function1(value1) function2(value2)
function3(value3);
```

La propriété CSS transform permet de manipuler un élément HTML sur les axes X et Y (horizontal et vertical) grâce à des fonctions diverses de transformation graphique. Il est donc possible de modifier l'apparence d'un élément grâce à un ensemble fonctions 2D :

- Translation (translate),
- Mise à l'échelle (scale),
- Rotation (rotate)
- Inclinaison (skew)

Les transformations

Utiliser les transformations

Pour pouvoir appliquer des transformations, nous avons besoin de savoir quel est le point d'origine (d'ancrage) de la transformation. La propriété transform-origin définit ce point d'origine.

La valeur initiale de cette propriété est le centre de l'élément, ce qui équivaut à la notation :

```
transform-origin: 50% 50%;
```

Il est possible de changer cette valeur en utilisant un mot-clef de position (top, right, bottom, left) suivi d'une valeur chiffrée dont l'unité peut varier (px, %, etc.)

```
div {
    transform-origin: top 0 left 0;
    transform: scale(1.25);
}
```

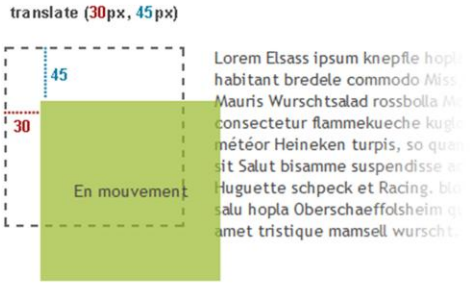
Il s'agit là de la syntaxe proposée par le W3C. À l'heure actuelle (2012) aucun navigateur n'implémente cette syntaxe correctement. Cependant, il suffit de supprimer les mots-clefs de position pour obtenir des résultats sur tous les navigateurs récents (toujours à condition d'utiliser les préfixes vendeurs -webkit-,-moz-,-ms-,-o- selon les versions).

Les transformations

Utiliser les transformations

Une fois l'origine choisie, nous pouvons affecter des transformations à nos éléments avec la propriété transform.

La fonction translate permet d'effectuer une translation (un déplacement) de l'élément sur les axes X et Y.



Il n'y a ici aucune notion de flux, l'élément part de son emplacement courant, quel que soit le type de positionnement que vous lui aurez attribué.

```
transform: translate(x,y);
```

y est une **valeur optionnelle** équivalente à 0 si elle n'est pas renseignée. Les deux valeurs peuvent être négatives.

Ces fonctions permettent de réaliser une translation sur l'axe X (translateX) ou Y (translateY).

```
transform: translateX(value) translateY(value);
```

Les transformations

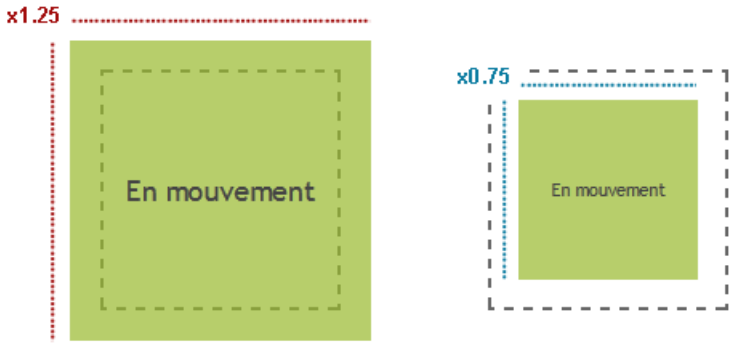
Utiliser les transformations

La fonction scale permet d'agir sur l'échelle (les dimensions) de l'élément. La valeur initiale est 1, tandis que les valeurs supérieures à 1 créent un effet d'agrandissement, et les valeurs inférieures créent un effet de réduction.

