

## Semaine 01 - Révision de quelques notions des cours 1W1 et 1J1

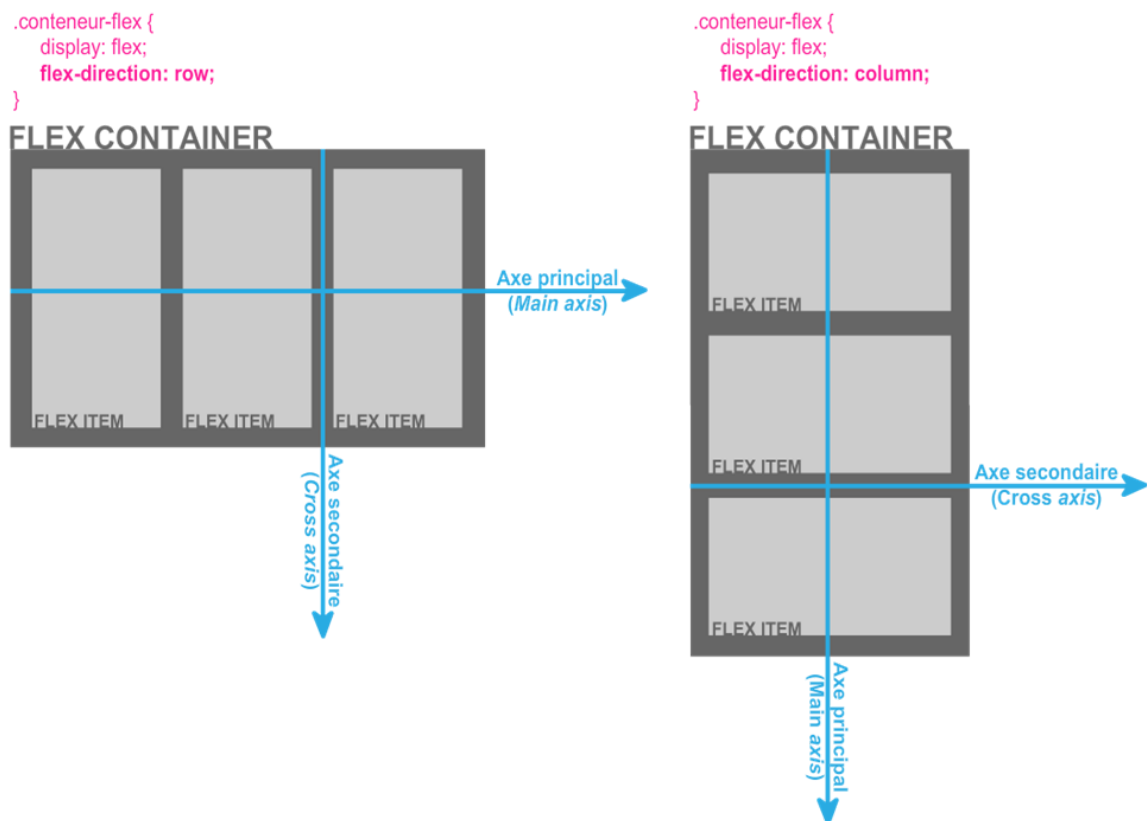
### La mise en page Web avec les boîtes flexibles (CSS Flexbox)

Flexbox (pour flexible box) est un mode de mise en page prévoyant une disposition des éléments d'une page de telle sorte que ses éléments possèdent un **comportement prévisible** lorsqu'ils doivent **s'accommoder de différentes tailles d'écrans/appareils**.

Flexbox diffère un peu des types d'affichage habituels car le changement va ici non seulement **impacter l'élément sélectionné**, mais également **ses enfants directs**. Ainsi on parlera de **conteneur** (flex-container) et d'**éléments de contenu** (flex-items) et il est possible de passer des **propriétés spécifiques à ces deux composants**. Il est tout à fait possible aussi qu'un élément soit à la fois un *flex-item* et un *flex-container* : autrement dit, les **imbrications flexbox sont possibles**.

Le contenu de ce chapitre n'est qu'un **mémo** des principales propriétés d'un *flex-container* et des *flex-items*. Vous pouvez consulter les sites donnés en référence pour obtenir plus d'informations sur le sujet.

Reportez-vous aux diagrammes suivants lorsque vous lirez le vocabulaire des différents éléments listés dans le tableau ci-dessous. Ils représentent deux conteneurs flex. Le premier possède la propriété *flex-direction* à *row*, signifiant que tous les éléments flex vont se suivre horizontalement (en ligne) au travers de l'axe principal. Le deuxième possède la propriété *flex-direction* à *column*, signifiant que tous les éléments flex vont se suivre verticalement (en colonne) au travers de l'axe principal.



