













Transformations, transitions et animations

Sommaire

I. Transformation 2D	3	III. Animations	17
A. Les méthodes de 2D Transforms	3	A. La règle @keyframes	17
B. La méthode translate ()	3	B. Liaison de la « myfirst » animation à un élément div, durée : 5 secondes	18
C. La méthode rotate ()	5	C. Animation pas à pas	19
D. La méthode scale ()	6	D. Propriétés d'animation	20
E. La méthode skew ()	7	E. Animation multi- propriétés	21
F. La méthode matrix()	9	IV. Conclusion	21
G. La propriété transform-origin	10		
II. CSS3 : Animations	14		
A. Transitions	14		
B. Transition unique	14		
C. Transitions multiples	15		
D. Propriétés de transition	15		

Crédits des illustrations: © Skill&You

Les repères de lecture

 Retour au chapitre	 Définition	 Objectif(s)	 Espace Élèves	 Vidéo / Audio
 Point important / À retenir	 Remarque(s)	 Pour aller plus loin	 Normes et lois	 Quiz

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 Transforms

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

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

Le rendu :

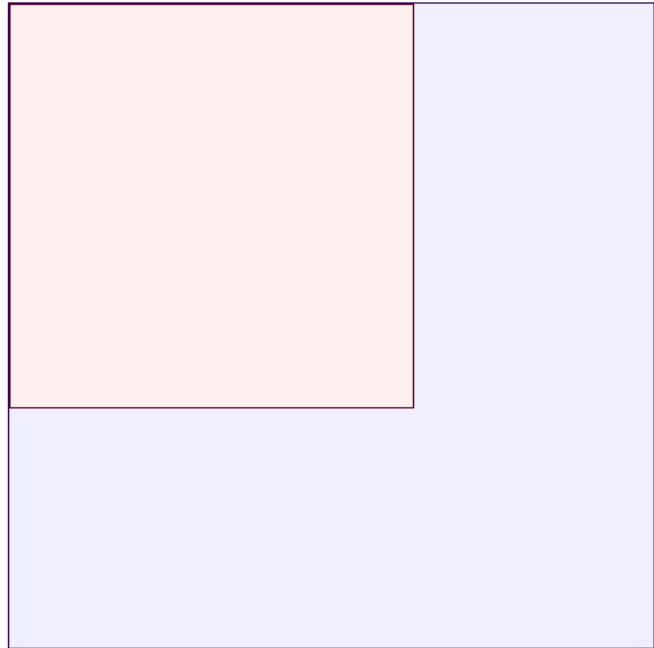


Fig.3

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

Ce qui nous donne :

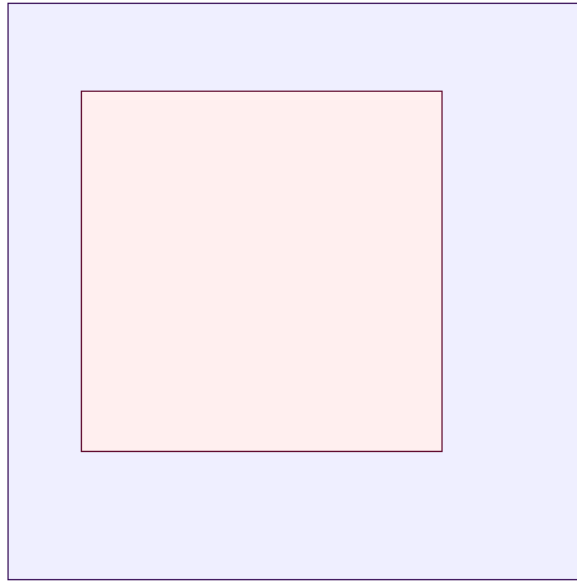


Fig. 5

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

On obtient :