```
transform: scale(x,y);
```

La valeur y est optionnelle et **sera égale à la valeur de x si elle est non renseignée**, par exemple pour un agrandissement d'un facteur 1.25 :

```
transform: scale(1.25);
```



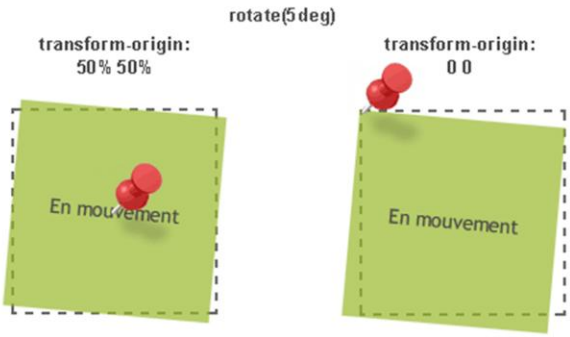
Les valeurs de x et y peuvent être aussi négatives. Il est possible d'effectuer ce "zoom" sur tous les éléments a priori... mais ce n'est pas parce que vous pouvez le faire qu'il faut le faire. N'oubliez pas qu'une image agrandie pourra être floue ou mal interpolée selon sa résolution initiale et celle de l'écran de destination.

Sur le même principe que pour les fonctions dérivées de translate, ces deux fonctions permettent de définir indépendamment les valeurs x et y.

Les transformations

Utiliser les transformations

Il s'agit d'une des plus simples fonctions (rotate) à comprendre. Comme son nom l'indique, elle permet d'effectuer une **rotation** de l'élément ciblé.



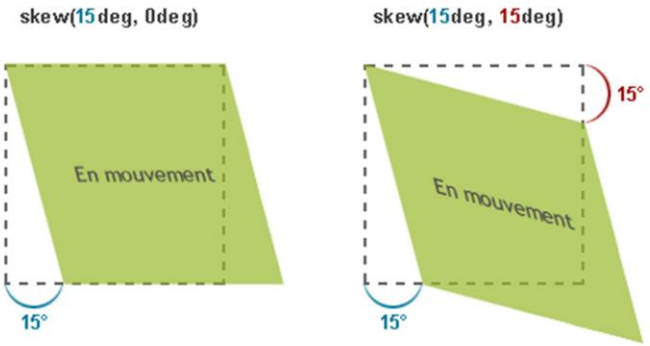
Cette rotation s'exprime en **degrés** (unité deg). et peut être négative et supérieure de manière absolue à 360. Ce dernier point n'a de réel intérêt que lors d'une animation d'un état à un autre afin de présenter, par exemple, une rotation de plusieurs tours d'un élément. Autrement, sans animation de la rotation, la valeur 380° équivaut visuellement à une rotation de 20°.

```
transform: rotate(5deg);
```

Les transformations

Utiliser les transformations

La fonction skew permet d'obliquer la forme d'un élément. À ma grande surprise, la documentation du W3C ne parle que des fonctions skewX et skewY, et pour cause :



Il s'agit des fonctions dérivées de skew. Voici deux exemples de transformation en utilisant les deux fonctions. Vous aurez compris la syntaxe de base :

```
transform: skewX(15deg);
```

...ou en changeant les valeurs de X et Y :

