Transformations

Avec CSS3 sont arrivées de nouvelles façons de positionner et de modifier des éléments. Maintenant les techniques générales de mise en page peuvent être revisitées avec d'autres moyens de retailler, positionner et modifier des éléments. Toutes ces nouvelles techniques sont rendues possibles grâce à la propriété `transform`.

La propriété `transform` se décline selon deux paramètres différents, en deux dimensions et en trois dimensions. Chacun d'entre eux viennent avec leurs propres propriétés individuelles et leurs valeurs.
Au sein de cette leçon, nous allons jeter un oeil à la fois aux transformations en deux et en trois dimensions. D'une manière générale, le support des navigateurs pour la propriété `transform` s'améliore de jours en jours mais vous devez encore utiliser les préfixes des navigateurs pour un support le plus large possible.

Les exemples de ce cours peuvent se trouver à l'url suivante

https://codepen.io/collection/qcbus/

La syntaxe

La syntaxe réelle de la propriété `transform` est assez simple, y compris la propriété `transform` suivi de la valeur. La valeur spécifie le type de transformation suivie d'une quantité spécifique entre parenthèses.

```
div {
  -webkit-transform: scale(1.5);
     -moz-transform: scale(1.5);
       -o-transform: scale(1.5);
          transform: scale(1.5);
}
```

Remarquez comment la propriété `transform` comprend plusieurs préfixes fournisseurs pour obtenir le meilleur support de tous les navigateurs. La déclaration non préfixée vient en dernier pour remplacer les versions préfixées si le navigateur prend en charge la propriété `transform`.

Dans un souci de brièveté, le reste de cette leçon ne comprend pas de préfixes fournisseurs. Ils sont cependant fortement encouragés pour tout code dans un environnement de production. Au fil du temps, nous serons en mesure d'éliminer ces préfixes, mais les garder est l'approche la plus sûre pour le moment.

Transformation 2D

Les éléments peuvent être déformées, ou transformées, à la fois sur un plan à deux dimensions ou en trois dimensions. Le travail se transforme en deux dimensions sur les axes `x` et `y`, appelés axes horizontaux et verticaux. Les transformations en trois dimensions fonctionnent aussi sur les axes `x` et `y` ainsi que sur l'axe `z`. Ces trois dimensions vont permettre de définir non seulement la longueur et la largeur d'un élément, mais aussi la profondeur. Nous allons commencer par voir la façon de transformer des éléments sur un plan en deux dimensions, et nous irons jusqu'aux transformations en trois dimensions.

2D Rotate

La propriété `transform` accepte une poignée de valeurs différentes. La valeur `rotate` fournit la possibilité de faire pivoter un élément de `0` à `360` degrés. En utilisant une valeur positive l'élément va tourner dans le sens horaire, et en utilisant une valeur négative l'élément va tourner dans le sens antihoraire. Le point par défaut de rotation est le centre de l'élément, soit `50% 50%` à la fois horizontalement et verticalement. Plus tard, nous verrons comment vous pouvez changer ce point de rotation par défaut.

HTML

```
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
```

CSS

```
.box-1{
  transform: rotate(20deg);
}
.box-2{
  transform: rotate(-55deg);
}
```

Démo rotation

La boîte grise derrière l'élément en rotation symbolise la position initiale de l'élément. En outre, si vous passez la souris sur la zone l'élément tourne de 360 degrés horizontalement. Cette boîte grise sera pour chaque démonstration une référence pour la position initiale de l'élément et la rotation horizontale pour aider à démontrer une altération des éléments et de la profondeur.

2D Scale

L'utilisation de la valeur `scale` dans la propriété `transform` vous permet de changer la taille d'un élément. La valeur d'échelle par défaut est `1`, par conséquent , une valeur quelconque entre `.99` et `.01` fait apparaître un élément de taille inférieure alors que toute valeur supérieure ou égale à `1.01` fait apparaître un élément plus grand.

