CSS3 : Transformations 2D

CSS3 apporte les transformations en 2 dimensions à travers la propriété `transform` et une liste de fonctions prédéfinies. Voyons ensemble la prise en charge actuelle de cette propriété et des fonctions qui l'accompagnent.

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 (translate),
- Mise à l'échelle » #scale (scale),
- Rotation » #rotate (rotate)
- Inclinaison » #skew (skew)
- Matrice de transformation » #matrix (matrix)

Syntaxe

La syntaxe est simple d'emploi.

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

Il est souvent nécessaire d'utiliser les préfixes vendeurs (`-webkit-`, `-moz-`, etc.) devant la propriété `transform` pour utiliser les transformations 2D sur les versions actuelles et passées de navigateurs. Reportez-vous aux tableaux de compatibilité pour savoir ce qu'il en est des moteurs, et à partir de quelles versions.

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 `transform-origin`

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.

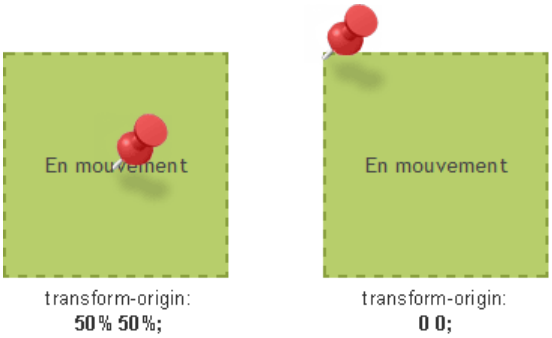
La valeur initiale de cette propriété est le centre de l'élément, ce qui équivaut à la notation :

```
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);



Quelques exemples de positionnements :

Point d'origine en haut à gauche

```
transform-origin: 0 0;
```

Point d'origine en bas à droite

```
transform-origin: 100% 100%;
```

Point d'origine en bas et centré

```
transform-origin: 50% 100%:
```

Voici quelques exemples en démonstration de transformations avec la fonction `scale` (cf. le détail plus loin) dont le point d'origine varie.

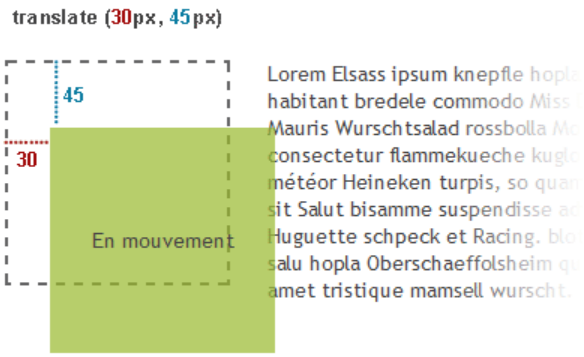
Démonstration » /xmedia/tuto/css3/transformations-2d/transform-origin.html

Les fonctions de la propriété `transform`

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

La fonction `translate`

Elle 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.

Démonstration » /xmedia/tuto/css3/transformations-2d/translate.html

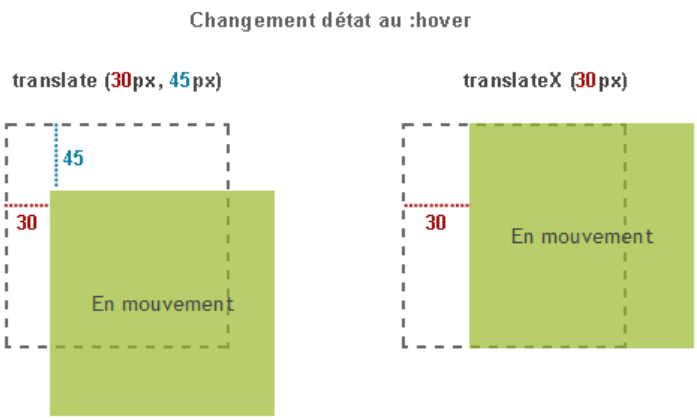
Les fonctions `translateX` et `translateY`

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

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

Si vous pensiez comme moi que ces fonctions permettent une modification de l'une des deux valeurs du vecteur initial, dans le cas par exemple où la transformation vient en écraser une autre, alors vous faites erreur. Essayez le code suivant pour comprendre :

