Transformation, transitions et animations

I. Transformation 2D

Avec la propriété CSS3 **transform**, vous pourrez déplacer, redimensionner, incliner, pivoter, déformer un élément.

La transformation pourra aussi se faire en 3 dimensions.

A. Les méthodes de 2D transform

transform: translate()

transform: rotate()

transform: scale()

transform: skew()

transform: matrix()

B. La méthode translate()

Avec la méthode translate (), on peut déplacer un objet à partir du point zéro situé dans le coin supérieur gauche de celui-ci et suivant les axes X et Y.

Soit le HTML suivant :

```
<div class="parent">
    <div class="translate">

    </div>
</div>
```

Fig. 1 Fichier HTML © Skill and You

Le CSS :

```
.parent {
  width:400px;
  height:400px;
  background-color: #ededff;
  border:1px solid #210042;
}

.translate {
  width: 250px;
  height: 250px;
  background-color: #ffeded;
  border:1px solid #420021;
}
```

Fig. 2 Fichier CSS avec la méthode translate() © Skill and You

Le rendu :

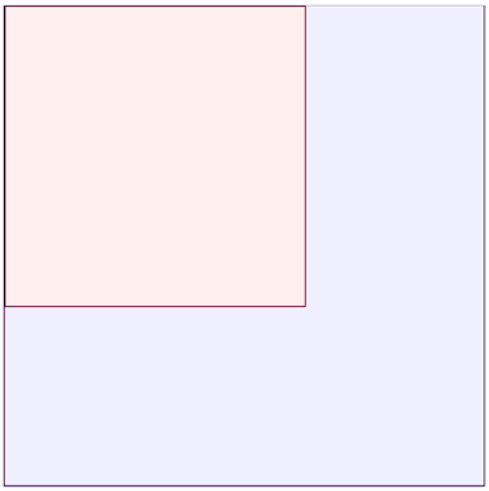

Fig. 3 Affichage dans un navigateur avec la méthode translate() © Skill and You

L'enfant couleur chair et le parent couleur violet ont les mêmes coordonnées d'origine. Avec **transform: translate()**, on va déplacer l'enfant par rapport à sa position.

Ainsi :

```
.translate {
  transform: translate(50px, 60px);
  width: 250px;
  height: 250px;
  background-color: #ffeded;
  border:1px solid #420021;
}
```

Fig.4 Déplacement de l'enfant © Skill and You

Ce qui nous donne :

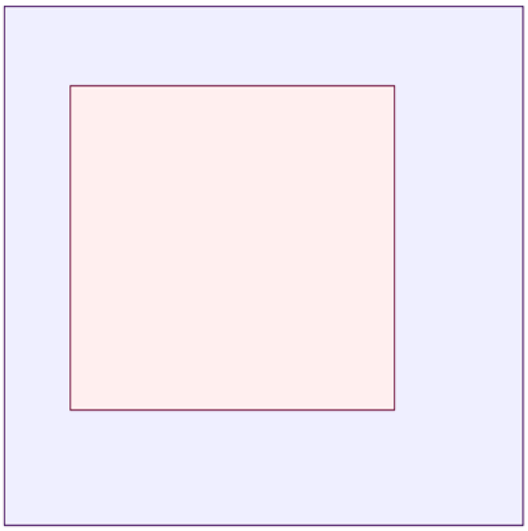

Fig.5 Affichage dans un navigateur © Skill and You

Le bloc div.translate est déplacé de 50px vers la droite et de 60 vers le bas.

C. La méthode rotate()

Avec la méthode rotate(), l'élément tourne dans le sens horaire à un degré donné. Les valeurs négatives sont autorisées et font pivoter l'élément dans le sens antihoraire.