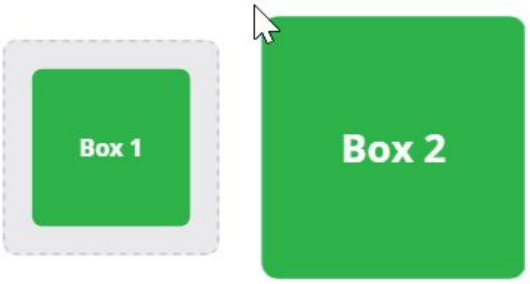
HTML

```
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
```

CSS

```
.box-1 {
  transform: scale(.75);
}
.box-2 {
  transform: scale(1.25);
}
```

Démo échelle



Il est possible de ne mettre à l'échelle que la hauteur ou la largeur d'un élément en utilisant la valeurs `scaleX` et `scaleY`. La valeur `scaleX` met à l'échelle la largeur d'un élément pendant que la valeur `scaleY` met à l'échelle la hauteur d'un élément. Pour mettre à l'échelle à la fois la hauteur et la largeur d'un élément, mais à des tailles différentes, les valeurs `x` et les `y` peuvent être réglées simultanément. Pour ce faire, utilisez la valeur `scale` en déclarant la valeur de l'axe `x` en premier, suivi d'une virgule, puis la valeur de l'axe `y` .

HTML

```
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
<figure class="box-3">Box 3</figure>
```

CSS

```
.box-1 {
  transform: scaleX(.5);
}
.box-2 {
  transform: scaleY(1.15);
}
.box-3 {
  transform: scale(.5, 1.15);
}
```

Démo multiple “Mise à l'échelle”

2D Translate

La valeur `translate` fonctionne un peu comme celle de positionnement relatif, pousser et tirer un élément dans des directions différentes sans interrompre le flux normal du document. L’utilisation de la valeur `translateX` va changer la position d'un élément sur l'axe horizontal, la valeur `translateY` change la position d'un élément sur l'axe vertical.

Comme avec la valeur `scale`, on peut définir les deux valeurs d'axe à la fois, en utilisant la valeur `translate` et déclarer en premier la valeur `x` de l'axe, suivi d'une virgule, puis la valeur `y` de l' axe.
Les valeurs de distance utilisées dans la valeur `translate` peut être toute mesure de longueur générale, plus couramment ce seront les pixels ou les pourcentages. Les valeurs positives pousseront un élément vers le bas et à droite de sa position par défaut tandis que les valeurs négatives tireront un élément vers le haut et à gauche de sa position par défaut.

HTML

```html
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
<figure class="box-3">Box 3</figure>
```

CSS

```css
.box-1 {transform: translateX(-10px);}
.box-2 {transform: translateY(25%);}
.box-3 {transform: translate(-10px, 25%);}
```

Démo Translate

2D Skew

La dernière valeur `transform` dans le groupe, est `skew` utilisée pour déformer les éléments sur l'axe horizontal, l'axe vertical ou les deux. La syntaxe est très similaire à celle du `scale` et des valeurs `translate`. Utilisation de la valeur `skewX` déforme un élément sur l'axe horizontal tandis que la valeur `skewY` déforme un élément sur l'axe vertical. Pour déformer un élément sur les deux axes la valeur `skew` est utilisée, en déclarant la valeur de l'axe `x` en premier, suivi d'une virgule, puis la valeur de `y` l'axe.
Le calcul de la distance de la valeur `skew` est mesurée en unités de degrés. Les mesures de longueur, tels que les pixels ou en pourcentage, ne sont pas applicables ici.

HTML

```html
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
<figure class="box-3">Box 3</figure>
```

CSS

```css
.box-1 {
  transform: skewX(5deg);
}
.box-2 {
  transform: skewY(-20deg);
}
.box-3 {
  transform: skew(5deg, -20deg);
}
```

Demo skew

La combinaison Transform

Il est fréquent que plusieurs transformations soient utilisées à la fois, la rotation et la mise à l'échelle de la taille d'un élément en même temps par exemple. Dans ce cas, plusieurs transformations peuvent être combinées ensemble. Pour combiner les transformations, il faut ajouter la liste des valeurs de transformation au sein de la propriété `transform` l'une après l'autre sans l'utilisation de virgules.
L'utilisation de plusieurs déclarations `transform` ne fonctionnera pas, car chaque déclaration remplace celle au-dessus. Le comportement dans ce cas serait le même que si vous deviez définir les `height` d'un élément à plusieurs reprises.