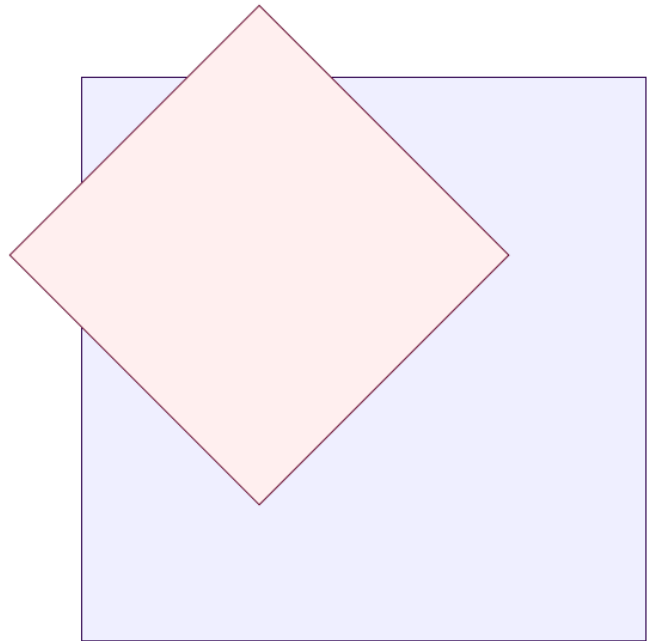


Fig.7

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

On obtient :

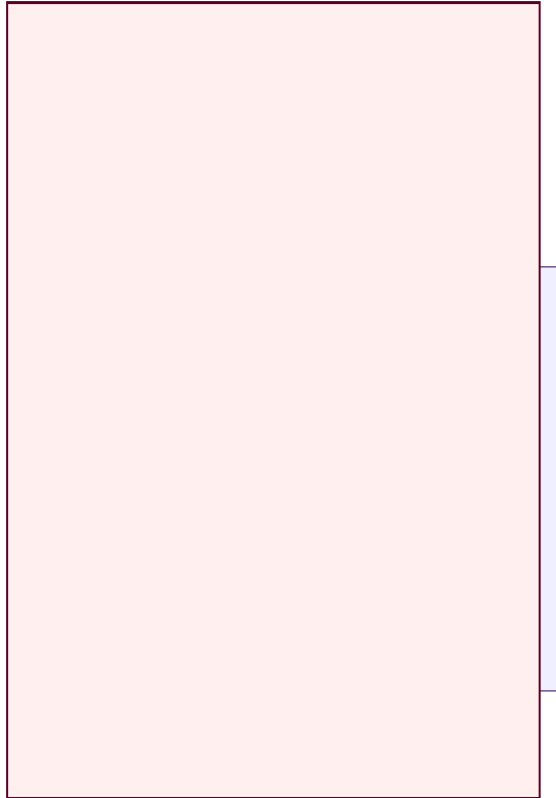


Fig.9

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

Donne :

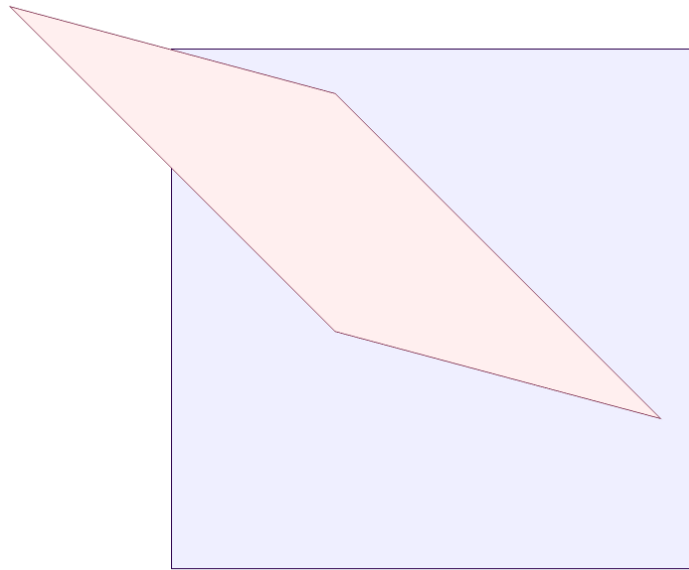


Fig.11

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 transformation en concaténant les différentes valeurs des différentes méthodes **transform**, 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

