

CSS- 07- a – Transformations et Transitions

Transformations

La propriété CSS transform permet d'effectuer des transformations géométriques sur un élément. Contrairement aux propriétés traditionnelles, elle accepte plusieurs valeurs. Il est même possible d'effectuer des combinaisons afin d'accomplir des transformations complexes.

Comparativement aux autres propriétés, par exemple: top ou left, la propriété transform n'a aucun impact sur le positionnement des éléments adjacents.

translate ↔

Par défaut, cette valeur accepte un ou deux arguments entre parenthèses. Le 1^{er} correspond à une translation sur les X ↔ et le 2^e, (*si présent*), sur les Y ↑↓. Les arguments peuvent-être n'importe quelle [unité de mesure CSS](#) (px, em, %, etc.)

Un translate avec un seul argument
transform: translate(100%);
déplace l'élément sur les X ↔.

Tandis qu'avec deux arguments **séparés par une virgule**
transform: translate(100%, 100%);
l'élément se déplace sur les X et Y ↘.

Voir l'exemple ci-dessous:

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame no1">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
  <!-- ===== -->
  <div>
    <div class="frame no2">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

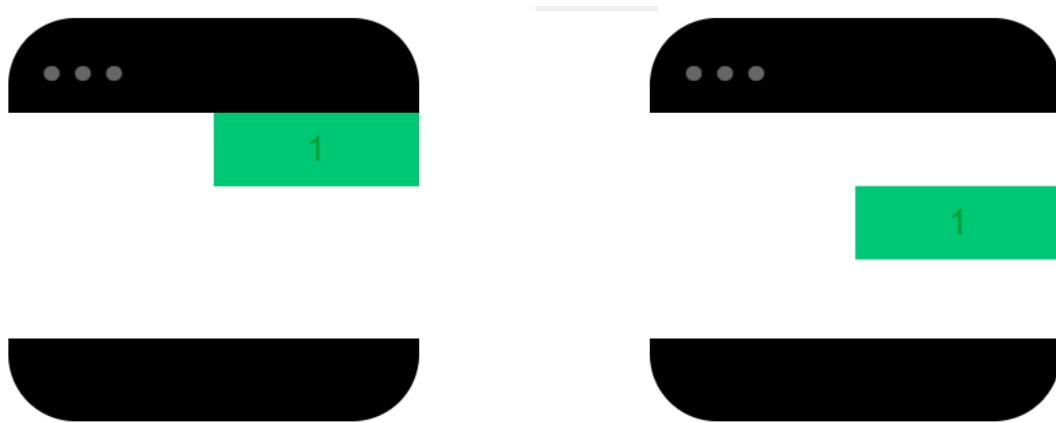
Code CSS :

```
.element {
  width: 50%;
}

.frame.no1 .element {
  transform: translate(100%);
}

.frame.no2 .element {
  transform: translate(100%, 100%);
}
```

Rendu :



À gauche, transform: translate(100%, 0);

À droite, transform: translate(100%, 100%);

Dans le cadre d'une translation, les % sont relatifs à la dimension de l'élément en question.

- 100% sur les X = Une largeur de l'élément.
- 100% sur les Y = Une hauteur de l'élément.

Il est possible d'effectuer une translation sur un seul axe en spécifiant cet axe dans la nom de la valeur. Par exemple:

- translateX(...) uniquement sur les X (*équivalent à* translate(...))
- translateY(...) uniquement sur les Y
- translateZ(...) uniquement sur les Z

Pour qu'une translation sur l'axe des Z soit perceptible, il est nécessaire qu'une perspective soit donnée à l'élément ou à l'un de ses parents.

Il est également possible de combiner les 3 axes dans une valeur en utilisant translate3d(...).

- [transform: translate\(\)](#)

rotate

Cette valeur accepte comme argument les unités de type:

- deg degrés | 360 degrés dans un cercle
- turn turns | 1 turn/tour dans un cercle

Qui plus est, cette valeur peut-être:

- Positive pour aller dans le sens horaire des aiguilles d'une montre
- Négative pour aller dans le sens inverse

Par exemple, une rotation de 45° dans le sens horaire:

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Code CSS :

```
.element {
  width: 50%;
  transform: translate(50%) rotate(45deg);
}
```

Rendu :



Remarquez dans l'exemple précédent comment il est possible de combiner deux valeurs de transformation en les séparant avec un espace.