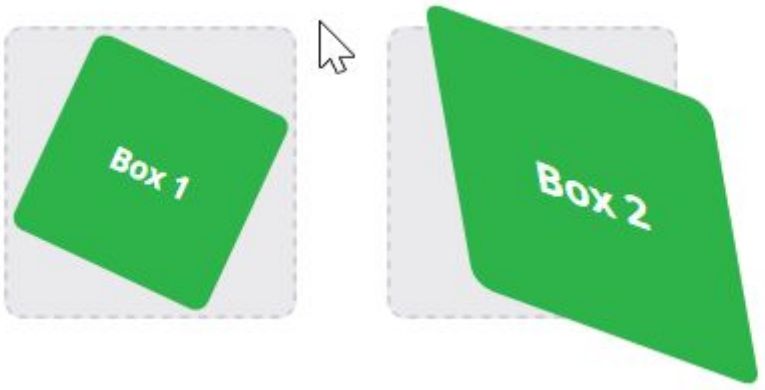
HTML

```html
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
```

CSS

```css
.box-1 {
  transform: rotate(25deg) scale(.75);
}
.box-2 {
  transform: skew(10deg, 20deg) translateX(20px);
}
```

La combinaison Demo Transforms



Derrière chaque transformation il y a aussi une matrice définissant explicitement le comportement de la transformation. L'utilisation de `rotate`, `scale`, `transition`, et des valeurs `skew` offrent un moyen facile d'établir cette matrice. Cependant, si vous êtes matheux, et que vous souhaitez approfondir les transformations, essayez-vous de vous renseigner sur la propriété `matrix`.

Démo Cube 2D

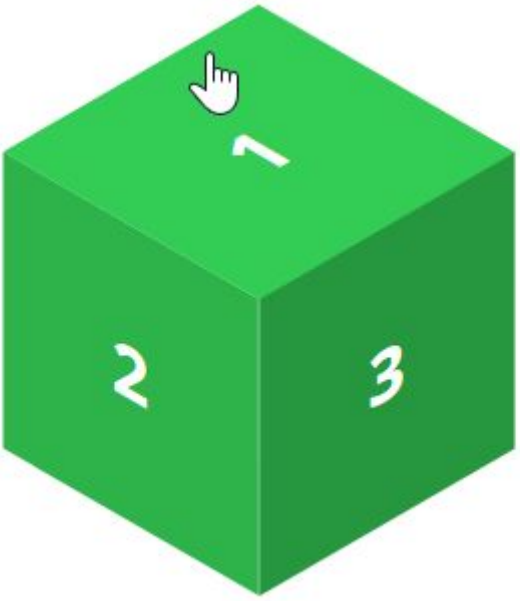
HTML

```html
<div class="cube">
  <figure class="side top">1</figure>
  <figure class="side left">2</figure>
  <figure class="side right">3</figure>
</div>
```

CSS

```css
.cube { position: relative;}
.side { height: 95px;position: absolute;width: 95px;}
.top { background: #9acc53;
transform: rotate(-45deg) skew(15deg, 15deg);}
.left { background: #8ec63f;
transform: rotate(15deg) skew(15deg, 15deg)
translate(-50%, 100%);}
.right { background: #80b239;
  transform: rotate(-15deg) skew(-15deg, -15deg)
translate(50%, 100%);}
```