Ainsi à partir du même HTML, en indiquant dans le CSS :

```
.rotate {
  transform: rotate(45deg);
  width: 250px;
  height: 250px;
  background-color: #ffeded;
  border:1px solid #420021;
}
```

Fig.6 Fichier CSS avec la méthode rotate() © Skill and You

On obtient :

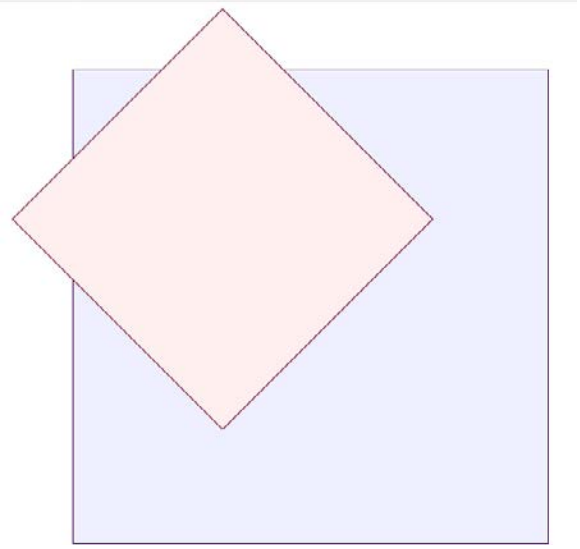

Fig.7 Affichage dans un navigateur avec la méthode rotate() © Skill and You

La valeur (45deg) de rotation fait pivoter l'élément dans le sens horaire de 45 degrés.

D. La méthode scale()

Avec la méthode scale(), l'élément augmente ou diminue de taille, en fonction des paramètres donnés pour la largeur (axe X) et la hauteur (axe Y) :

Ainsi à partir du même HTML, en indiquant dans le CSS :

```
.scale {
  transform: scale(2,3);
  width: 250px;
  height: 250px;
  background-color: #ffeded;
  border:1px solid #420021;
}
```

Fig. 8 Fichier CSS avec la méthode scale() © Skill and You

On obtient :

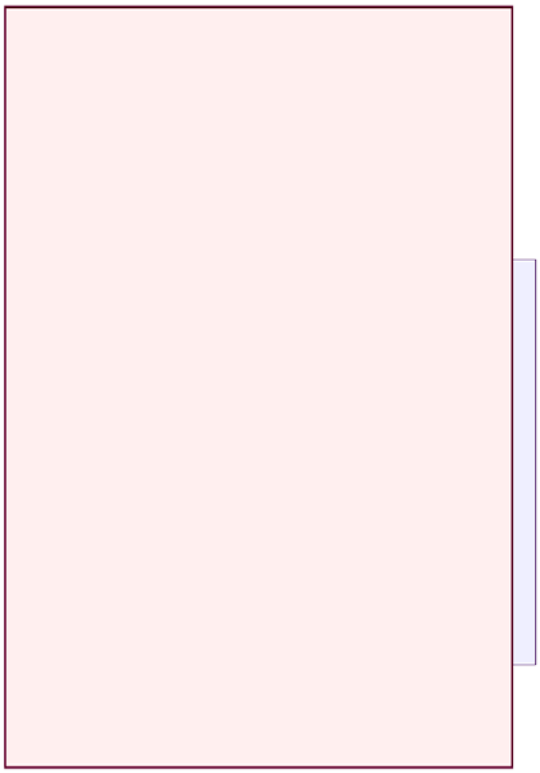

Fig. 9 Affichage dans un navigateur avec la méthode scale() © Skill and You

L'échelle de valeur (2,3) transforme la largeur à deux fois sa taille d'origine, et la hauteur à 3 fois sa taille initiale.

E. La méthode skew()

Avec la méthode d'inclinaison skew(), l'élément tourne dans un angle donné, en fonction des paramètres donnés pour l'horizontale (axe X) et la verticale (axe Y) des lignes. Ainsi, le CSS :

```
.skew {
  transform: skew(45deg, 15deg);
  width: 250px;
  height: 250px;
  background-color: #ffeded;
  border:1px solid #420021;
}
```

Fig. 10 Fichier CSS avec la méthode skew() © Skill and You

Donne :

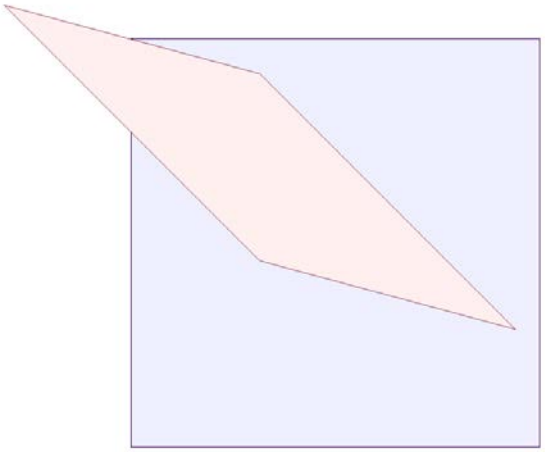

Fig. 11 Affichage dans un navigateur avec la méthode skew() © Skill and You

La valeur d'inclinaison (45deg, 15deg) tourne l'élément de 45 degrés autour de l'axe X, et 15 degrés autour de l'axe Y.