Donne :

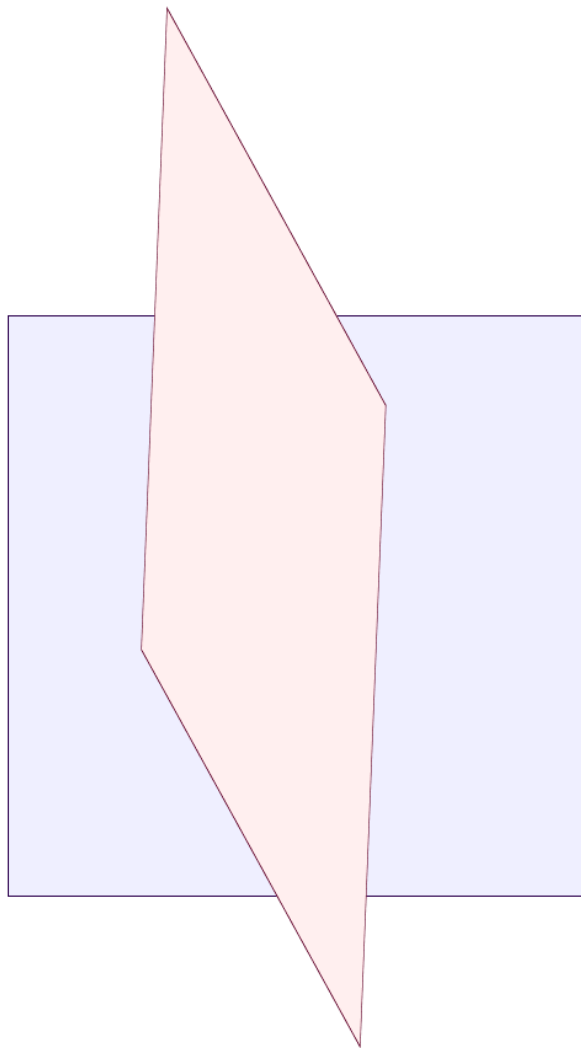


Fig. 13

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.

```
transform:matrix(a,b,c,d,e,f);
```

Nous n'entrerons pas dans les détails complexes de cette matrice de transformation. Le plus simple est alors de se rendre sur cette page :

<http://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

Le rendu :

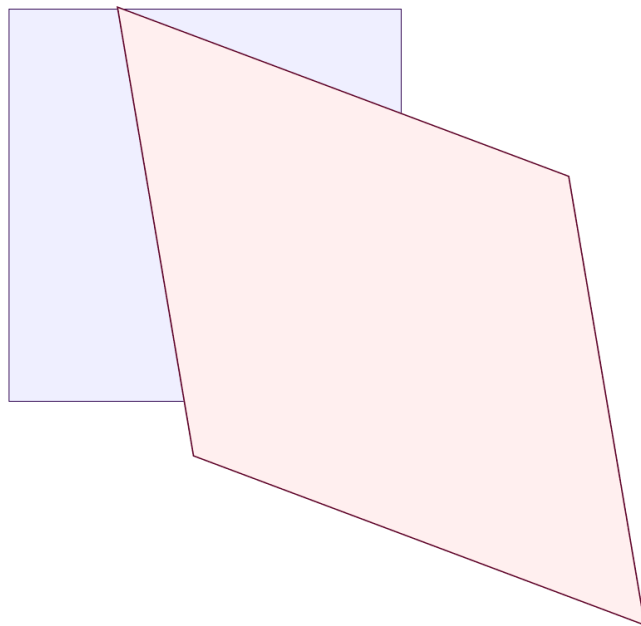


Fig. 15

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 des 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