```
div {
   transform: translate(20px, 35px);
}
div:hover {
   /* redéfinition de la valeur X ? */
   transform: translateX(20px);
}
```



En définissant uniquement `translateX` sur l'évènement `:hover`, `translateY` est implicitement **redéfini** avec la valeur `0`.

Démonstration » /xmedia/tuto/css3/transformations-2d/translatex-translatey.html

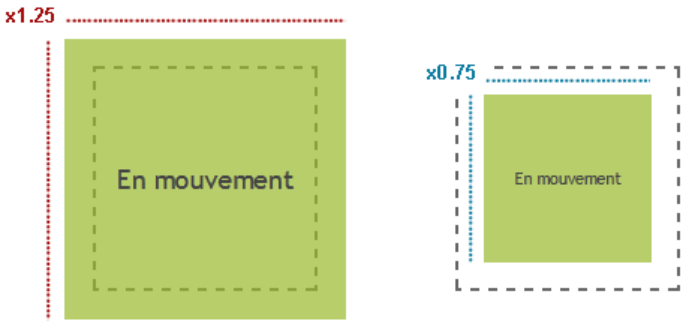
La fonction `scale`

Cette fonction 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);
```

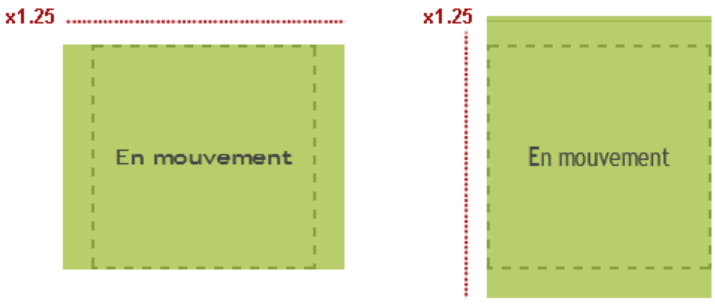


Les valeurs de `x` et `y` peuvent être aussi négatives. Il est possible d'effectuer ce "zoom" sur tous les éléments a priori... mais ce n'est pas parce que vous pouvez le faire qu'il faut le faire. N'oubliez pas qu'une image agrandie pourra être floue ou mal interpolée selon sa résolution initiale et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

Démonstration » /xmedia/tuto/css3/transformations-2d/scale.html

Les fonctions `scaleX` et `scaleY`

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



Si `scaleX` est uniquement renseigné, `scaleY` vaudra implicitement 1 (aucun effet de `scale` en `y` donc), et inversement.

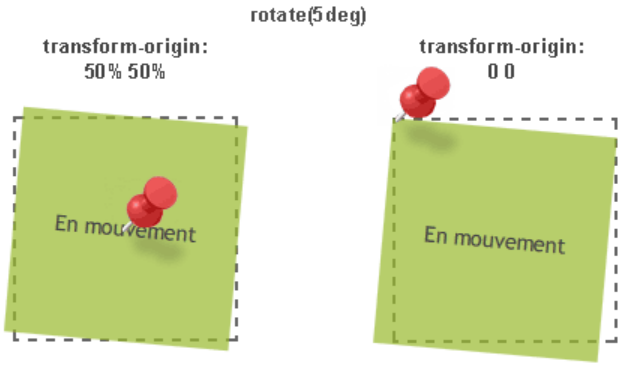
```
transform: scaleY(1.25);
```

Cette transformation va élargir l'élément ainsi que son contenu éventuel. Il suffit de regarder la démonstration suivante pour constater la transformation (prêtez attention au texte de l'encadré vert).

Démonstration » /xmedia/tuto/css3/transformations-2d/scalex-scaley.html

La fonction `rotate`

Il s'agit d'une des plus simples fonctions 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);
```

Attention cependant, le point d'origine de la transformation a son importance comme le présente la démonstration qui suit.

Démonstration » /xmedia/tuto/css3/transformations-2d/rotate.html

La fonction `skew`

Cette fonction 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 :

> Important, transform: skew() is not supported anymore (spec and gecko). (But skewX an skewY are).
>
> — Paul Rouget(@paulrouget) Avril 12, 2012 »
> https://twitter.com/paulrouget/status/190455471470161920

Attention : les navigateurs qui prennent en charge la fonction `skew()` font du zèle. Cette fonction n'est pas prévue par le W3C, et ne sera plus supportée (prochainement par Gecko), dans un avenir proche. Pour faire les choses correctement, utilisez les fonctions `skewX` et `skewY`.

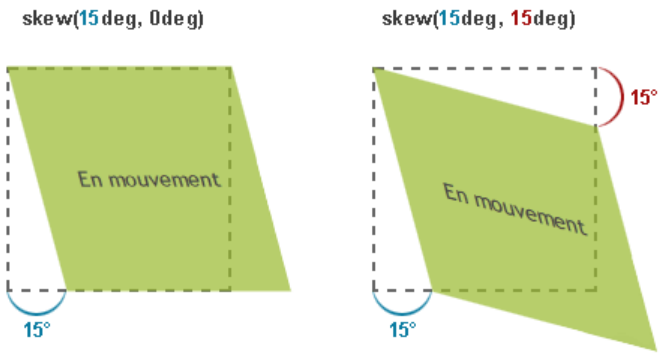
Les fonction `skewX` et `skewY`

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);
```



Démonstration » /xmedia/tuto/css3/transformations-2d/skewx-skewy.html

La fonction "absolue" `matrix`

Cette fonction permet de réunir en une seule déclaration toutes les fonctions précédentes que nous avons vu ensemble sous la forme d'une matrice. Très concrètement, les détails ne seront pas abordés ici car ils font beaucoup plus appel à des notions mathématiques complexes que beaucoup d'entre nous ont mis de côté.