Il est possible d'effectuer une rotation sur un seul axe en spécifiant cet axe dans la nom de la valeur. Par exemple:

- rotateZ(...) uniquement sur les Z (*équivalent à rotate(...)*)
- rotateX(...) uniquement sur les X
- rotateY(...) uniquement sur les Y

Par exemple: RotationX vs RotationY

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame no1">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
  <!-- ===== -->
  <div>
    <div class="frame no2">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Code CSS :

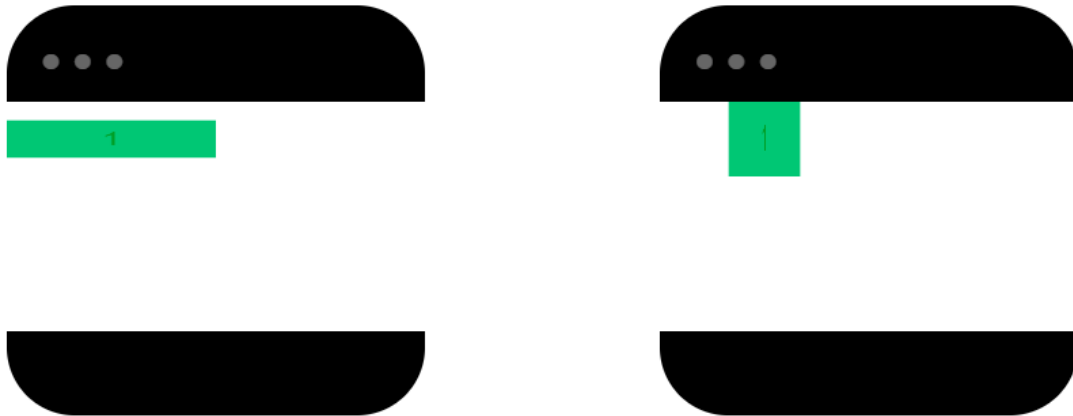
```
.frame {
  perspective: 800px;
}

.element {
  width: 50%;
}

.frame.no1 .element {
  transform: rotateX(60deg);
}

.frame.no2 .element {
  transform: rotateY(70deg);
}
```

Rendu :



À gauche, transform: rotateX(60deg).
À droite, transform: rotateY(60deg).

- [transform: rotate\(\)](#)

scale

Par défaut, cette valeur accepte un ou deux arguments entre parenthèses. Si un seul argument est passé, il correspond à un facteur d'agrandissement/réduction. Si un 2^e argument est ajouté, le premier correspond à un facteur d'agrandissement/réduction sur les X et le deuxième sur les Y. Il est donc possible de déformer un élément.

Voir l'exemple ci-dessous. Le 1^{er} est agrandi de façon proportionnelle de 150% (1.5) et le 2^e de 150% (1.5) sur les X et 300% (3) sur les Y.

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame no1">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
  <!-- ===== -->
  <div>
    <div class="frame no2">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

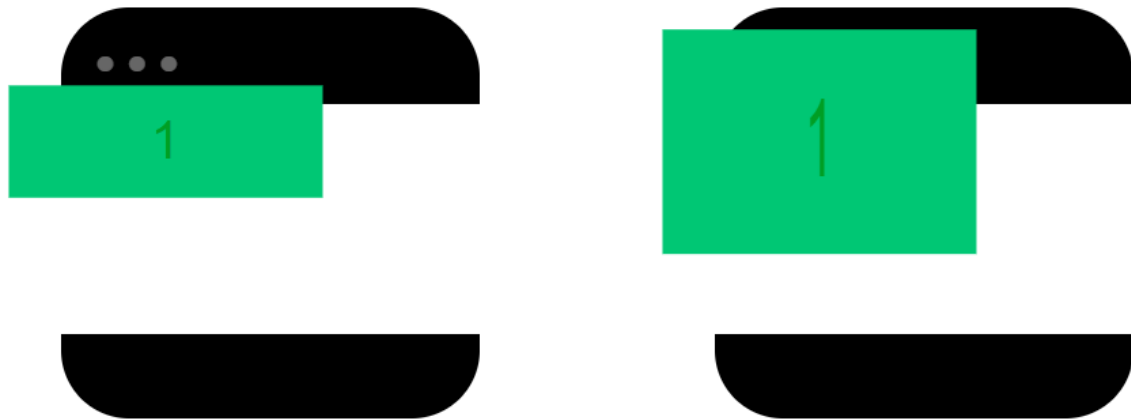
Code CSS :

```
.element {
  width: 50%;
}

.frame.no1 .element {
  transform: scale(1.5);
}

.frame.no2 .element {
  transform: scale(1.5, 3);
}
```


Rendu :



À gauche, transform: scale(1.5).