Démo



Transform-origin

Comme mentionné précédemment, la valeur par défaut de transform-origin est le point mort d'un élément, à la fois `50%` horizontalement et `50%`verticalement. Pour modifier cette position d'origine par défaut la propriété `transform-origin` peut être utilisée. La propriété `transform-origin` peut accepter une ou deux valeurs. Lorsqu'une seule valeur est spécifiée, cette valeur est utilisée pour les deux axes horizontal et vertical. Si deux valeurs sont spécifiées, le premier est utilisé pour l'axe horizontal et le second est utilisé pour l'axe vertical.
Individuellement, les valeurs sont traitées comme celle d'une position de l'image d'arrière-plan, en utilisant soit une longueur ou de la valeur clé. Cela dit, `0 0` est la même valeur que `top left`, et `100% 100%` est la même valeur que `bottom right`. Des valeurs spécifiques peuvent également être définies, par exemple `20px 50px` fixerait l'origine de 20 pixels vers la gauche et 50 pixels vers le bas l'élément.

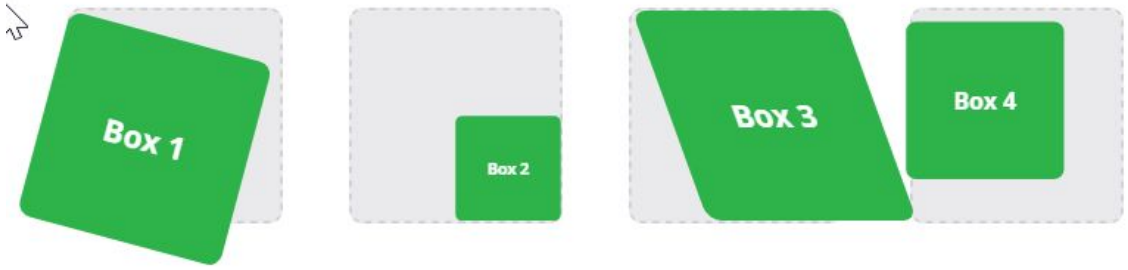
HTML

```
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
<figure class="box-3">Box 3</figure>
<figure class="box-4">Box 3</figure>
```

CSS

```css
.box-1 {
  transform: rotate(15deg);
  transform-origin: 0 0;
}
.box-2 {
  transform: scale(.5);
  transform-origin: 100% 100%;
}
.box-3 {
  transform: skewX(20deg);
  transform-origin: top left;
}
.box-4 {
  transform: scale(.75) translate(-10px, -10px);
  transform-origin: 20px 50px;
}
```

Transformer Origin Demo



La propriété `transform-origin` se heurte à certains problèmes lorsque vous utilisez également la valeur de transformation `translate`. Etant donné que les deux tentent de positionner l'élément, leurs valeurs peuvent entrer en collision. Utilisez les deux avec prudence et assurer vous que le résultat souhaité est atteint.

Perspective

Pour les transformations en trois dimensions et pour fonctionner les éléments ont besoin d'une perspective à partir de laquelle se transformer. La perspective pour chaque élément peut être considéré comme un *point de fuite*, semblable à celle que l'on voit dans les dessins en trois dimensions.

La perspective d'un élément peut être défini de deux façons différentes. Une façon consiste à utiliser la valeur `perspective` à l'intérieur de la propriété `transform` sur des éléments individuels, tandis que l'autre comprend l'utilisation de la propriété `perspective` sur l'élément parent résidant sur les éléments enfants à transformer.
L'utilisation de la valeur `perspective` dans la propriété `transform` fonctionne très bien pour transformer un élément d'un seul et unique point de vue. Lorsque vous voulez transformer un groupe d'éléments tous avec la même perspective, ou point de fuite, le plus simple est d'appliquer la propriété `perspective` à leur élément parent.
L'exemple ci-dessous montre une poignée d'éléments tous transformés en utilisant leurs perspectives individuelles avec la valeur `perspective`.

HTML

```html
<figure class="box">Box 1</figure>
<figure class="box">Box 2</figure>
<figure class="box">Box 3</figure>
```

CSS

```css
.box {
  transform: perspective(200px) rotateX(45deg);
}
```

Démo valeur Perspective

L'exemple suivant montre une poignée d'éléments côte à côte, toutes transformées en utilisant la même perspective, réalisée en utilisant la propriété `perspective` de leur élément parent direct.