## Mémo des propriétés d'un conteneur FlexBox (flex-container)

Propriété	Définition et valeurs possibles	Valeur initiale
display	Type d'affichage d'un conteneur Flexbox <a href="#">flex</a>   <a href="#">inline-flex</a>	Selon l'élément HTML
flex-direction	Direction de l'axe principal <a href="#">row</a>   <a href="#">row-reverse</a>   <a href="#">column</a>   <a href="#">column-reverse</a>	<a href="#">row</a>
flex-wrap	Gestion des débordements des flex-items dans les lignes ou dans les colonnes  <a href="#">nowrap</a>   <a href="#">wrap</a>   <a href="#">wrap-reverse</a>	<a href="#">nowrap</a>
flex-flow	Propriété raccourcie cf. flex-direction et flex-wrap	<a href="#">row nowrap</a>
justify-content	<b>Alignement</b> des items sur l'axe principal <a href="#">flex-start</a>   <a href="#">flex-end</a>   <a href="#">center</a>   <a href="#">space-between</a>   <a href="#">space-around</a>   <a href="#">space-evenly</a>	<a href="#">flex-start</a>
align-items	<b>Alignement</b> des items sur l'axe secondaire <a href="#">flex-start</a>   <a href="#">flex-end</a>   <a href="#">center</a>   <a href="#">baseline</a>   <a href="#">stretch</a>	<a href="#">stretch</a>
align-content	<b>Alignement</b> des lignes ou des colonnes d'un conteneur Flexbox sur l'axe <b>secondaire</b> (s'applique à un <i>flex-container</i> <b>multi-lignes</b> ou <b>multi-colonnes</b> ) <a href="#">flex-start</a>   <a href="#">flex-end</a>   <a href="#">center</a>   <a href="#">space-between</a>   <a href="#">space-around</a>   <a href="#">stretch</a>	<a href="#">stretch</a>

## Mémo des propriétés des items Flexbox (items-flex)

Propriété	Définition et valeurs possibles	Valeur initiale
flex-grow	Possibilité de s'étirer (par défaut NON) <a href="#">Nombre positif</a>	<a href="#">0</a>
flex-shrink	Possibilité de rétrécir (par défaut OUI) <a href="#">Nombre positif</a>	<a href="#">1</a>
flex-basis	Largeur de référence de l'item <a href="#">auto (largeur réelle du contenu)</a>   <a href="#">%</a>   <a href="#">px</a> , <a href="#">vh</a> , <a href="#">vw</a> , etc. ou autre unité valide de longueur	<a href="#">auto</a>
order	Ordre d'affichage de l'item dans le flux <a href="#">Nombre entier positif ou négatif</a>	<a href="#">0</a>
flex	Propriété raccourcie cf. flex-grow, flex-shrink et flex-basis	<a href="#">0 1 auto</a>
align-self	<b>Alignement individuel</b> d'un item sur l'axe secondaire (Passe outre la valeur de la propriété align-items spécifiée pour les enfants du conteneur Flexbox). <a href="#">auto</a>   <a href="#">flex-start</a>   <a href="#">flex-end</a>   <a href="#">center</a>   <a href="#">baseline</a>   <a href="#">stretch</a>	<a href="#">auto</a>

## Quelques références utiles sur les Flexbox

### Site officiel du W3C:

- Spécifications officielles du W3C, CSS Flexible Box Layout Module Level 1 (Novembre 2018). Repéré à : <http://www.w3.org/TR/css-flexbox-1/>

### Bonnes références en français

- CSS3 Flexbox Layout module d'AlsaCréations (Avril 2020). Repéré à : <http://www.alsacreations.com/tuto/lire/1493-css3-flexbox-layout-module.html>
- Faites votre mise en page avec Flexbox d'OpenClassrooms (Janvier 2023). Repéré à : <https://openclassrooms.com/fr/courses/1603881-creez-votre-site-web-avec-html5-et-css3/8061384-faites-votre-mise-en-page-avec-flexbox>

### Bonnes références en anglais

- A Complete Guide to Flexbox | CSS-Tricks (Décembre 2022). repéré à : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- Flexbox - CSS Reference (s.-d.). repéré à : <https://cssreference.io/flexbox/>

### Environnement Web pour tester les propriétés Flexbox en direct :

- <http://demo.agektmr.com/flexbox/>

### Petit jeu sur le Web pour jouer avec les propriétés Flexbox :

- <https://flexboxfroggy.com/#fr>

## Utilisation des transformations 2D en CSS

En modifiant l'espace des coordonnées, les transformations CSS permettent de changer la position d'un contenu affecté **sans perturber le flux normal du document HTML**. Elles sont implémentées en utilisant un ensemble de propriétés CSS qui permettent d'appliquer des transformations affinées aux éléments HTML. Ces transformations incluent la rotation, l'inclinaison et la translation à la fois dans un espace 2D ou dans un espace 3D (pas couvert ici...).