```
transform: skewX(15deg) skewY(15deg);
```

Le principe de base d'une **transition CSS3** est de permettre une transition douce entre l'ancienne valeur et la nouvelle valeur d'une propriété CSS lorsqu'un événement est déclenché :

- soit via une pseudo-classe telles que :hover, :focus ou :active
- soit via JavaScript

Précédemment, ce genre de comportement n'était possible qu'avec l'usage de JavaScript. Ce nouveau module CSS3 permet dorénavant de s'en affranchir au profit exclusif des feuilles de style.

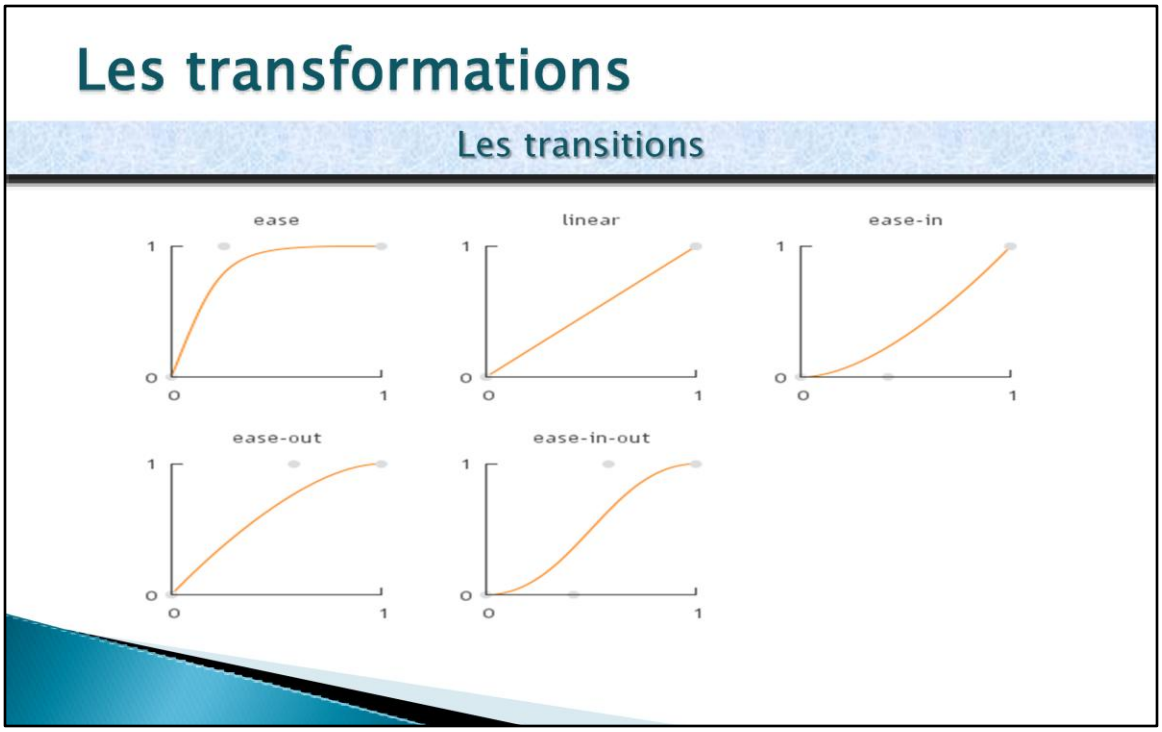
Pour définir une nouvelle transition animée, il est nécessaire de préciser au minimum :

- La ou les propriété(s) à animer
- La durée de l'animation

Les deux propriétés minimales nécessaires pour rendre fonctionnelle une transition en CSS 3 sont transition-property et transition-duration.
Il existe d'autres propriétés CSS spécifiques aux transitions : transition-timing-function, transition-delay et la notation raccourcie transition, que nous allons voir en détails.

Transition-property
Elle permet de définir la propriété qui sera animée lors de la transition. Les valeurs possibles sont :
❖All : toutes les propriétés animables
❖None : aucune transition.
❖Une ou plusieurs propriétés animables séparées par une virgule.
Transition-property: color, width;

Transition-duration
Cette propriété définie tous simplement la durée totale de la transition exprime en secondes. La valeur par défaut est zéro seconde ce qui ne provoque aucune transition lors du changement d'état. Voici la syntaxe pour une transition d'une durée de trois secondes.
Transition-duration: 3s;



Transition-timing-function
Cette propriété définit la progression ou l'évolution de la vitesse de transition. Une fonction mathématique va permettre le calcul des différentes valeurs interpolées lors de la transition.

Les fonctions prédéfinies sont les suivantes :
ease : Rapide sur le début et ralenti sur la fin.
linear : La vitesse est constante sur toute la durée de l'animation.
ease-in : Lent sur le début et accélère de plus en plus vers la fin.
ease-out : Rapide sur le début et décèlere sur la fin.
ease-in-out : Le départ et la fin sont lents.

Les transformations

Les transitions

Il reste une propriété simple relative aux transitions, il s'agit de la propriété transition-delay. Cette propriété permet un départ retardé de la transition.