On peut regrouper plusieurs transformations en concaténant les différentes valeurs des différentes méthodes **translate**, ainsi :

```
.transform {
  transform: translate(50px,20px) rotate(45deg) scale(1.2,1.3) skew(45deg, 15deg);
  width: 250px;
  height: 250px;
  background-color: #ffeded;
  border:1px solid #420021;
}
```

Fig. 12 Fichier CSS avec les méthodes skew(), translate(), rotate() et scale() © Skill and You

Donne :

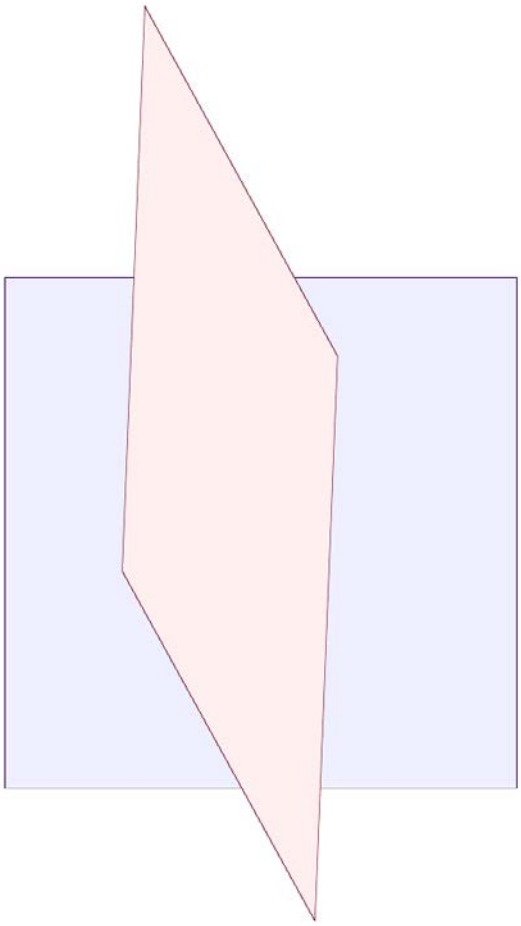

Fig. 13 Affichage dans un navigateur avec les méthodes skew(), translate(), rotate() et scale() © Skill and You

F. La méthode matrix()

Cette méthode combine toutes les méthodes de transformation 2D en une seule.

La méthode de la matrice peut prendre six paramètres, contenant des fonctions mathématiques, qui vous permettent de faire pivoter, redimensionner, déplacer, et déformer des éléments.

Nous n'entrerons pas dans les détails complexes de cette matrice de transformation. Le plus simple est alors de se rendre sur cette page :
https://www.useragentman.com/matrix/#
qui est un générateur de la méthode matrix.

Soit le CSS suivant :

```
.matrix {
  transform: matrix(1.84, 0.69, 0.31, 1.83, 145, 60);
  width: 250px;
  height: 250px;
  background-color: #ffeded;
  border:1px solid #420021;
}
```

Fig. 14 Fichier CSS avec la méthode matrix() © Skill and You

Le rendu :

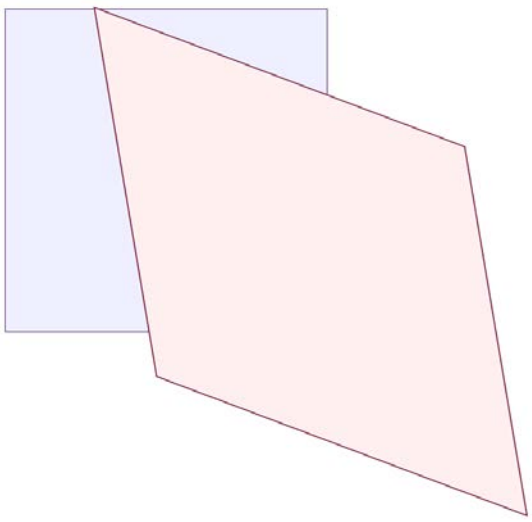

Fig. 15 Affichage dans un navigateur avec la méthode matrix() © Skill and You

G. La propriété transform-origin

La propriété **transform-origin** vous permet de modifier la position d'origine des éléments transformés.