À droite, transform: scale(1.5, 3).

Il est possible d'effectuer un scale sur un seul axe en spécifiant cet axe dans la nom de la valeur. Par exemple:

- scaleX(...) uniquement sur les X
- scaleY(...) uniquement sur les Y
- [transform: scale\(\)](#)

skew

Par défaut, cette valeur accepte un ou deux arguments entre parenthèses. Si un seul argument est passé, il correspond à une distorsion sur les X. Si un 2e argument est ajouté, le premier correspond à une distorsion sur les X et le deuxième sur les Y.

Cette valeur accepte comme argument les unités de type:

- deg degrés | 360 degrés dans un cercle
- turn tour | 1 tour dans un cercle

Par exemple, une distorsion de 45° sur les X:

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Code CSS :

```
.element {
  width: 50%;
  transform: translate(50%) skew(45deg);
}
```

Rendu :



Il est possible d'effectuer un skew sur un seul axe en spécifiant cet axe dans la nom de la valeur. Par exemple:

- `skewX(...)` uniquement sur les X (*équivalent à `skew(...)`*)
- `skewY(...)` uniquement sur les Y
- [transform: skew\(\)](#)

transform-origin

Toutes les transformations des exemples précédents sont effectuées à partir d'un point d'origine se situant au centre de chaque élément. Soit l'équivalent de `transform-origin: 50% 50%;`

Cependant, il est possible de modifier cette valeur avec n'importe quelle [unité CSS](#).

Par exemple, voici la même rotation de 45°:

- À gauche avec un point de rotation central (*50% 50%*)
- À droite avec un point de rotation en haut à gauche (*0 0*).

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame no1">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
  <!-- ===== -->
  <div>
    <div class="frame no2">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

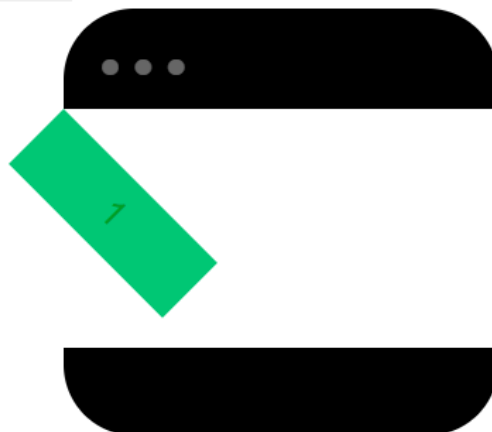
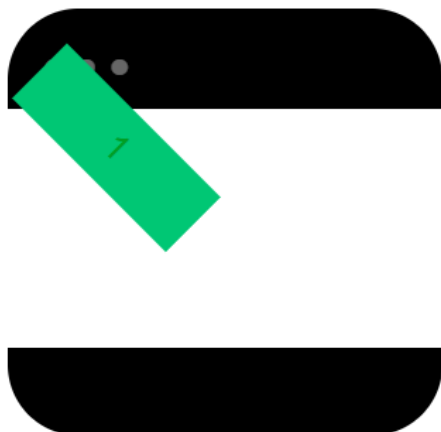
Code CSS

```
.element {
  width: 50%;
}

.frame.no1 .element {
  transform-origin: 50% 50%;
  transform: rotate(45deg);
}

.frame.no2 .element {
  transform-origin: 0 0;
  transform: rotate(45deg);
}
```

Rendu :



À gauche, transform-origin: 50% 50%.

À droite, transform-origin: 0 0;

Transitions

Les transitions permettent de passer d'un état A à un état B d'une façon élégante.

transition-property

Définis la ou les propriété(s) affectées par une transition.

Il est possible, d'indiquer que toutes les propriétés capables de transitionner doivent transitionner en utilisant la valeur all.

Utiliser la valeur all avec parcimonie afin d'éviter d'impacter négativement les performances de votre transition.

Il est aussi possible de spécifier un certain nombre de propriétés en les séparant par une virgule.

Par exemple:

transition-property: background-color, transform;

ERREUR FRÉQUENTE

Ce ne sont pas toutes les propriétés qui peuvent effectuer une transition. Par exemple, display ou background-image ne peut pas effectuer de transition. Voir la [liste MDN des propriétés capable d'effectuer une transition](#).

- [transition-property](#)
- [transition-property](#)

transition-duration

Définis la durée de la transition. Ce nombre peut-être en seconde ou en milliseconde.

1s = 1000ms.