Deux propriétés majeures sont utilisées pour définir les transformations 2D avec CSS : **transform-origin** et **transform**.

### Propriété CSS : 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. Elle est utilisée par de nombreuses transformations, comme les rotations, les mises à l'échelle ou les inclinaisons, qui nécessitent un point spécifique pour paramètre.

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

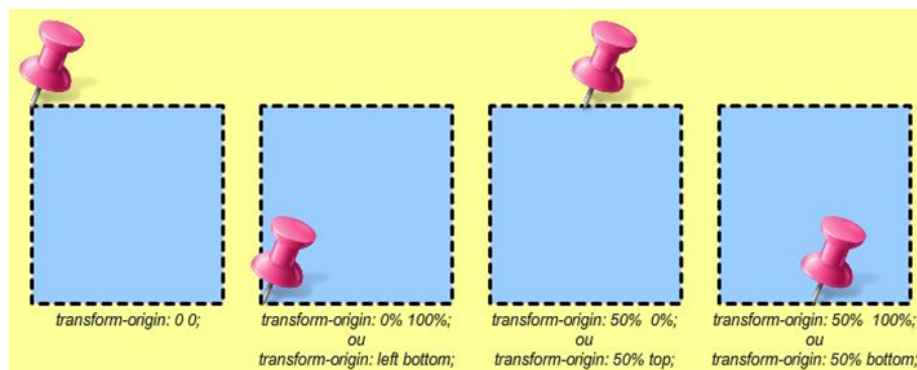
```
transform-origin: 50% 50%;
```



Il est possible de changer cette valeur en utilisant un mot-clé de position (top, right, bottom, left, etc.) ou une valeur chiffrée dont l'unité peut varier (px, %, etc.). Il est à noter que cette propriété accepte une ou deux valeurs :

- si **une seule valeur** est indiquée, cela définit la position horizontale, la position verticale est alors fixée à 50 %;
- si **deux valeurs** sont définies, la première définit la position horizontale puis la seconde, la position verticale.

Voici quelques exemples de positionnement:



## Propriété CSS:transform

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 de fonctions 2D**, qui sont les suivantes:

Translation (Déplacement sur l'axe des X ou des Y)	
<code>translate(tX,tY)</code>	Applique un déplacement sur l'axe des X et des Y.
<code>translateX(tX)</code>	Applique un déplacement sur l'axe des X uniquement.
<code>translateY(tY)</code>	Applique un déplacement sur l'axe des Y uniquement.
Mise à l'échelle	
<code>scale(sX, sY)</code>	Applique une modification de l'échelle de l'élément, soit un agrandissement ou une réduction sur les axes X et Y.
<code>scaleX(sX)</code>	Applique une modification de l'échelle de l'élément uniquement sur l'axe X.
<code>scaleY(sY)</code>	Applique une modification de l'échelle de l'élément uniquement sur l'axe Y.
Rotation	
<code>rotate(angle)</code>	Applique une rotation dans le sens horaire autour de son origine (voir plus loin la propriété <code>transform-origin</code> ) selon l'angle spécifié.
Inclinaison	
<code>skew(angleX, angleY)</code>	Incline l'élément autour des axes X et Y selon les angles spécifiés.
<code>skewX(angleX)</code>	Incline l'élément autour de l'axe X selon l'angle spécifié.
<code>skewY(angleY)</code>	Incline l'élément autour de l'axe Y selon l'angle spécifié.
Matrice de transformation	
<code>matrix(n,n,n,n,n,n)</code>	Cette fonction permet de réunir en une seule déclaration toutes les fonctions précédentes sous la forme d'une matrice. <i>Très concrètement, les détails ne seront pas abordés ici car ils font beaucoup plus appel à des <b>notions mathématiques complexes</b> que beaucoup d'entre nous ont mis de côté...</i>

## Les fonctions de la propriété transform

Une fois l'origine choisie avec la propriété **transform-origin**, nous pouvons affecter des transformations à nos éléments HTML avec les différentes fonctions de la propriété **transform**.