En transformation 2D vous pouvez changer l'axe x et y de l'élément transformé.

En transformation 3D vous pouvez également modifier l'axe z de l'élément transformé.

Remarque : Cette propriété doit être utilisée conjointement avec la propriété transform.

La valeur par défaut :

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

Fig. 16 Valeur par défaut avec la propriété transform-origin © Skill and You

Syntaxe

```
transform-origin: x-axis y-axis z-axis;
```

Fig. 17 Syntaxe avec la propriété transform-origin © Skill and You

Tableau n°1 Propriétés de la transformation

VALEUR DE LA PROPRIÉTÉ	DESCRIPTION
x-axis	Définissant où la vue est placée par rapport à l'axe x. Valeurs possibles : – left – center – right – une valeur en pixels – une valeur en %
y-axis	Définissant où la vue est placée par rapport à l'axe y. Valeurs possibles : – left – center – right – une valeur en pixel – une valeur en %
z-axis	Définissant où la vue est placée par rapport à l'axe z. Ne peut pas être un pourcentage.

On peut ainsi régler le centre de gravité d'un élément en rotation. Par exemple, regardons le CSS suivant :

```
.rotate-origin {
 transform: rotate(45deg);
 transform-origin: center center;
 width: 250px;
 height: 250px;
 background-color: #ffeded;
 border:1px solid #420021;
}
```

Fig. 18 Fichier CSS avec la propriété transform-origin © Skill and You

Qui donne :

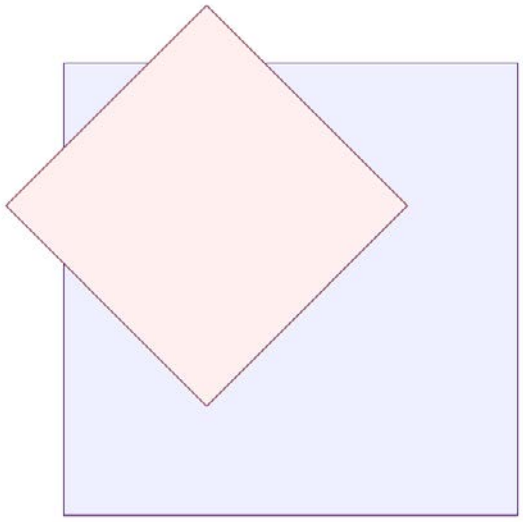

Fig. 19 Affichage dans un navigateur avec la propriété transform-origin © Skill and You

Par défaut on a donc :

```
transform-origin: 50% 50%;
/*ou bien */
transform-origin: center center;
```

Fig. 20 Fichier CSS par défaut © Skill and You

Ce qui veut dire que si l'on n'indique rien, le centre de gravité est au centre de l'élément.

On peut modifier le centre de gravité, la rotation ne se faisant plus par rapport au centre mais par rapport au coin haut de gauche avec :

```
.rotate-origin {
 transform: rotate(45deg);
 transform-origin: top left;
 width: 250px;
 height: 250px;
 background-color: #ffeded;
 border:1px solid #420021;
}
```

Fig. 21 Modification du centre de gravité © Skill and You

Le rendu :

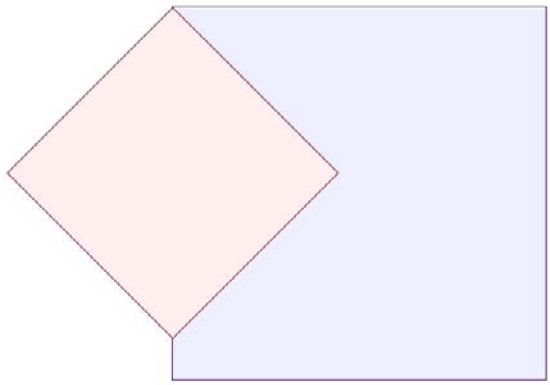

Fig. 22 Affichage dans un navigateur © Skill and You

Si maintenant on modifie le degré de rotation :

```
.rotate-origin {
  transform: rotate(60deg);
  transform-origin: top left;
  width: 250px;
  height: 250px;
  background-color: #ffeded;
  border:1px solid #420021;
}
```