Par exemple, au survole nous avons trois fois la même transition, mais avec des durées différentes:

1. 0.5 seconde
2. 1 seconde
3. 1.5 secondes

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame no1">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
          <div class="element no2">2</div>
          <div class="element no3">3</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Code CSS :

```
.element {
  transform: translateX(0);
  transition-property: transform;
}

.frame:hover .element {
  transform: translateX(400%);
}

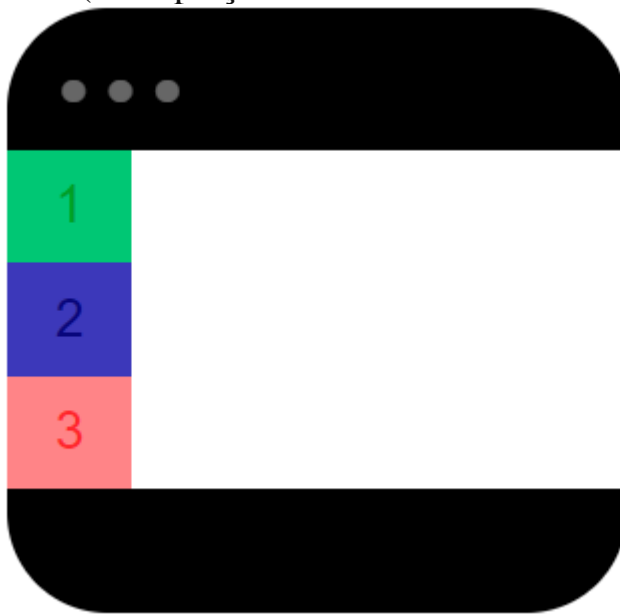
.no1 { transition-duration: 0.5s; }
.no2 { transition-duration: 1s; }
.no3 { transition-duration: 1.5s; }

/* Layout */
.flex {
  display: flex;
  flex-direction: column;
  height: 100%;
  overflow: hidden;
}

.element {
  flex: 0 0 33.333%;
```

```
width: 20%;  
}
```

Rendu (en déplaçant la souris sur vos carrés)






- [transition-duration](#)
- [transition-duration](#)

transition-timing-function

Dicte le rythme de la transition. Par exemple, dans la transition précédente on remarque que chaque carré accélère progressivement avant de ralentir ensuite. Ce rythme est appelé ease et est celui par défaut des transitions.

L'exemple suivant contient six fois la même transition au survole, mais avec des rythmes différents.

1. linear n'accélère ou ne ralentis jamais 
2. ease accélère progressivement et ralentis en fin de parcours (*Comportement par défaut*)
3. ease-in-out commence lentement et ralentis en fin de parcours.
4. ease-in commence lentement 
5. ease-out ralentis en fin de parcours 
6. cubic-bezier Rythme personnalisable via une courbe de Bézier.

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame no1">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
          <div class="element no2">2</div>
          <div class="element no3">3</div>
        </div>
      </div>
    </div>
  </div>

  <div>
    <div class="frame no2">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no4">4</div>
          <div class="element no5">5</div>
          <div class="element no6">6</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Code CSS :

```
.element {
  transition-property: transform;
  transition-duration: 0.5s;
  transform: translateX(0);
}

.frame:hover .element {
  transform: translateX(400%);
}

.no1 { transition-timing-function: linear; }
.no2 { transition-timing-function: ease; }
.no3 { transition-timing-function: ease-in-out; }

.no4 { transition-timing-function: ease-in; }
.no5 { transition-timing-function: ease-out; }
.no6 { transition-timing-function: cubic-bezier(.6,-0.85,.21,1.53); }

/* Layout */
.flex {
  display: flex;
  flex-direction: column;
  height: 100%;
  overflow: hidden-y;
}
```

CSS 07 a

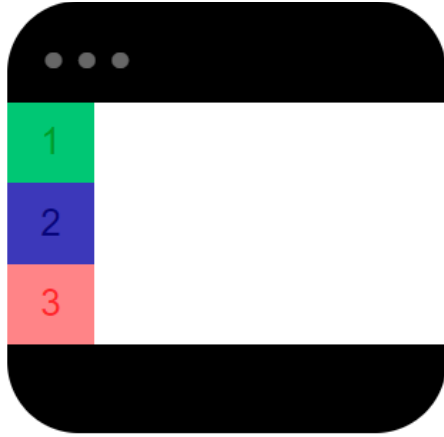
Réalisation : Guillaume DELACROIX Formateur AFPA

11 janvier 2023

Afpa

```
.element {  
  flex: 0 0 33.333%;  
  width: 20%;  
}
```

Rendu (en survolant les carrés avec la souris)



- [transition-timing-function](#)
- [transition-timing-function](#)

transition-delay

Définis le délai d'attente avant de démarrer une transition. Par défaut, cette propriété est à 0s. Si une valeur négative est attribuée, la transition débutera déjà commencée, comme-ci l'équivalent de la valeur s'était déjà écoulé.