HTML

```html
<div class="group">
  <figure class="box">Box 1</figure>
  <figure class="box">Box 2</figure>
  <figure class="box">Box 3</figure>
</div>
```

CSS

```css
.group {
  perspective: 200px;
}
.box {
  transform: rotateX(45deg);
}
```

Demo de la propriété Perspective

La valeur de profondeur de Perspective

La valeur de point de vue peut être définie comme `none` ou une mesure de longueur. La valeur none désactive toute perspective, tandis que la valeur de longueur définit la profondeur de la perspective. Plus la valeur est élevée, plus la perspective paraît lointaine, créant ainsi une perspective d'intensité assez faible et un petit changement tridimensionnel. Plus la valeur est basse, plus la perspective paraît proche, créant ainsi une perspective à haute intensité et un grand changement tridimensionnel.

Imaginez-vous debout à 3 mètres d'un cube de 3 mètres par rapport à une distance de 300 mètres du même cube. A 3 mètres, votre distance au cube est identique aux dimensions du cube, donc le changement de

perspective est beaucoup plus grand qu'il le sera à 300 mètres, où les dimensions du cube ne seront qu'un centième de votre distance au cube. Le même raisonnement s'appliquera aux valeurs de profondeur en perspective.

HTML

```
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
```

CSS

```
.box-1 {
  transform: perspective(100px) rotateX(45deg);
}
.box-2 {
  transform: perspective(1000px) rotateX(45deg);
}
```

Démo de la valeur de profondeur de Perspective

Origine de la perspective

Comme avec la fixation de `transform-origin` vous pouvez également définir une `perspective-origin`. Les mêmes valeurs utilisées pour la propriété `transform-origin` peuvent également être utilisées avec la propriété `perspective-origin`.

La grande différence entre les deux est que l'origine d'une transformation détermine les coordonnées utilisées pour calculer la variation d'une transformation, tandis que l'origine d'un point de vue identifie les coordonnées du point de fuite d'une transformation.

HTML

```html
<div class="original original-1">
  <figure class="box">Box 1</figure>
</div>
<div class="original original-2">
  <figure class="box">Box 2</figure>
</div>
<div class="original original-3">
  <figure class="box">Box 3</figure>
</div>
```

CSS

```css
.original {
  perspective: 200px;
}
.box {
  transform: rotateX(45deg);
}
.original-1 {
  perspective-origin: 0 0;
}
.original-2 {
  perspective-origin: 75% 75%;
}
.original-3 {
  perspective-origin: 20px 40px;
}
```

Démo Perspective-origin

3D Transforms

Le fait de travailler avec deux dimensions de transformation nous sommes en mesure de modifier des éléments sur les axes horizontaux et verticaux, mais il y a un autre axe le long duquel nous pouvons transformer des éléments. L'utilisation de la transformation en trois dimensions, nous permet de changer des éléments sur l'axe `z`, nous donnant le contrôle de la profondeur en plus de la longueur et la largeur.

Rotation 3D

Jusqu'à présent, nous avons discuté de la façon de faire tourner un objet soit dans le sens anti-horaire ou sur un plan plat. Nous pouvons tourner avec des transformations en trois dimensions un élément autour des axes. Pour ce faire, nous utilisons trois nouvelles valeurs `transform`, à savoir `rotateX`, `rotateY` et `rotateZ`.
L'utilisation de la valeur `rotateX` vous permet de faire pivoter un élément autour de l'axe `x`, comme si elle était plié en deux horizontalement. Utilisation de la valeur `rotateY` vous permet de faire pivoter un élément autour de l'axe `y`, comme si elle était pliée en deux verticalement. Enfin, la valeur `rotateZ` permet de faire pivoter un élément autour de l'axe `z`. Comme dans le cas général de la valeur `rotate`, les valeurs positives vont faire tourner l'élément autour de son axe dans le sens horaire, tandis que les valeurs négatives vont tourner l'élément dans le sens antihoraire.