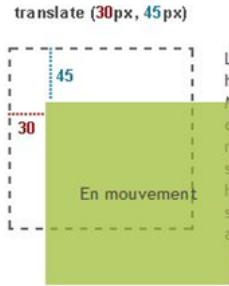
### La fonction translate

Cette fonction permet d'effectuer un déplacement de l'élément HTML sur les axes X et Y. La syntaxe est la suivante :

**transform: translate(tX [, tY]) /\* une ou deux <longueurs> (<length> \*/**

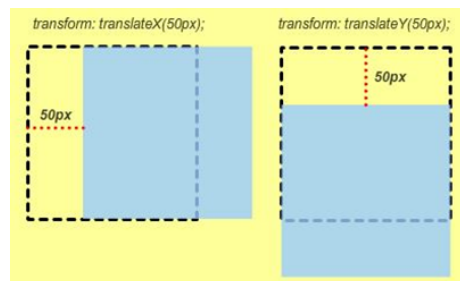
Si **tY** n'est pas spécifié, la valeur supposée est **zéro**. Les unités de valeurs peuvent être exprimées en absolu (px, pt, cm, etc.) ou en relatif (em, rem, vw, vh, etc.). **IMPORTANT!** Si les valeurs sont exprimées en **pourcentage (%)**, la

valeur réelle du déplacement est calculée en fonction de la taille de l'élément HTML et **non** en fonction de la taille de son conteneur parent. Par exemple, pour un élément HTML dont la largeur serait de 150px, une translation sur l'axe des X de 50 % équivaldrait à un déplacement de 75px (50 % de 150px)...

	<p><b>ATTENTION! Ici, il n'y a aucune notion de flux, l'élément part de son emplacement courant, quel que soit le type de positionnement qu'on lui a attribué initialement et la position des autres éléments n'est pas affectée.</b></p>
---	---

## Les fonctions translateX et translateY

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

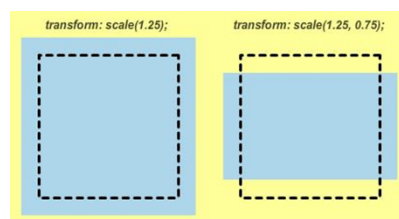


## La fonction scale

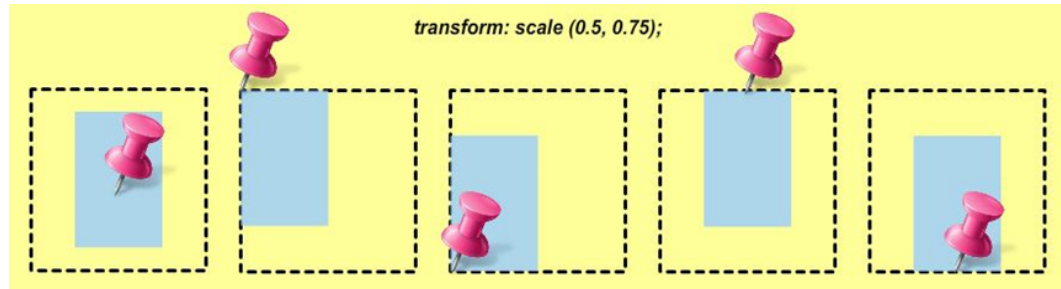
Cette fonction permet d'agir sur **l'échelle** (les dimensions) de l'élément. La valeur initiale est 1, tandis que les valeurs supérieures à 1 créent un effet d'agrandissement, et les valeurs inférieures, créent un effet de réduction. La syntaxe est la suivante:

**transform: scale(sX[, sY]);** /\* un ou deux <nombre à virgule flottante> sans unité\*/

La valeur **sY** est **optionnelle** et sera égale à la valeur de **sX**, si elle n'est pas renseignée.



Attention cependant, le **point d'origine de la transformation** (*transform-origin*) a son importance comme le présente le graphique qui suit :



## 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 d'échelle sur l'axe X et sur l'axe Y.

**transform: scaleX(sX);** /\* un <nombre à virgule flottante > sans unité\*/

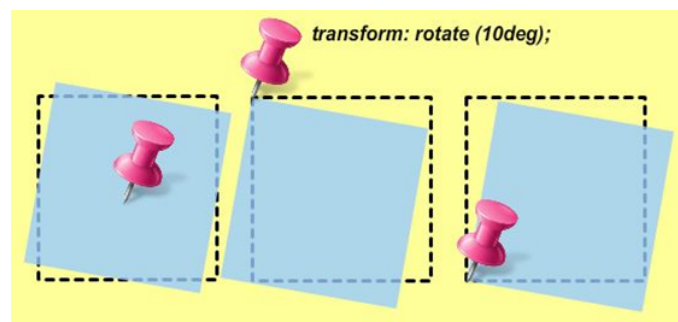
**transform: scaleY(sY);** /\* un <nombre à virgule flottante > sans unité\*/