Exemple d'une transformation avec beaucoup de fonctions :

```
div {
    transform-origin: 0 0;
    transform: rotate(15deg) translateX(230px)  scale(1.5, 2.6) skew(220deg, -150deg)
translateX(230px);
}
```

Équivaut à la matrice suivante :

```
div {
    transform-origin: 0 0;
    transform: matrix(1.06, 1.84, 0.54, 2.8, 466px, 482px);
}
```

Pour comprendre comment fonctionne tout ceci, lisez l'excellent article de Useragentman.com : The CSS3 matrix() Transform for the Mathematically Challenged » http://www.useragentman.com/blog/2011/01/07/css3-matrix-transform-for-the-mathematically-challenged/ . Sachez qu'il existe un éditeur en ligne qui vous permet de manipuler une boîte dans le but d'obtenir des coordonnées matricielles : www.useragentman.com/matrix/ » http://www.useragentman.com/matrix/

Ordre des déclarations

La propriété `transform` accepte plusieurs fonctions les unes à la suite des autres, comme proposé précédemment. Cependant, l'**ordre des fonctions a son importance**.
En effet, les deux transformations suivantes n'ont pas le même résultat :

```
div {
    transform: scale(2) translate(20px, 20px);
```

```
}
```

et différent de :

```
div {
    transform: translate(20px, 20px) scale(2) ;
}
```

De la même manière :

```
div {
    transform: translate(40px, 40px) scale(2) ;
}
```

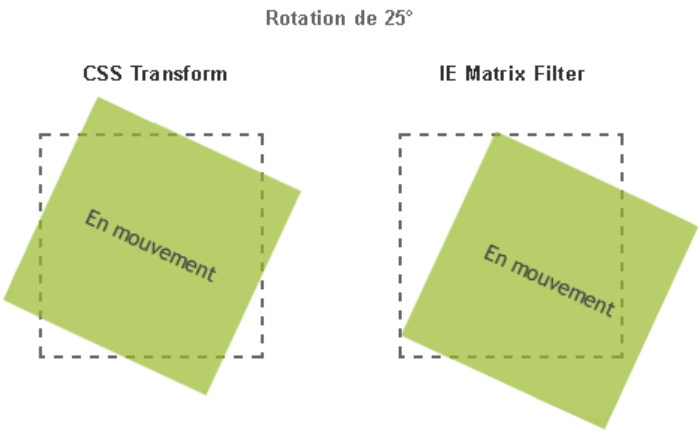
propose le même résultat que :

```
div {
    transform: scale(2) translate(20px, 20px);
}
```

En effet, les transformations se font dans l'ordre où elles sont déclarées, ainsi une translation de `20px` est équivalente à `40px` si un `scale` de `2` la précède. Vous me suivez ?
Attention donc à l'ordre de déclaration de ces fonctions.

Du `transform` sur tous les navigateurs

Internet Explorer possède son propre filtre pour effectuer des transformations. En plus du filtre propriétaire "Matrix" » http://msdn.microsoft.com/en-us/library/ms533014%28VS.85%29.aspx , le comportement des transformations est différent puisque IE va tenter de transformer l'élément en restant collé aux bords haut et gauche de la boîte originelle (représentée par le cadre en pointillés dans les démo de cet article).



L'outil IE TransformsTranslator » http://www.useragentman.com/IETransformsTranslator/ permet de corriger le tir en appliquant automatiquement des marges aux éléments transformés pour IE.

Tableau des compatibilités

Navigateurs	Versions	Détails
		Avant la version 9, il est possible d'utiliser les filtres propriétaires.

	**Internet Explorer 9+**	A partie de la 10 : sans préfixe -ms-.
	**Firefox 3.5+**	à partir de 16.0 : sans préfixe -moz-
	**Chrome 4+** **Chrome Mobile (Android 4)**	avec préfixe -webkit- jusqu'à 23.0 (+ ?)
	**Opera 10.5+** **Opera Mobile 11+**	à partir de 12.5 : sans préfixe -o- et 11.0 sans préfixe pour Opera Mobile
	**Safari 3.1+** **Safari Mobile (iOS 3.2)**	avec préfixe -webkit-
	**Android Browser 2.1+**	avec préfixe -webkit-

Ressources

- http://www.w3.org/TR/css3-2d-transforms/ » http://www.w3.org/TR/css3-2d-transforms/
- http://www.useragentman.com/matrix/ » http://www.useragentman.com/matrix/