Syntaxe

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

Fig. 17

Tableau n°1

VALEUR DE LA PROPRIÉTÉ	DESCRIPTION
x-axis	Définissant où la vue est placée par rapport à l'axe des x. Valeurs possibles : <ul style="list-style-type: none">– left– center– right– une valeur en pixel– une valeur en %
y-axis	Définissant où la vue est placée par rapport à l'axe des y. Valeurs possibles : <ul style="list-style-type: none">– 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

Qui donne :

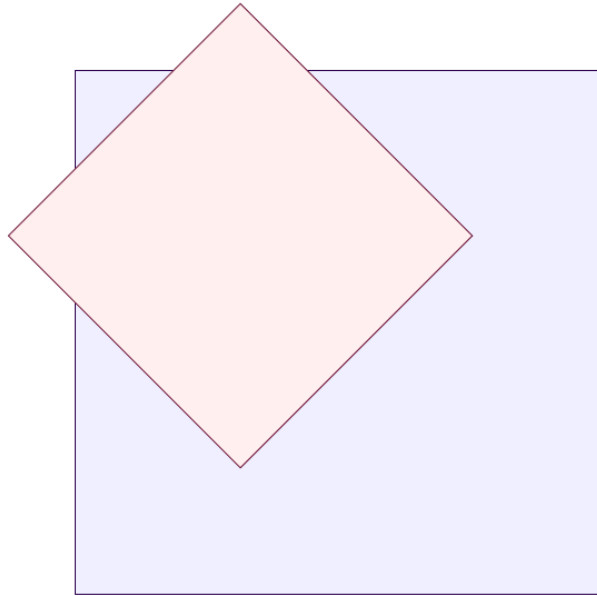


Fig. 19

Par défaut on a donc :

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

Fig. 20

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

Le rendu :

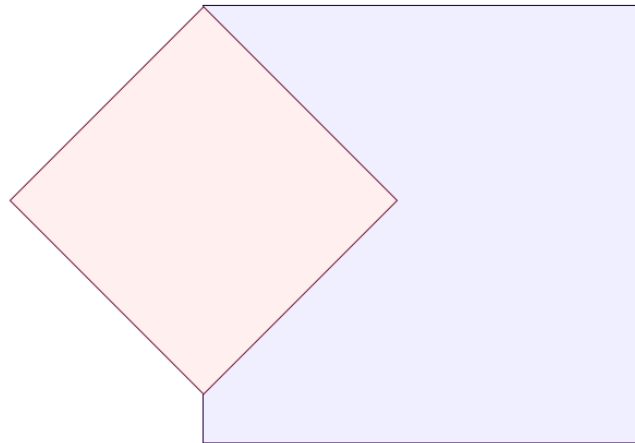


Fig.22

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

On aura :

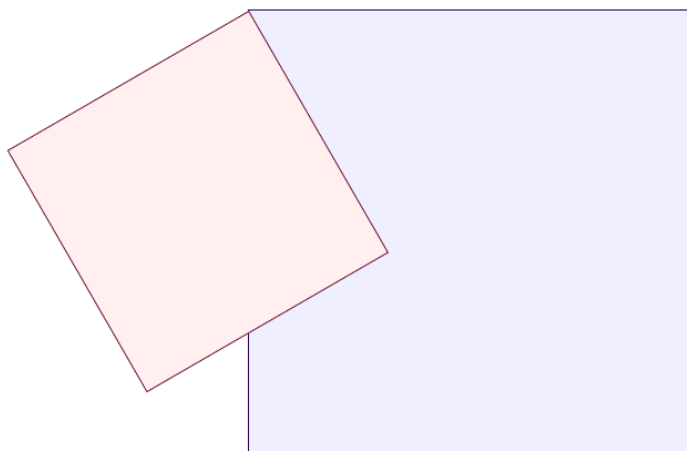


Fig.24

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

1. 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'arrivé (deux images-clés).