```
div{
     -webkit-transition-delay: 500ms;
        -moz-transition-delay: 500ms;
         -ms-transition-delay: 500ms;
          -o-transition-delay: 500ms;
             transition-delay: 500ms;
}
```

La propriété transition-delay permet un départ retardé de la transition. Si la valeur précisée positive, alors le départ de la transition sera décalé d'autant. Si cette valeur est négative, l'animation de la transition débute immédiatement mais depuis l'État interpolé correspondant au delta de temps indiqué en valeur négative. La durée totale de l'animation est diminuée de cette valeur.

Transition

Comme pour beaucoup de propriétés composées en CSS, il existe un raccourci qui permet de renseigner la plupart des valeurs avec une seule instruction. Il faut donc saisir les différentes valeurs séparées par un espace.

transition: width 2s ease;

Les animations CSS3 sont semblables aux transitions. La majeure différence entre les deux est que **les animations permettront un contrôle très précis dans le temps** de la valeur que prendront les différentes propriétés CSS.

Une animation peut être lancée lors d'un changement de pseudo-classe (:hover, :focus, :active, :target, :checked, etc), lors d'un changement de class via Javascript, ou simplement lors du chargement de la page.

Si l'animation n'est répétée qu'une fois par défaut, elle peut être itérée à souhait.
Pour réaliser une animation CSS, nous avons besoin de deux éléments distincts: une déclaration d'images-clés (keyframe) définie au sein d'une @-rule, et un lien d'un sélecteur vers cette animation au sein d'un bloc de code CSS.

Les animations étant encore à l'état d'ébauche, il vous faudra évidemment ajouter les préfixes navigateurs (présentement, Webkit -webkit-, Gecko -moz- et Opera -o- supportent les animations. Il faudra également pensé à IE 10 -ms-).

Ainsi, nous devrons écrire @-moz-keyframes et -moz-animation afin que nos animations soient comprises sur Firefox. Le même principe s'appliquera aux autres navigateurs.

Les transformations

Les animations

@keyframes va permettre de définir les différentes étapes de l'animation.

Voici l'exemple d'une règle définissant les étapes d'une transition classique :