Par exemple au survole:

1. Aucun délai
2. Délais de 0.25s
3. Délais de -0.25s (*commence au milieu de l'animation*)

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame no1">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
          <div class="element no2">2</div>
          <div class="element no3">3</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

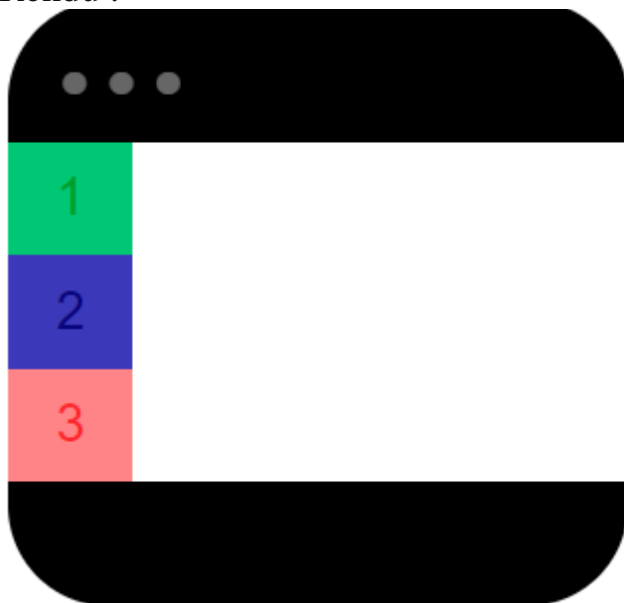
Code CSS :

```
.element {
  transform: translateX(0%);
  transition-property: transform;
  transition-duration: 0.5s;
}

.frame:hover .element {
  transform: translateX(400%);
}

.no2 { transition-delay: 0.25s; }
.no3 { transition-delay: -0.25s; }
```

Rendu :



- [transition-delay](#)
- [transition-delay](#)

Syntaxe courte

Afin de simplifier l'écriture des transitions, il est possible de regrouper toutes les précédentes propriétés en une seule comme le font les propriétés margin et padding avec top, right, bottom et left.

Pour ce faire, il suffit d'appeler la propriété transition et de lui passer au minimum deux valeurs:

1. La ou les propriétés à transitionner
2. La durée de la transition

Par exemple:

```
transition-property: transform;
```

```
transition-duration: 1s
```

Pourrait devenir

```
transition: transform 1s;
```

Il est aussi possible de spécifier les autres propriétés si désiré en les ajoutant à la suite.

Par exemple, pour avoir un délai de 0.5s et une transition linéaire:

```
transition: transform 1s 0.5s linear;
```

Où définir sa transition

Si les propriétés d'une transition sont définies sur un état en particulier, par exemple `:hover`, et non à sa base, cette propriété ne s'appliquera que lorsque cet état sera activé. Par exemple, à gauche la durée de la transition est appliquée sur l'élément de base. Le navigateur applique donc le `transition-duration` en tout temps sur l'élément, qu'il soit survolé ou non. Tandis qu'à droite, le `transition-duration` est défini sur le `:hover` uniquement, donc la transition ne s'effectue qu'au survol. Dès que l'élément n'est plus survolé, il retourne abruptement à sa position de départ.

Code HTML :

```
<div class="layout">
  <div>
    <div class="frame no1">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>

  <div>
    <div class="frame no2">
      <div>
        <!-- Code that matter -->
        <div class="flex">
          <div class="element no1">1</div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Code CSS :

```
.no1 .element {
  transition-property: transform;
  transition-duration: 0.5s;
  transform: translateX(0);
}

.frame.no1:hover .element {
  transform: translateX(400%);
}

.no2 .element {
  transform: translateX(0);
}

.frame.no2:hover .element {
  transition-property: transform;
  transition-duration: 0.5s;
  transform: translateX(400%);
}

/* Layout */
.element {
  flex: 0 0 33.333%;
  width: 20%;
}
```

Rendu :



À gauche, transition sur l'élément de base.
À droite, transition sur l'élément survolé seulement.