B. Transition unique

1. Etat 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

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. Etat 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

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

Ajoutez des effets sur la largeur, la hauteur et la transformation :

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

Fig.27

D. Propriétés de transition

Le tableau suivant répertorie toutes les propriétés de la transition :

Tableau n°2

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)).
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 : http://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

Une autre façon plus simple pour le même rendu.

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

Fig.29

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

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 :

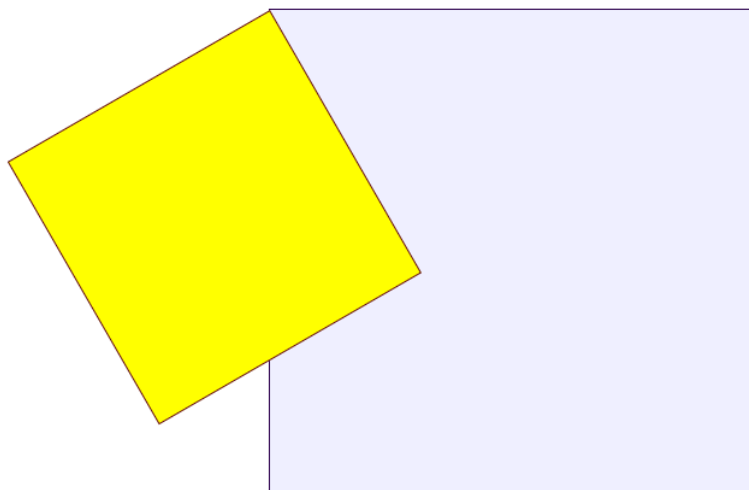


Fig. 31

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

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 :

1. Indiquez le nom de l'animation ;
2. Indiquez la durée de l'animation.

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

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

Fig.33

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 :

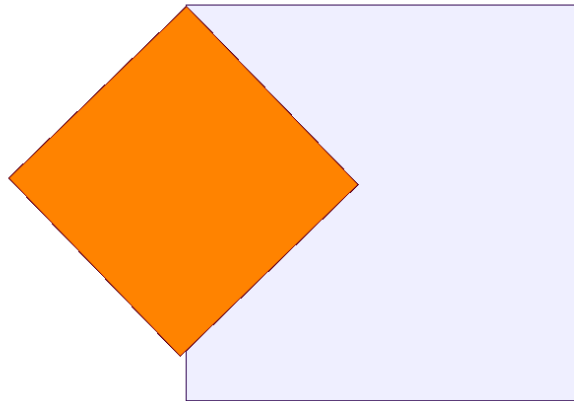


Fig.34

A la fin de l'animation :

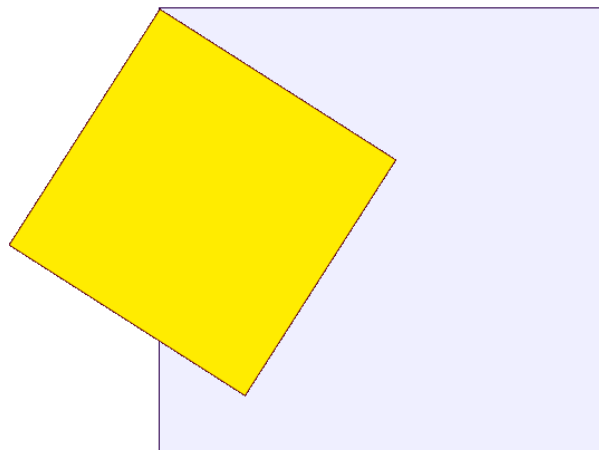


Fig.35

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 pourcent, ou les mots-clés « from (de) » 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

D. Propriétés d'animation

Tableau n°4

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

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

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

Fig.38

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.