```
@keyframes hide{
        0%{
            opacity: 1;
        }
        100%{
            opacity: 0;
        }
}
```

Elle est pour l'instant encore préfixé pour Webkit, Gecko et IE10 et son implémentation est assez bonne dans les navigateurs concernés. Les étapes de l'animation vont être identifiées par un pourcentage ou l'un des mots-clés from qui équivaut à 0 % et to qui équivaut à 100 %. Pour chaque étape, comme lors d'une transition simple, nous allons définir les règles qui subissent une modification.

Il ne reste ensuite à donner un nom à cette @keyframes pour pouvoir l'associer avec une animation.

Les transformations

Les animations

Maintenant que nos keyframes sont créés et référencées par le nom qui leur a été donné, il ne nous reste plus qu'à appeler cette animation à partir d'un sélecteur. Les différentes propriétés d'animation citées ci-dessous :

Lancer les animations

❖animation-name: Où on indique le nom de l'animation à utiliser. (obligatoire)

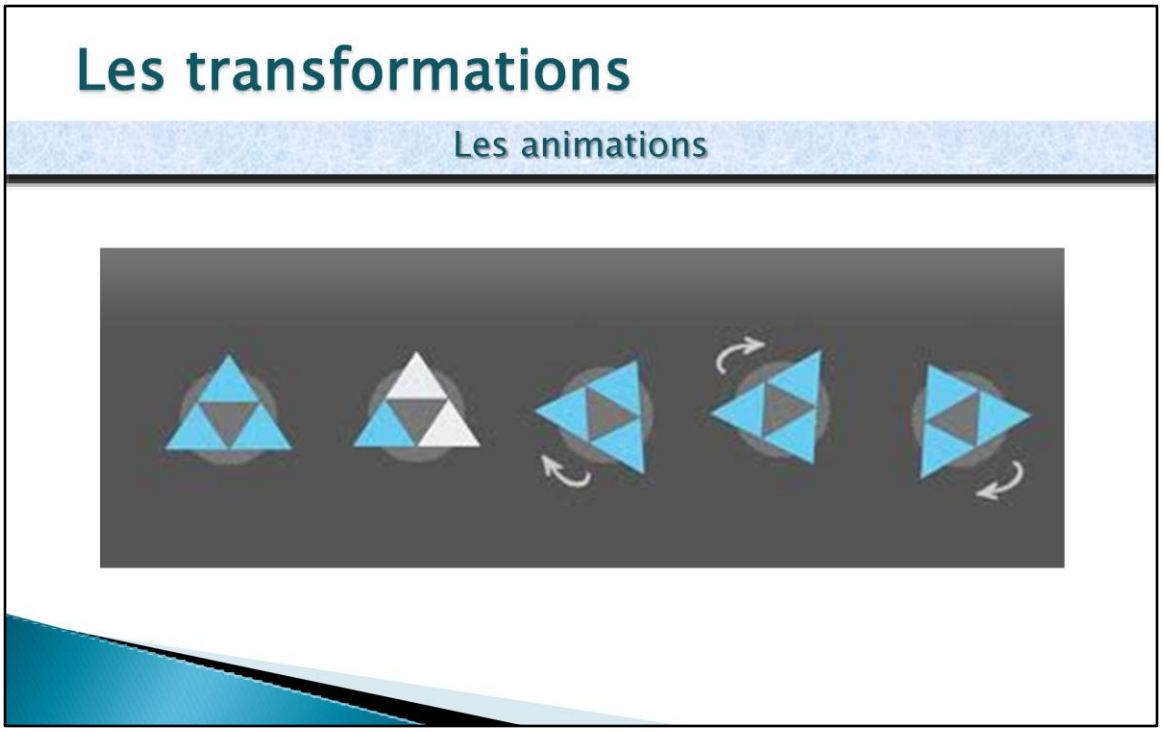
❖animation-duration: Où on indique la durée en milliseconde (ms) ou en seconde (s) de l'animation. (Obligatoire)

❖animation-iteration-count: Le nombre de fois que l'animation doit être effectué. La valeur devra être un nombre entier ou le mot clé infinite pour que l'animation se répète à l'infinie. La valeur par défaut est de 1. (Facultative)

❖animation-direction: Définira si l'animation doit toujours jouer du début à la fin, ou si une fois rendu à la fin, elle doit refaire l'animation en sens inverse. Par défaut l'animation recommencera du début (normal), mais l'utilisation du mot clé alternate alternera la direction de l'animation. (Facultative)

❖animation-timing-function: Définit un effet de timing à utiliser (le modèle d'interpolation) entre chaque keyframe. Comme vu précédemment, cette propriété est également modifiable au sein même d'une keyframe. (Facultative)

❖animation-delay: Cette valeur définira un délai d'attente avant le début de l'animation, ou dans le cas d'une valeur négative, l'avance que doit prendre l'animation avant de débuter. (Facultative)



Lancer les animations (suite)

❖animation-fill-mode: Cette propriété définit l'état de départ et de fin de votre animation. Voici les différentes valeurs possibles:

- forwards: indique au navigateur de laisser l'élément dans sont état final lors de la dernière itération. (l'élément ne revient donc pas à son état initial)
- backwards: indique au navigateur de placer l'élément dans son état définit au keyframe 0% au chargement de la page, même si un délais négatif est indiqué.
- both: appliquera les deux valeurs précédentes.
- none: indiquera au navigateur de styler l'élément selon son état à la première keyframe visible (dans le cas d'un délais négatif) et de ramener l'animation à la keyframe 0% après la dernière itération. Ceci est le comportement par défaut.

Les transformations

Les animations

Évidemment, autant de valeurs (avec les préfixes navigateurs en plus) vous créeront vite 21 lignes supplémentaires dans votre bloc de code. Ainsi, nous utiliserons plus souvent qu'autrement la syntaxe simplifiée:

```
selecteur {
  animation: <name> <duration> <timing-function> <delay>
<iteration-count> <direction> <fill-mode>;
}
```

Ou si vous désirez appliquer plusieurs animations sur un élément:

```
selecteur {
  animation:  <name> <duration> <timing-function> <delay>
<iteration-count> <direction> <fill-mode>,
  <name> <duration> <timing-function> <delay> <iteration-
count> <direction> <fill-mode>;
}
```

Les transformations

Les animations

Mettre l'animation sur pause

Et pour clore ce chapitre sur les animations CSS3, il vous est également possible de mettre en pause une animation via la propriété animation-play-state qui prendra l'une des deux valeurs suivantes : running (par défaut) ou paused pour arrêter l'animation.

```
selecteur:hover {
  -webkit-animation-play-state:paused;
  -moz-animation-play-state:paused;
  animation-play-state:paused;
}
```