Fig. 23 Modification du degré de rotation © Skill and You

On aura :

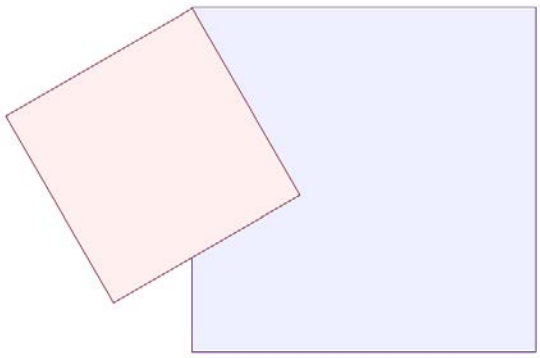

Fig. 24 Affichage dans un navigateur © Skill and You

II. CSS3 : animations

Nous avons regroupé sous le titre animations deux types d'effets qui en réalité ne diffèrent guère l'un de l'autre. Les transitions sont en réalité des animations simplifiées dont la réalisation est réduite au strict minimum.

Pour bien comprendre comment les mettre en œuvre, il faut commencer par la notion fondamentale pour la compréhension d'une animation : l'image-clé. Un mouvement peut se décomposer en série d'images fixes, que l'on nomme en animation les images-clés. Il suffit même de deux images fixes. Je vous renvoie aux travaux de Muybridge et des pionniers du cinéma si vous désirez en savoir plus.

Lorsque vous allez commencer à réaliser des transitions et des animations, demandez-vous toujours quelles sont les images-clés.

A. Transitions

Comment ça marche ?

Les transitions CSS3 sont des effets de transitions de forme et/ou de position.

Pour ce faire, vous devez spécifier deux choses :
- la propriété CSS sur laquelle vous souhaitez ajouter un effet ;
- la durée de l'effet.

Gardez en tête qu'il doit y avoir deux états dans la transition : l'état de départ et l'état d'arrivée (deux images-clés).

B. Transition unique

1. État 1 de la transition

Effet de transition sur la propriété width (largeur), durée : 2 secondes.

C'est ici l'état initial de la transition ou notre première image-clé.

```
.transition-width {
  width:200px;
  transition: width 2s;
}
```

Fig. 25 Tr25 © Skill and You

Remarque : si la durée n'est pas précisée, la transition n'aura pas d'effet, parce que la valeur par défaut est 0.

L'effet va commencer lorsque la propriété CSS spécifiée change de valeur. Un changement de propriété classique s'effectue quand un utilisateur survole ou clique sur un élément :

2. État 2 de la transition

Précisez : hover pour les éléments clés.

C'est ici l'état final de l'animation ou notre deuxième et dernière image-clé :

```
.transition-width:hover {
  width:600px;
}
```

Fig. 26 Tr26 © Skill and You

Remarque : lorsque le curseur de la souris passe sur l'élément, il se transforme progressivement et revient à son style d'origine lorsque la souris n'est plus au-dessus.

C. Transitions multiples

Pour ajouter plusieurs effets de transition, il suffit d'ajouter d'autres propriétés, séparées par des virgules.

Exemple *: ajouter des effets sur la largeur, la hauteur et la transformation.*

```
.transition-multiple {
  transition: width 2s, height 2s, transform 2s;
}
```

Fig. 27 Tr27 © Skill and You

D. Propriétés de transition

Le tableau suivant répertorie toutes les propriétés de la transition :

Tableau n°2 Propriétés de la transition

PROPRIÉTÉ	DESCRIPTION
transition-property	Indique le nom de la propriété CSS à laquelle la transition est appliquée (exemples : width, height, opacity…)
transition-duration	Définit la durée d'une transition en secondes. Valeur par défaut 0.
transition-timing-function	Décrit comment la vitesse lors d'une transition sera calculée. Par défaut « ease ».
transition-delay	Définit à partir de quel moment la transition va commencer (en secondes). Valeur par défaut 0.