## La fonction rotate

Comme son nom l'indique, cette fonction permet d'effectuer une **rotation** de l'élément HTML ciblé. La rotation est appliquée dans le sens horaire autour de son origine (comme spécifiée par la propriété *transform-origin*) de l'angle spécifié. Sa syntaxe est la suivante:

**transform: rotate(angle);** /\* un <angle>, p. ex. rotate(30deg) \*/

Attention, ici également, le **point d'origine de la transformation** (*transform-origin*) a son importance, comme le présente le graphique qui suit:

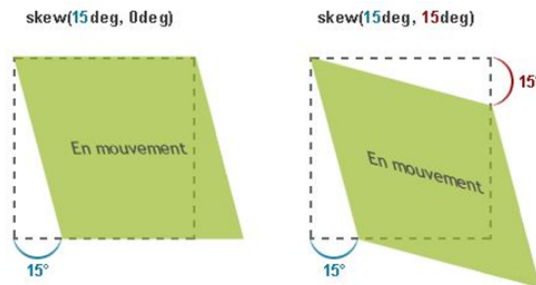


## La fonction skew

Cette fonction permet d'incliner l'élément HTML autour des axes X et Y selon les angles spécifiés. La syntaxe est la suivante:

**transform: skew(angleX[, angleY])** /\* un ou deux <angles> \*/

La valeur **angleY** est **facultative**. Si elle n'est pas spécifiée, aucune inclinaison n'est réalisée suivant l'axe Y.



## Les fonctions skewX et skewY

Sur le même principe que pour les fonctions dérivées de « translate », ces deux fonctions permettent de définir indépendamment des valeurs d'inclinaison l'axe X et sur l'axe Y :

**transform: skewX(angle) /\* un <angle>, p. ex. skewX(-30deg) \*/**

**transform: skewY(angle) /\* un <angle>, p. ex. skewY(-30deg) \*/**

## Ordre des déclarations des fonctions de transformation

La propriété **transform** accepte plusieurs fonctions les unes à la suite des autres. Pour cela, il faut séparer les différentes fonctions par un espace, comme l'illustre cet exemple:

**transform: scale(2) translate(20rem, 20rem) rotate(20deg);**

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(20rem, 20rem);  
}
```

est différent de :

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

En effet, les **transformations se font dans l'ordre où elles sont déclarées**, et donc l'effet final peut varier. Ainsi une translation de 20rem est équivalente à 40rem si un scale de 2 la précède. **Vous me suivez ? Attention donc à l'ordre de déclaration de ces fonctions.**



## Quelques références utiles sur les transformations CSS

### Site officiel du W3C:

- Spécifications officielles du W3C. (Février 2019). CSS Transforms Module Level 1. Repéré à : <http://www.w3.org/TR/css-transforms-1/>

### Bonnes références en français

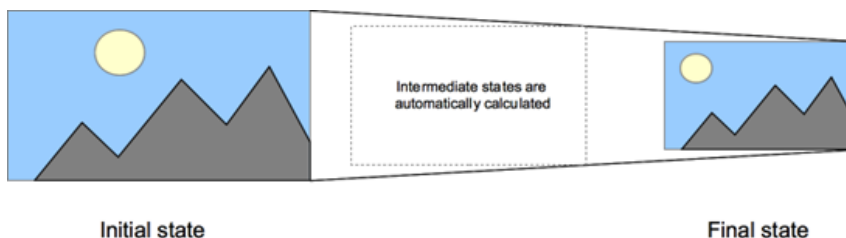
- AlsaCréations. (Février 2015). CSS3 : Transformations 2D. Repéré à : <http://www.alsacreations.com/article/lire/1418-css3-transformations-2d.html>
- Mozilla Corporation. (Octobre 2022). Transform- CSS | MDN.. Repéré à : <https://developer.mozilla.org/fr/docs/Web/CSS/transform>

### Bonne référence en anglais

- W3Schools. (s.d.). CSS3 2D Transforms. Repéré à : [http://www.w3schools.com/css/css3\\_2dtransforms.asp](http://www.w3schools.com/css/css3_2dtransforms.asp)

## Utilisation des transitions CSS

Les **transitions CSS** permettent de **modifier les valeurs de propriétés CSS** dans le **temps**, d'un **état initial A** à un **état final B** souhaité. Par exemple, il est possible de modifier la taille d'un élément progressivement d'une valeur de 200px à 100px, au lieu que les changements entre les deux valeurs de cette propriété prennent effet instantanément.



Source : [https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using\\_CSS\\_transitions](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_transitions) (Page consultée le 20 janvier 2023)

## Sous-propriétés des transitions CSS

Les transitions en CSS se décomposent en plusieurs sous-propriétés, décrites sommairement ici :

- **transition-property** : la(les) propriété(s) CSS à modifier.
- **transition-duration** : le temps total de la transition ( en secondes, millisecondes).
- **transition-timing-function** : la méthode d'interpolation (accélération, décélération)
- **transition-delay** : le temps avant que la transition ne démarre.

Pour définir un effet de transition, il n'est pas nécessaire d'utiliser toutes ces quatre sous-propriétés. Les **deux premières** (transition-property et transition-duration) sont **obligatoires** tandis que les deux autres (transition-timing-function, transition-delay) sont **facultatifs**. À noter que la sous-propriété *transition-timing-function* n'est pas abordée dans cette note de cours.

## Définir des transitions simples

Pour définir des **transitions simples**, il suffit d'utiliser conjointement les **deux sous-propriétés** : **transition-property** et **transition-duration**.

### transition-property

La sous-propriété **transition-property** permet de définir **la ou les propriétés CSS à animer**. Seules les propriétés énumérées seront animées lors des transitions et, celles-ci, sauf exception, **doivent d'abord** être définies dans la feuille de style de l'élément HTML concerné. Le code CSS qui suit indique que la propriété CSS «opacity» sera éventuellement animée pour les éléments « div » d'une page HTML donnée :