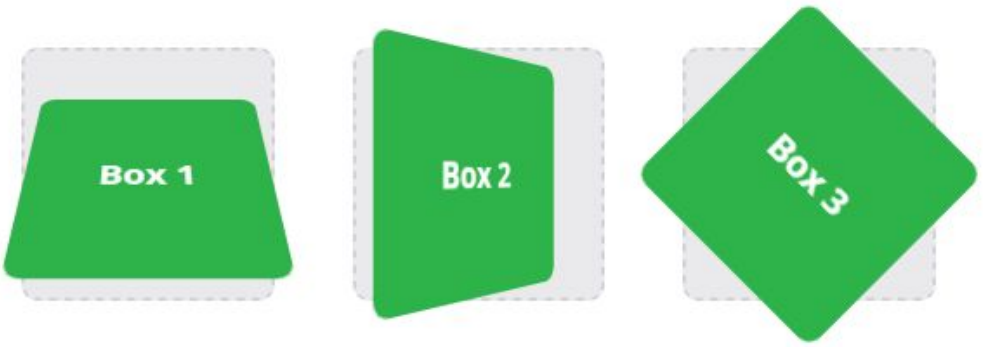
HTML

```html
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
<figure class="box-3">Box 3</figure>
```

CSS

```css
.box-1 { transform: perspective(200px) rotateX(45deg);}
.box-2 { transform: perspective(200px) rotateY(45deg);}
.box-3 { transform: perspective(200px) rotateZ(45deg);}
```

Demo Rotation 3D



Echelle 3D

En utilisant `scaleZ` les éléments de transformation en trois dimensions peuvent être mis à l'échelle sur l'axe `z`. Ce n'est pas très excitant quand aucune autre transformation en trois dimensions est mise en place, car il n'y a rien en particulier à mettre à l'échelle. Dans la démonstration ci-dessous les éléments sont mis à l'échelle et vers le bas sur l'axe `z`, mais la valeur `rotateX` est ajoutée afin de voir le comportement de la valeur `scaleZ`. Lors du retrait de la valeur `rotateX` dans ce cas, les éléments apparaissent être les mêmes.

HTML

```
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
```

CSS

```
.box-1 {
  transform: perspective(200px) scaleZ(1.75)
rotateX(45deg);
}
.box-2 {
  transform: perspective(200px) scaleZ(.25)
rotateX(45deg);
}
```

Démo 3D échelle

3D Translate

Les éléments peuvent également être traduits sur l'axe `z` en utilisant la valeur `translateZ`. Une valeur négative ici va pousser un élément plus loin sur l'axe `z`, ce qui en fait un plus petit élément. En utilisant une valeur positive cela va tirer un élément plus prêt de l'axe `z`, ce qui en fait un plus grand élément.

Bien que cela puisse sembler être très similaire à la transformation en deux dimensions de la valeur `scale`, il est en réalité tout à fait différent. La transformation se déroule sur l'axe `z`, et pas sur les axes `x` ou `y`. Lorsque vous travaillez avec des transformations tridimensionnelles, pouvoir déplacer un élément sur l'axe z présente de grands avantages, comme lors de la construction du cube ci-dessous, par exemple.

HTML

```
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
```

CSS

```
.box-1 {
  transform: perspective(200px) translateZ(-50px);
}
.box-2 {
  transform: perspective(200px) translateZ(50px);
}
```

Démo 3D Translate

3D Skew

Skew est la transformation bidimensionnelle qui ne peut pas être transformée sur une échelle tridimensionnelle. Les éléments peuvent être faussés sur les axes `x` et `y`, puis transformés en trois dimensions comme souhaité, mais ils ne peuvent être biaisées sur l'axe `z`.

Raccourcis 3D Transforms

Comme pour combiner les transformations bidimensionnelles, il existe également des propriétés pour écrire des transformations tridimensionnelles abrégées. Ces propriétés comprennent `rotate3d`, `scale3d`, `transition3d` et `matrix3d`. Ces propriétés nécessitent un bon niveau de maths, ainsi qu'une bonne compréhension des matrices derrière chaque transformation. Si vous êtes intéressé à regarder vous pourrez approfondir la question par vous même !