Tableau n°3 Valeurs de transition-timing-function

VALEUR	DESCRIPTION
linear	Indique un effet de transition avec la même vitesse du début à la fin (l'équivalent de cubes de Bézier (0,0,1,1)).
ease	Indique un effet de transition avec un démarrage lent, puis rapide, puis terminer lentement (l'équivalent de cubes de Bézier (0.25,0.1,0.25,1)).
ease-in	Indique un effet de transition avec un démarrage lent (l'équivalent de cubes de Bézier (0.42,0,1,1)).

VALEUR	DESCRIPTION
ease-out	Indique un effet de transition avec une fin lente (l'équivalent de cubes de Bézier (0,0,0.58,1)).
ease-in-out	Indique un effet de transition avec un démarrage lent et à la fin (l'équivalent de cubes de Bézier (0.42,0,0.58,1)).
cubic-bezier(n,n,n,n)	Définissez vos propres valeurs dans la fonction cube-Bézier. Les valeurs possibles sont les valeurs numériques de 0 à 1. Voir le site : https://www.netzgesta.de/dev/cubic-bezier-timing-function.html

Exemple :

```
.transition-property {
  transition-property: width;
  transition-duration: 3s;
  transition-timing-function: linear
  transition-delay: 1s;
}
```

Fig. 28 Tr28 © Skill and You

Une autre façon plus simple pour le même rendu :

```
.transition-property-quick {
  transition: width 3s linear 1s;
}
```

Fig. 29 Tr29 © Skill and You

III. Animations

Avec CSS3, nous pouvons créer des animations, qui peuvent remplacer les images animées, animations Flash, et les animations créées avec du JavaScript.

A. La règle @keyframes

La règle @keyframes (images-clés) est l'endroit où l'animation est créée. Spécifiez un style CSS à l'intérieur de la règle @keyframes et l'animation va progressivement changer du style d'origine pour le nouveau style.

Cas pratique : réalisation d'une animation nommée myfirst.

Soit le HTML suivant :

```
<div class="parent">
    <div class="bloc animation">
    </div>
</div>
```

Fig. 30 Tr30 © Skill and You

On a isolé l'animation proprement dite dans une classe et le style du bloc dans la classe **.bloc** qui nous donne avant de styliser l'animation :

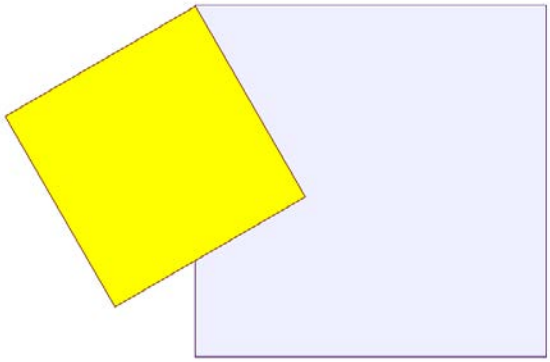

Fig. 31 Tr31 © Skill and You

Ensuite on crée l'animation avec @keyframes. On va simplement modifier la couleur de fond et changer le degré de rotation du bloc (le centre de gravité – **transform-origin** est réglé sur un coin afin qu'il pivote autour de celui-ci) :

```
@keyframes myfirst
{
  from {
    background: red;
    transform: rotate(60deg);
  }
  to {
    background: yellow;
    transform: rotate(30deg);
  }
}
```

Fig. 32 Tr32 © Skill and You

Lorsque l'animation est créée dans l'image-clé @, il faut la lier à un sélecteur, sinon l'animation n'aura aucun effet.

Liez l'animation à un sélecteur en précisant au moins ces deux propriétés CSS3 animation :
– nom de l'animation ;
– durée de l'animation.

B. Liaison de l'animation « myfirst » à un élément div, durée : 5 secondes

```
.animation  {
  animation: myfirst 5s,
}
```

Fig. 33 Tr33 © Skill and You

Rappel : Vous devez définir le nom et la durée de l'animation. Si la durée est omise, l'animation ne fonctionne pas, parce que la valeur par défaut est 0.