```
/*Effet de transition d'une propriété CSS*/
div{
  transition-property: opacity;
}
```

Pour le même élément HTML, il est possible de faire des effets de transitions sur plusieurs propriétés CSS. Il suffit de séparer les noms de propriétés par des virgules (,). Exemple :

```
/*Effet de transition de plusieurs propriété CSS*/
div{
    transition-property: opacity, border, color, background-color;
}
```

Il est à noter, que lorsque l'on souhaite effectuer des transitions sur plusieurs propriétés CSS, il est possible d'utiliser la valeur « **all** » (transition-property: all ;). **Pour des questions d'optimisation, l'usage de ce mot-clé n'est toutefois PAS recommandé.**

## transition-duration

La sous-propriété transition-duration permet de préciser la durée de la ou des transition(s). Si plusieurs propriétés ont été précisées à l'aide de la sous-propriété précédente, il est possible de préciser plusieurs valeurs pour cette sous-propriété en les séparant également d'une virgule. Les valeurs acceptées sont des valeurs de temps. Une valeur de temps est donnée par un nombre suivi d'une unité de temps. Les deux unités de temps définies en CSS sont :

- **s** : seconde
- **ms** : milliseconde

Pour préciser une durée de 5 secondes pour une transition, nous pouvons alors utiliser la déclaration suivante :

```
/*Effet de transition d'une propriété CSS*/
div{
    transition-property: opacity;
    transition-duration: 5s;
}
```

Lorsque plusieurs propriétés CSS sont animées, il est possible de définir une ou plusieurs valeurs de temps. Dans l'exemple qui suit, toutes les propriétés CSS déclarées seraient animées sur 3 secondes :

```
/*Effet de transition de plusieurs propriété CSS*/
div{
    transition-property: opacity, border, color, background-color;
    transition-duration: 3s;
}
```

Dans cet autre exemple, les propriétés « opacity et border » seraient animées sur 3 secondes, tandis que les deux autres seraient animées sur 2 secondes :

```
/*Effet de transition de plusieurs propriété CSS*/
div{
    transition-property: opacity, border, color, background-color;
    transition-duration: 3s, 2s;
}
```

Ainsi, dans l'exemple ci-haut, le navigateur interpréterait les valeurs de durée comme suit :

**transition-duration : 3s, 2s, 3s, 2s;**

À elles seules, les sous propriétés `transition-property` et `transition-duration` ne permettent pas de déclencher un effet de transition. Pour que l'effet de transition se réalise il faut modifier les valeurs des propriétés avec l'une ou l'autre des façons suivantes :

- soit via une **pseudo-classe CSS**, telles que `:hover`, `:focus` ou `:active`;
- soit via **JavaScript** (ce que nous verrons plus tard dans la session...).

**L'effet de transition débutera donc dès que la propriété CSS spécifiée changera de valeur.**

Le code final suivant permettrait d'animer deux propriétés CSS d'un élément HTML « `div` » au survol de ce dernier par le pointeur de la souris :