Transform-style

À l'occasion, des transformations tridimensionnelles seront appliquées sur un élément imbriqué dans un élément parent qui se transforme également. Dans cet événement, les éléments imbriqués et transformés n'apparaîtront pas dans leur propre espace tridimensionnel. Pour permettre aux éléments imbriqués de se transformer dans leur propre plan tridimensionnel, utilisez la propriété `transform-style` avec la valeur `preserve-3d`.
La propriété `transform-style` doit être placée sur l'élément parent, au-dessus des transformations imbriquées. La valeur `preserve-3d` permet aux éléments enfants transformés d'apparaître dans leur propre plan en trois dimensions alors que la valeur `flat` oblige les éléments enfants transformés à être à plat sur le plan à deux dimensions.

HTML

```
<div class="rotate three-d">
  <figure class="box">Box 1</figure>
</div>
<div class="rotate">
  <figure class="box">Box 2</figure>
</div>
```

CSS

```css
.rotate {
  transform: perspective(200px) rotateY(45deg);
}
.three-d {
  transform-style: preserve-3d;
}
.box {
  transform: rotateX(15deg) translateZ(20px);
  transform-origin: 0 0;
}
```

Démo transform-style

Pour voir un exemple supplémentaire de la propriété `transform-style` en action suivez ces explications.

Backface-visibility

Lorsque vous travaillez avec des transformations tridimensionnelles, des éléments seront occasionnellement transformés de manière à faire face à l'écran. Cela peut être causé par le réglage de la valeur `rotateY(180deg)` par exemple. Par défaut, ces éléments sont affichés à l'arrière. Donc, si vous préférez ne pas voir ces éléments du tout, définissez la propriété `backface-visibility` à `hidden`, et vous cacherez l'élément lorsqu'il est éloigné de l'écran.

L'autre valeur `backface-visibility` est `visible` qui est la valeur par défaut, affichera toujours un élément, quelle que soit la direction à laquelle il est confronté.

Dans la démonstration ci-dessous, notez comment la deuxième case n'est pas affichée car la déclaration `backface-visibility:hidden;` a été définie. La propriété `backface-visibility` prend encore plus d'importance lors de l'utilisation d'animations.

HTML

```html
<figure class="box-1">Box 1</figure>
<figure class="box-2">Box 2</figure>
```

CSS

```css
.box-1 {
  transform: rotateY(180deg);
}
.box-2 {
  backface-visibility: hidden;
  transform: rotateY(180deg);
}
```

Démo Backface-visibility

Démo Cube 3D

HTML

```html
<div class="cube-container">
  <div class="cube">
    <figure class="side front">1</figure>
    <figure class="side back">2</figure>
    <figure class="side left">3</figure>
    <figure class="side right">4</figure>
    <figure class="side top">5</figure>
    <figure class="side bottom">6</figure>
  </div>
</div>
```

CSS

```css
.cube-container {
height: 200px;
perspective: 300;
position: relative;
width: 200px;
}

.cube {
  height: 100%; position: absolute;
  transform: translateZ(-100px);
  transform-style: preserve-3d; width: 100%;
}

.side {
  background: rgba(45, 179, 74, .3);
  border: 2px solid #2db34a;
  height: 196px; position: absolute; width: 196px;
}

.front{
transform: translateZ(100px);
}
.back{
transform: rotateX(180deg) translateZ(100px);
}
.left{
transform: rotateY(-90deg) translateZ(100px);
}
.right{
transform: rotateY(90deg) translateZ(100px);
}
.top {
  transform: rotateX(90deg) translateZ(100px);
}
.bottom {
  transform: rotateX(-90deg) translateZ(100px);
}
```

Ressources et liens

- Transformer la propriété via CSS3 fichiers
- Comprendre le CSS Transforms matrice via Dev.Opera
- Introduction à CSS 3D Transformspar 24 voies
- Transformer Fonction via Mozilla Developer Network
- Transformer style via WebKit
- Visibilité Backface via CSS-Tricks