Le rendu :

Origine (voir le rendu plus haut)

Au bout de 2 secondes :

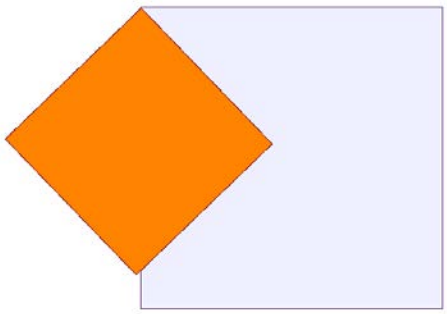

Fig. 34 Tr34 © Skill and You

À la fin de l'animation :

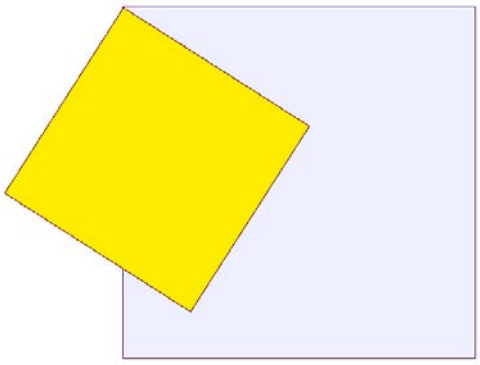

Fig. 35 Tr35 © Skill and You

C. Animation pas à pas

Vous pouvez changer autant de styles que vous voulez, autant de fois que vous voulez.

Indiquez quand le changement se produira en pourcentage, ou les mots-clés « **from** » et « **to** » (à), ce qui est la même chose que 0 % et 100 %.

0 % est le début de l'animation, 100 % quand l'animation est terminée.

Changer la couleur de fond et la rotation lorsque l'animation est à 25 %, 50 %, et de nouveau lorsque l'animation est terminée à 100 % :

```
@keyframes stepByStep
{
  0% {
    background: yellow;
    transform: rotate(60deg);
  }
  25% {
    background: blue;
    transform: rotate(0deg);
  }
  50% {
    background: red;
    transform: rotate(-60deg);
  }
  100% {
    background: green;
    transform: rotate(-120deg);
  }
}
```

Fig. 36 Tr36 © Skill and You

D. Propriétés d'animation

Tableau n°4 Propriétés d'animation

PROPRIÉTÉ	DESCRIPTION	CSS
@keyframes	Spécifie l'animation.	3
animation	Un raccourci pour toutes les propriétés de l'animation, à l'exception de la propriété « animation-play-state ».	3
animation-name	Indique le nom de l'animation d'images clés @.	3
animation-duration	Indique combien de secondes ou millisecondes sont nécessaires pour qu'une animation termine un cycle. Valeur par défaut 0.	3
animation-timing-function	Décrit le type de transition entre deux états. Par défaut « ease ».	3
animation-delay	Indique le moment où l'animation démarre. Valeur par défaut 0.	3
animation-iteration-count	Indique le nombre de fois qu'une animation est jouée. Par défaut 1 et si elle est jouée à l'infini : infinite.	3
animation-direction	Indique si oui ou non l'animation devrait jouer en sens inverse (alternate) sur des cycles alternés. Par défaut « normal ».	3
animation-play-state	Indique si l'animation est en cours d'exécution (running) ou en pause (paused). Par défaut « en cours d'exécution » (running).	3

Les deux exemples ci-dessous définissent de la même façon mais avec une syntaxe différente toutes les propriétés de l'animation.

E. Animation multi-propriétés

Exécuter une animation appelée myfirst définie plus haut, avec l'ensemble des propriétés :

```
.animation {
  animation-name: stepByStep;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 1s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
  animation-play-state: running;
}
```

Fig. 37 Tr37 © Skill and You

La même animation comme ci-dessus, en utilisant la propriété d'animation raccourcie :

```
.animation {
  animation: stepByStep 1s linear 0s infinite alternate running;
}
```

Fig. 38 Tr38 © Skill and You

IV. Conclusion

Nous avons exploré dans cette leçon ce qui permet au CSS, grâce aux transformations, transitions et animations, de concurrencer ce que l'on trouvait auparavant dans un logiciel tel que Flash.