```
/*Effet de transition de plusieurs propriété CSS*/
div{
  transition-property: opacity, color;
  transition-duration: 2s;
}

div:hover{
  opacity:1;
  color:red;
}
```

## transition-delay

La sous-propriété **transition-delay** permet de définir un délai de déclenchement. Par défaut, une transition commence dès que la valeur d'une propriété est changée, suite à un événement. La sous-propriété **transition-delay** permet d'adapter ce comportement en **retardant** le début de la transition. Lorsqu'une valeur positive est donnée, le démarrage de la transition est retardé. Par exemple :

```
/*Effet de transition de plusieurs propriété CSS*/
div{
  transition-property: opacity, color;
  transition-duration: 2s;
  transition-delay: 1s;
}

div:hover{
  opacity:1;
  color:red;
}
```

La définition ci-haut fait en sorte que la **transition démarre une seconde après** le changement de la valeur.

## La notation raccourcie : transition

Tout comme pour d'autres propriétés CSS, il existe une **notation raccourcie** pour déclarer les transitions. Cette notation permet de décrire facilement et de manière concise les différentes propriétés en jeu à l'aide d'une seule propriété. La syntaxe est la suivante :

```
sélecteurCSS {  
    transition: <property> <duration> <timing-function> <délai>;  
}
```

Comme précédemment, il est possible de préciser plusieurs transitions à l'aide de la propriété en séparant les déclarations par des virgules. Par exemple :

```
sélecteurCSS {  
    transition: opacity 2s 1s, color 3s;  
}
```

Il est à noter que pour des raisons **d'optimisation de la synchronisation des effets de transition**, les développeurs Mozilla **recommandent** d'utiliser la **syntaxe raccourcie** ou abrégée.

NOTONS aussi que lorsqu'on utilise la notation raccourcie, la **première valeur de temps** correspond toujours à la **durée**.

## Les propriétés animables et optimisation

De nombreuses propriétés CSS sont « animables ». Citons par exemple, les propriétés suivantes :

- background-color, background-position, border, bottom, color, font-size, font-weight, height, left, letter-spacing, line-height, margin-bottom, margin-left, margin-right, margin-top, max-height, max-width, min-height, min-width, opacity, outline-color, outline-width, padding-bottom, padding-left, padding-right, padding-top, right, text-indent, text-shadow, top, vertical-align, visibility, width, word-spacing, z-index.

Précisons toutefois que pour des questions d'**optimisation**, il est **fortement recommandé** de ne **PAS animer** des propriétés qui affectent la **disposition** des éléments (*layout*) de la page Web, comme par exemple, les propriétés *height*, *width*, *top*, *left*, *margin*, etc.

En effet, **certaines propriétés CSS sont beaucoup plus chères à animer que d'autres**. Par exemple, lorsque la hauteur d'un élément augmente ou diminue cela provoque une réaction en chaîne; tous ses frères et sœurs devront également monter ou descendre, pour remplir l'espace! Tous ces déplacements exigent beaucoup de calculs de la part du navigateur, ce qui risque de produire une animation saccadée.

**Les deux propriétés les moins coûteuses à animer sont : opacity et transform (fonctions: translate, scale et rotate).**

## Quelques références utiles sur les transitions CSS

### Site officiel du W3C:

- Spécifications officielles du W3C. (Octobre 2018). CSS Transitions. Repéré à <https://www.w3.org/TR/css-transitions-1/>

### Bonne référence en français

- Mozilla Corporation. (Octobre 2022). Utiliser les transitions CSS | MDN. Repéré à : [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_Transitions/Utiliser\\_transitions\\_CSS](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Transitions/Utiliser_transitions_CSS)

### Bonnes références en anglais

- w3schools. (s.-d.). CSS Transitions. Repéré à http: [https://www.w3schools.com/css/css3\\_transitions.asp](https://www.w3schools.com/css/css3_transitions.asp)
- Comeau, Josh (2020). An interactive Guide to CSS Transitions. Repéré à : <https://www.joshwcomeau.com/animation/css-transitions/>

### Propriétés CSS animables

- w3schools. (s.-d.). CSS Animatable. Repéré à : [http://www.w3schools.com/cssref/css\\_animatable.asp](http://www.w3schools.com/cssref/css_animatable.asp)
- Mozilla Corporation. (Septembre 2022). CSS animated properties - CSS | MDN. Repéré à : [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_animated\\_properties](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties)

## Cibler et modifier un élément HTML avec JavaScript

### Accéder aux éléments HTML d'une page Web

JavaScript dispose de plusieurs fonctions nous permettant de récupérer par programmation un élément HTML d'une page Web. À cet effet, un objet JavaScript appelé **document** qui référence simplement le document HTML et, plus précisément la balise <html>. La fonction de l'objet document vue et utilisée dans le cours 1J1 Animation et interactivité en jeu est **document.querySelector()**.

#### document.querySelector()

La méthode `querySelector()` permet d'accéder à un élément HTML à partir d'un sélecteur CSS. Cette fonction renvoie le **premier** élément trouvé correspondant au sélecteur CSS indiqué. La syntaxe est la suivante:

**document.querySelector(selecteurCSS);**

où **selecteurCSS** représente un **sélecteur CSS valide** passé sous forme de **chaîne de caractères**. Il peut s'agir du nom d'une balise HTML (p, div, img, etc.), d'un id ou d'un autre élément **OBLIGATOIREMENT** présent dans le <body> de la page HTML.

La référence au sélecteur est mémorisée dans une variable et on peut ensuite modifier les propriétés du sélecteur avec JavaScript. Dans l'exemple qui suit, on récupère la balise <img> et on lui attribue une image source par programmation (propriété `src`):

```
<img src = " " >
<script>
  //Récupérer la balise <img>
  let baliseImg = document.querySelector("img");
  //Changer l'image source de cette balise
  baliseImg.src = "monImage.png"; //l'image doit être présente dans le dossier
</script>
```

### Modifier un style CSS d'un élément HTML

La majorité des **styles CSS** applicables à un élément HTML (marges, contours, arrière-plan, bordures, etc.) sont modifiables avec JavaScript en ciblant la propriété générale **style** de la balise, puis en indiquant la propriété de styles CSS à modifier. Pour changer le style d'un élément HTML, on utilise cette syntaxe:

**elementHTML.style.property = "valeur";**

- **elementHTML**, est l'élément HTML récupéré avec JavaScript, en appelant notamment **document.querySelector()**;
- **style**, fait référence à **tous** les styles CSS **attribuables** à l'élément HTML ciblé;
- **property**, est le nom d'une propriété CSS. Attention, en JavaScript, tous les **noms** des propriétés CSS sont écrites **sans trait-d'union** et respectent la norme *camelCase*. Ainsi `background-color`, s'écrit comme suit: `backgroundColor`;
- **"valeur"**, est une valeur CSS valide pour la propriété CSS spécifiée, à l'exception que **toutes les valeurs doivent être écrites sous forme de chaîne** (entre guillemets).

Par exemple, le code suivant affecte par programmation une image d'arrière-plan à la balise <canvas>:

```
<script>
  //Récupérer la balise <canvas>
  let leCanvas = document.querySelector("canvas");
  //Changer l'image source de cette balise
  leCanvas.style.backgroundImage = "monImage.png";
</script>
```

## Rappel au sujet de la gestion des événements

Les **événements** sont des **actions** qui se produisent et **auxquelles on va pouvoir répondre** en exécutant un code. Les événements et leur prise en charge sont l'un des mécanismes principaux de JavaScript qui vont nous permettre d'ajouter de l'interactivité à nos pages Web. Par exemple, on va pouvoir afficher ou cacher du texte suite à un clic d'un utilisateur sur un élément, changer la taille d'un texte lors du passage de la souris d'un utilisateur sur un élément, etc.

En JavaScript, un **événement** est une **action** qui se produit et qui possède **deux** caractéristiques essentielles :

- C'est une action qu'on peut « **écouter** », c'est-à-dire une action qu'on peut détecter car le navigateur va nous informer qu'elle se produit;
- C'est une action à laquelle on peut « **répondre** », c'est-à-dire qu'on va pouvoir attacher un code à cette action qui va s'exécuter dès qu'elle va se produire.

Par exemple, on va pouvoir détecter le clic d'un utilisateur sur l'élément HTML <canvas> et afficher une boîte de dialogue ou une image suite à ce clic.

### Utiliser la méthode addEventListener()

**Pour écouter et répondre à un événement, nous allons définir ce qu'on appelle des gestionnaires d'événement.** En JavaScript, les gestionnaires d'événement sont implémentés avec la méthode `addEventListener()`, dont la syntaxe est la suivante:

`elementHTML.addEventListener(eventement, fonctionAexecuter)`

Cette fonction prend deux principaux paramètres :

- Le **nom de l'événement** qu'on souhaite prendre en charge. Celui-ci est passé sous forme de **chaînes** (String);
- La **fonction à exécuter** en cas de déclenchement de cet événement.

Dans le code qui suit, on affiche une boîte de dialogue si l'utilisateur clique sur l'élément HTML <canvas>:

```
<script>
  //Gestionnaire d'événement
  leCanvas.addEventListener("click", afficherAlerte);
  //Fonction permettant d'afficher une boîte de dialogue
  function afficherAlerte(){
    alert("On a cliqué sur la balise <canvas>");
  }
</script>
```



```
}  
</script>
```

Le résultat est le suivant:

