Les effets avancés en CSS

Table des matières

I. Contexte

Durée : 1 h 15

Environnement de travail : Visual Studio Code 1.42

Pré-requis : Bases de HTML, bases de CSS, notion de boîtes.

Pour élaborer des styles plus complexes et plus modernes pour les projets de sites web, il est nécessaire de se munir de nouveaux outils. C'est dans cette optique que nous allons voir une succession de propriétés, qui ont pour la plupart été ajoutées avec CSS3. Même si ces propriétés sont assez récentes, elles sont généralement compatibles avec les navigateurs modernes. En cas de doute, il convient de se rendre sur le site https://developer.mozilla.org/fr/Mozilla Developer Network (MDN)Mozilla Developer Network (MDN) ou https://caniuse.com/Can I useCanIUse pour vérifier la compatibilité de ces propriétés avec les navigateurs.

II. Styliser ses pages web

Objectifs

- Découvrir une série de propriétés pour mettre en valeur son contenu HTML
- Donner à son site web un aspect plus moderne et professionnel
- Savoir où placer le curseur des effets pour un visuel réussi

Plus de couleurs

Nous avons vu jusqu'ici deux manières d'exprimer une couleur en CSS : soit par une couleur nommée, par exemple "*black*", "*red*", ou "*lightpink*" (vous pourrez trouver la liste sur la page w3 School[1]) ; soit par la représentation de la couleur sous la forme d'un code hexadécimal.

```
/* Les quatre exemples ci-dessous sont toutes des écritures de la couleur noire */
color: black;
color: #000000;
color: rgb(0, 0, 0);
color: hsl(0, 0%, 0%);

/* Les quatre exemples ci-dessous sont toutes des écritures de la couleur blanche */
color: white;
color: #ffffff;
color: rgb(255, 255, 255);
color: hsl(0, 0%, 100%);
```

Une fonctionnalité très intéressante, apportée par les deux dernières notations, est la possibilité de créer une couleur semi-transparente. Pour cela, c'est très simple : il suffit d'ajouter un quatrième paramètre exprimé sous la forme d'un nombre entre 0 (transparent) et 1 (opaque).

Les notations deviennent alors RGBA et HSLA (le **a** signifie "alpha").

```
color: rgb(0, 0, 0); /* Noir opaque */
color: rgba(0, 0, 0, 0.8); /* Noir avec 80% d'opacité */

color: hsl(0, 0%, 0%); /* Noir opaque */
color: hsla(0, 0%, 0%, 0.2); /* Noir avec 20% d'opacité */
```

---

1 W3 School [en]

Remarque

Vous pouvez utiliser ces différentes notations de manière indifférente, chacune permettant au final de pouvoir écrire les mêmes couleurs.

box-shadow

La propriété `box-shadow` permet d'ajouter une ombre à une boîte comme une * *. Cela permet de donner un effet de profondeur à la page web en faisant ressortir certains éléments.

Afin de comprendre comment s'utilise la propriété, nous pouvons nous rendre sur la documentation MDN[1].

Comme souvent en CSS, `box-shadow` prend un nombre de paramètres qui peut varier. Dans son cas d'utilisation le plus classique, nous devons renseigner quatre paramètres :

```
/* décalage sur l'axe "x", décalage sur l'axe "y", "rayon" de flou, couleur */
box-shadow: 2px 4px 10px hsla(0, 0%, 0%, 0.8);
```

Revenons ensemble sur ces paramètres :

- **Décalage sur l'axe "x"**

  Une valeur positive, comme ici "2px", signifie que l'ombre sera décalée vers la droite par rapport à la boîte sur laquelle on applique l'effet.

  On pourrait très bien indiquer une valeur négative, par exemple "-2px", pour faire l'effet inverse et décaler l'ombre vers la gauche de la boîte.

- **Décalage de l'axe "y"**

  Même chose, une valeur positive va décaler l'ombre vers le bas, une valeur négative, vers le haut.

- **"Rayon" de flou**

  Représente l'intensité de l'ombre. Il faut comprendre qu'une ombre n'est rien de plus qu'un dégradé de la couleur choisie vers le transparent. Plus la valeur est élevée, plus l'ombre sera diffuse : le dégradé sera plus doux, mais prendra plus d'espace pour revenir à la couleur transparente. Attention, `box-shadow` peut dans certains cas déborder sur d'autres boîtes.

- **Couleur**

  Il est possible d'utiliser tous les modes de couleur supportés par CSS3 : couleur nommée, code hexadécimal, RGB, HSL.

  Il est également possible de donner une couleur semi-transparente, comme nous l'avons vu précédemment. Cet effet devrait être utilisé avec retenue : il est plus réussi quand l'ombre n'est pas trop prononcée. L'utilisation de gris plus clairs ou de noir avec de la transparence permet généralement d'atténuer l'effet de l'ombre.

1 Documentation MDN de box-shadow

Remarque

Il est possible de mettre plusieurs ombres sur un élément. Pour cela, il convient d'écrire la propriété `box-shadow` et de séparer chaque ensemble de paramètres par une virgule.

```
box-shadow: 2px 4px 10px hsla(0, 0%, 0%, 0.8), 0 0 5px red, -4px 0 8px #0984e3;
```

Méthode

Il peut être fastidieux de tester de nombreuses combinaisons de valeurs pour obtenir une ombre satisfaisante visuellement, c'est pourquoi il existe plusieurs générateurs de **box-shadow** trouvables en recherchant les termes *box-shadow generator* dans un moteur de recherche. Ces outils fonctionnent généralement avec plusieurs curseurs pour chaque valeur du box-shadow et offrent un aperçu en temps réel.

text-shadow

Nous venons de voir comment ajouter une ombre sur une boîte : il est possible de faire la même chose directement sur un texte, à l'aide de la propriété `text-shadow`.

Cette propriété s'utilise de la même manière que `box-shadow`, il y a quatre paramètres à renseigner : le décalage sur l'axe "x", le décalage sur l'axe "y", le "rayon" de flou et la couleur.

Tout comme `box-shadow`, il est possible de donner plusieurs ombres à appliquer sur le texte. À noter que, comme pour `box-shadow`, l'ordre selon lequel vous préciserez vos ombres aura une importance : votre navigateur les affichera d'avant en arrière, c'est-à-dire que la première ombre de la liste sera affichée "au-dessus", et les suivantes en dessous.

```
/* décalage sur l'axe "x", décalage sur l'axe "y", "rayon" de flou, couleur */
text-shadow: 2px 4px 10px hsla(0, 0%, 0%, 0.8);
```

border-radius

Tous les exemples utilisés jusqu'ici ont mis en évidence qu'en HTML CSS, tous les éléments sont rectangles. Il est cependant possible de créer des éléments arrondis avec l'aide de la propriété `border-radius`.

Pour cela, il suffit d'ajouter la propriété à une boîte, une * * par exemple :

```
.maBoite {
  background-color: blue;
  border-radius: 10px;
}
```

Complément

La propriété `border-radius` s'applique également sur les images. Sur des sites de réseaux sociaux comme Facebook ou Twitter, les photos de profil sont généralement rondes. Afin d'obtenir un résultat similaire, il suffit de spécifier un `border-radius` supérieur à la moitié de la taille du côté de l'image.

Exemple : Pour une image de 300 px par 300 px, il faudra préciser un border-radius de 150 px pour qu'elle soit ronde.

```
<img class="bear" src="https://placebear.com/300/300" alt="bear">
```

```
.bear {
  border-radius: 150px;
}
```

Il est également possible de créer un `border-radius` avec une valeur en pourcentage. Pour arrondir complètement une image, il faudra alors que le `border-radius` soit de **50 %**.

Attention

Pour obtenir un cercle, il est nécessaire que l'image soit carrée ; dans le cas d'un rectangle, la résultante sera une forme ovale.

`Border-radius` est également une propriété qui prend un nombre de paramètres variables. Sur l'exemple précédent, nous avons saisi une seule valeur, ce qui indique au navigateur que les quatre coins doivent être arrondis de la même manière. Cependant, il est possible de spécifier quatre valeurs distinctes pour donner aux quatre coins un arrondi plus ou moins important. L'ordre d'écriture sera alors : **haut, droite, bas et gauche**.

```
#example1 {
border-radius: 1em / 5em;
/* est équivalent à */
border-top-left-radius:     1em 5em;
border-top-right-radius: 1em 5em;
border-bottom-right-radius: 1em 5em;
border-bottom-left-radius:  1em 5em;
}

#example2 {
border-radius: 4px 3px 6px / 2px 4px;
/* est équivalent à : */
border-top-left-radius:     4px 2px;
border-top-right-radius:    3px 4px;
border-bottom-right-radius: 6px 2px;
border-bottom-left-radius:  3px 4px;
}

```

Syntaxe **background**

La propriété CSS background est une propriété raccourcie (propriétés qui permettent de définir la valeur de plusieurs propriétés via une seule déclaration), qui permet de définir les différentes valeurs des propriétés liées à la gestion des arrière-plans d'un élément (couleur, image, origine, taille, répétition, etc.).

Arrière-plans

Nous avons vu l'utilisation de la balise `<img>` pour afficher une image dans la page, mais CSS propose lui aussi une technique pour afficher une image.

Avec l'aide de la propriété `background-image`, nous pouvons cibler une boîte et lui appliquer une ou plusieurs images de fond, et paramétrer la manière dont ces images vont se comporter.

Le cas d'utilisation le plus simple est le suivant :

```
.maBoite {
  background-image: url('lien/vers/mon/image');
}
```

Davantage de possibilités s'offrent à vous. Le lien vers votre image peut être un lien vers une image locale, qui serait placée dans le répertoire de votre site web : le lien est alors un lien relatif qui permet d'aller jusqu'à l'image depuis l'emplacement du fichier où on écrit la propriété CSS. Le lien peut être une URL absolue pour une image stockée sur le Web.

```
<div class="with-background"></div>
```

```
.with-background {
  background-image: url('https://placebear.com/400/250');
}
```

Voici un exemple[1] de `background-image` avec une image hébergée sur le Web.

Attention

Si vous utilisez une image qui provient du Web et dont vous ne maîtrisez pas l'hébergement, il se pourrait que cette image soit un jour supprimée ou déplacée par son auteur : auquel cas, elle ne s'affichera plus sur votre site !

Il est également possible de superposer plusieurs images de fond. Dans ce cas là, il suffit d'une seule propriété `background-image` et de mettre plusieurs **url()** séparées par des virgules.

Les images sont alors affichées dans l'ordre : la première sera "au-dessus", les suivantes seront en dessous.

Comme nous l'avons dit, plusieurs autres propriétés CSS nous permettent de contrôler la manière dont sera affichée l'image de fond.

Syntaxe **background-repeat**

Par défaut, si vous essayez de couvrir une boîte avec une image trop petite, l'image va automatiquement se répéter comme le motif d'un papier peint pour couvrir toute la zone disponible.

Vous pouvez agir sur ce mécanisme avec les valeurs :

- *repeat-x* : répète l'image seulement sur l'axe X
- *repeat-y* : répète l'image seulement sur l'axe Y

1 exemple Repl.it background-image

- *no-repeat* : ne répète pas l'image

Il existe d'autres valeurs que vous pouvez consulter et expérimenter sur la documentation[1].

Syntaxe **background-position**

Cette propriété permet de gérer la position de l'image de fond par rapport à la boîte ciblée. Il y a un certain nombre de combinaisons de paramètres acceptées par cette propriété et nous ne les couvrirons pas toutes ici. Vous pouvez retrouver la liste sur la documentation[2].

L'utilisation la plus simple consiste à indiquer : *top*, *left*, *bottom*, *right, center* pour aligner l'image, respectivement en haut, à gauche, en bas, à droite ou au centre de la boîte. Il est également possible de donner deux valeurs en pourcentage, qui sont respectivement le décalage depuis la gauche et le décalage depuis le haut.

```
background-position: center; // centre l'image de fond dans la boîte.
background-position: 25% 75%; // place l'image à 25% de la gauche et 75% du haut de la boîte.
background-position: bottom: 10% right: 30%; // place l'image à 10% du bas et 30% de la droite de la boîte.
```

Syntaxe **background-size**

Cette propriété configure la taille de l'image de fond par rapport à la boîte. On peut également lui donner des valeurs qui s'occuperont de calculer automatiquement la taille par rapport à la configuration souhaitée.

La valeur *contain* va redimensionner l'image de sorte à ce qu'elle soit entièrement affichée dans la boîte. Le ratio largeur/hauteur sera respecté.

cover, de son côté, va redimensionner l'image de sorte que toute la boîte soit recouverte, même si une partie de l'image doit être masquée ; le ratio largeur/hauteur sera également respecté.

Il est également possible de paramétrer manuellement la taille de l'image, en donnant une valeur en pourcentage. L'image sera redimensionnée en conservant son ratio. Finalement, il est possible de donner deux valeurs : largeur et hauteur.

```
background-size: contain;
background-size: cover;
background-size: 50%;
background-size: 200px 150px;
```

A. Dégradés

Nous avons pu voir que la propriété `background` pouvait servir à afficher une couleur de fond simple, une ou plusieurs image(s) à l'aide de `background-image`, nous allons maintenant voir comment appliquer un dégradé de couleurs.

Il existe plusieurs types de dégradés que nous allons pouvoir utiliser.

1 Documentation MDN sur background-repeat
2 Documentation MDN sur background-position

Syntaxe **linear-gradient**

C'est le plus simple à comprendre ! Il prend un nombre variable de paramètres, mais, dans sa forme la plus simple, il prend deux couleurs et remplit de haut en bas :

```
background: linear-gradient(blue, red); // Affiche un dégradé de bleu vers rouge partant du haut de la boîte jusqu'en bas
```

Nous pouvons rajouter autant de couleurs que l'on souhaite : celles-ci seront ajoutées au dégradé et partageront l'espace disponible.

Il est possible d'ajouter un paramètre en première position pour déterminer l'orientation du dégradé. Ce paramètre s'écrit avec une unité un peu particulière, "*deg*", et représente un angle.

```
background: linear-gradient(45deg, blue, red) // Affiche un dégradé de bleu vers rouge partant de gauche jusqu'à droite
```

Syntaxe **radial-gradient**

Il fonctionne de manière similaire à `linear-gradient`, à la différence que ce dégradé est circulaire.

De la même manière qu'avec un dégradé linéaire, il est possible d'indiquer autant de couleurs que l'on souhaite : le premier paramètre définit l'origine du dégradé, par défaut un cercle au centre de l'écran.

Je vous invite à vous rendre sur la documentation pour obtenir plus d'informations sur les paramètres disponibles !

Complément

Il est important de noter qu'il est tout à fait possible de superposer plusieurs *backgrounds*, comme on l'a vu jusqu'ici avec plusieurs propriétés. C'est toujours le premier élément qui sera affiché au-dessus.

Un ours multicolore[1] ! Amusez-vous à tester diverses combinaisons de `backgrounds`, c'est un outil assez simple à prendre en main, mais pour autant très puissant !

Rappel

Vous en savez maintenant un peu plus sur ces propriétés CSS qui vous permettent de créer des effets visuels avancés sur votre HTML. Il existe plus de propriétés que ce que nous avons vu, et je vous invite à consulter la documentation[2] pour en savoir plus.

Attention

Il faut garder à l'esprit que toutes ces propriétés sont pour la plupart calculées par la carte graphique de l'ordinateur de l'utilisateur.

Une configuration modeste ou ancienne (éléments matériels "d'entrée de gamme", par exemple) aurait du mal à faire tourner un site web trop chargé en styles.

Le temps de chargement d'un site web est crucial dans l'expérience utilisateur, et il est bon de considérer ces aspects lors de la réalisation de son site web.

1 exemple Repl.it background-multiple
2 https://developer.mozilla.org/fr/docs/Web/CSS/Reference

Complément

Comme nous avons pu déjà le dire, ces éléments visuels permettent de créer des styles modernes et complexes, mais leur utilisation doit rester maîtrisée.

En effet, trop d'effets de styles ou des styles trop prononcés pourraient impacter négativement la lisibilité de votre page.

Une bonne chose à faire en cas de doute sur la manière de styliser une page web est de chercher l'inspiration auprès d'autres designers.

Le site Dribbble est un bon exemple de référence pour les web designers en manque d'inspiration.

III. Exercice : Appliquez la notion

Question 1

Écrivez le code qui permet de mettre une ombre à une * * dont la classe serait `.boxWithShadow`.

- L'ombre doit être décalée de 4 px sur la gauche de `.boxWithShadow`
- L'ombre doit être décalée de 6 px en bas de `.boxWithShadow`
- L'ombre doit s'étendre sur un rayon de 20 px
- L'ombre doit être de couleur noire à 80 % d'opacité exprimée en HSL

Question 2

Écrivez le code qui permet de mettre des coins arrondis et une image de fond à une * * dont la classe serait `.roundedBox`.

- Pas besoin d'une vraie image, vous pouvez mettre le chemin vers une image factice
- L'arrondi des coins doit suivre la consigne suivante :
    - Coin haut-gauche 8 px
    - Coin haut-droite 20 px
    - Coin bas-droite 4 px
    - Coin bas-gauche 6 px

Question 3

Écrivez le code qui permet de mettre un fond en dégradé à une * * dont la classe serait `.gradientBox`.

Le dégradé doit avoir un angle de 25 degrés et aller du jaune au vert.

Question 4

Écrivez le code qui permet de rendre une image de profil (avatar). Vous pouvez utiliser border-radius.

IV. Les transitions CSS

Objectifs

- Comprendre la différence entre « transition » et « animation »
- Mettre en place du feedback pour l'utilisateur
- Paramétrer plus finement son animation

Mise en situation

Le Web nous offre la possibilité au-delà de styliser les éléments de créer du mouvement sur la page. Le mouvement va automatiquement attirer l'œil des utilisateurs de votre site web.

Ce qui peut être une bonne chose pour attirer l'attention sur un élément en particulier, c'est faire comprendre à l'utilisateur qu'une action est disponible sur cet élément.

Il permet d'obtenir un contenu plus interactif, plus engageant et plus fun. Voyons tout de suite comment introduire du mouvement sur vos pages web !

Méthode **Les transitions**

La notion de transition en CSS désigne le changement d'une propriété CSS de manière fluide. Par exemple, faire changer une * * avec un fond rouge vers un fond bleu.

Pour réaliser une transition, il faut commencer par écrire la modification que l'on souhaite voir se réaliser.

Dans l'exemple, commençons par écrire le code permettant de faire changer la couleur du background sur la pseudo-classe `:hover` de notre élément.

Nous devrions alors voir un changement abrupt de la couleur lorsqu'on passe le curseur de la souris sur la * *.

Nous pouvons alors ajouter la propriété CSS `transition` pour ajouter un effet de fluidité entre deux états.

La propriété transition s'utilise avec un nombre variable de paramètres, mais on l'utilise généralement de la manière suivante :

```
transition: background 0.4s linear 0.5s;
```

Le premier paramètre, `property`, est le nom de la propriété que l'on souhaite voir bouger. Ici, il s'agit de *background* pour modifier la couleur de fond de rouge à bleu.

Ensuite vient le paramètre `duration`, qui accepte une durée exprimée en secondes. Ici, la durée de transition est de 0.4 s.

Puis vient `timing-fonction` (« modèle d'interpolation » en français), qui va définir le comportement de notre transition dans le temps.

Cette propriété peut avoir les valeurs suivantes :

- ease - spécifie un effet de transition avec un démarrage lent, puis rapide, puis une fin lente (c'est la valeur par défaut)
- linear - spécifie un effet de transition avec la même vitesse du début à la fin
- ease-in - spécifie un effet de transition avec un démarrage lent
- ease-out - spécifie un effet de transition avec une fin lente
- ease-in-out - spécifie un effet de transition avec un début et une fin lents
- cubic-bezier(n,n,n,n) - vous permet de définir vos propres valeurs dans une fonction cubique-bézier

```
div { transition-timing-function: linear;
 transition-timing-function: ease;
 transition-timing-function: ease-in;
 transition-timing-function: ease-out;
 transition-timing-function: ease-in-out;
}
```

Enfin, on trouve le `delay`, c'est-à-dire le délai "après" lequel la transition doit se déclencher quand l'utilisateur effectue l'action ciblée (ici, la pseudo-class `:hover`).

```css
div {
  transition-property: width;
  transition-duration: 2s;
  transition-timing-function: linear;
  transition-delay: 1s;
}
```

Exemple

Voici un exemple[1] de la transition du fond de rouge vers bleu au passage du curseur de la souris.

La propriété transition

La seconde composante de la transition est la propriété transition en elle-même, qui va définir la durée, le délai et la timing-function que l'on souhaite appliquer à notre transition.

```css
animation: 3s ease 1s infinite alternate changeBackground;
```

Prenons les paramètres dans l'ordre :

`duration` : durée de l'animation exprimée en secondes. Il est possible d'écrire un nombre décimal pour une animation plus rapide qu'une seconde, par exemple **0.5 s** ou de l'exprimer en millisecondes (**500 ms**),

`timing-function` : "modèle d'interpolation" (pas de panique, on verra ça dans une future partie !),

`delay` : délai avant déclenchement,

`iteration-count` : nombre de fois que l'on souhaite voir l'animation après son premier déclenchement (ici infinite, pour la jouer continuellement),

`direction` : sens de l'animation qui sera jouée. Les valeurs possibles sont :

- `running` pour aller dans le sens décrit dans la fonction `@keyframes`
- `reverse` pour jouer l'animation dans le sens inverse, c'est-à-dire commencer par la keyframe **100%** et terminer à la keyframe **0%**
- `alternate` pour jouer l'animation en alternant running et reverse (ce mode sera visible si vous jouez l'animation plus d'une seule fois dans le paramètre `iteration-count`)

`name` : nom de l'animation qui doit correspondre au nom de la fonction `@keyframes` que vous avez créée.

Attention

Attention, toutes les propriétés ne sont pas compatibles avec les transitions. Voici une liste des propriétés qui sont compatibles[2].

Complément

Nous avons vu dans l'exemple précédent que l'application de la transition se faisait avec la pseudo-classe `:hover`, mais toutes les pseudo-classes peuvent déclencher une transition.

De plus, ajouter ou enlever une classe CSS avec JavaScript peut également déclencher une transition sur ces classes.

1 Exemple Repl.it transition css
2 Liste des propriétés disponibles pour animation et transition

V. Exercice : Appliquez la notion

Question

Écrivez le code qui permet de faire une transition pour la * * avec la classe `.boxWithTransition`.

- `.boxWithTransition` doit faire par défaut 200 px de hauteur et 400 px de largeur
- `.boxWithTransition` doit avoir une couleur de fond grise par défaut
- Au passage de la souris, une transition doit se déclencher pour que `.boxWithTransition` change sa largeur à 700 px et sa couleur de fond à bleu
- La transition doit s'effectuer en 0.8 secondes
- La transition doit utiliser la `timing-function` `ease`

Indice :

Vous pouvez lancer la transition sur la propriété `all` pour faire la transition sur plusieurs propriétés d'un coup.

VI. Les animations CSS

Objectifs

- Comprendre la différence entre « transition » et « animation »
- Mettre en place du feedback pour l'utilisateur
- Paramétrer plus finement son animation

Mise en situation

Les animations permettent de créer un mouvement qui n'est pas forcément déclenché par une pseudo-classe ni basé spécifiquement sur la modification d'une propriété.

Méthode **Créer une animation**

La création d'une animation se fait généralement par l'intermédiaire d'une fonction `@keyframes` qui décrit tout le mouvement devant être effectué, et d'une propriété `animation` qui paramètre la durée et la vitesse.

Nous allons détailler ces deux composantes.

La fonction @keyframes

La fonction `@keyframes` va définir toutes les modifications que va subir notre élément cible.

Attention

À noter que, comme pour les transitions, la liste des propriétés compatibles est limitée et disponible sur ce lien[1].

Dans sa forme minimale, elle s'utilise de cette manière :

```
@keyframes changeColor {
  from { background: red; }
  to   { background: blue; }
}
```

Les fonctions `@keyframes` ne se placent pas à l'intérieur d'un sélecteur CSS, mais bien au même niveau que les sélecteurs. Elles sont généralement regroupées en haut ou en bas du fichier, voire dans un fichier séparé.

1 Documentation MDN liste des propriétés disponibles pour animation et transition

changeColor est le nom que l'on donne à notre animation : c'est arbitraire et vous pouvez donner le nom que vous souhaitez.

Nous voyons ensuite deux fonctions, `from` et `to`, qui exposent les états de départ et d'arrivée souhaités pour notre élément cible.

Ici, on peut voir que l'état souhaité au départ pour notre élément est un fond rouge, et nous souhaitons arriver à un fond bleu.

Conseil

À la place de `from` et `to`, nous aurions pu utiliser des pourcentages pour exprimer :

- 0%, soit le début de l'animation
- 100%, soit la fin de l'animation

De cette manière, nous pouvons ajouter des états intermédiaires avec les valeurs en pourcentages :

```
@keyframes changeColor {
  0% { background: red; }
  40% { background: yellow; }
  90% { background: green; }
  100% { background: blue; }
}
```

Les deux seules valeurs obligatoires sont 0% (ou `from`) et 100% (ou `to`). Vous pouvez ensuite ajouter autant d'étapes intermédiaires que vous le souhaitez.

La propriété animation

La seconde composante de l'animation est la propriété `animation` en elle-même, qui va définir la durée, le délai et la `timing-function` que l'on souhaite appliquer à notre animation.

```
animation: 3s ease 1s infinite alternate changeBackground;
```

Prenons les paramètres dans l'ordre :

- `duration` : durée de l'animation exprimée en secondes. Il est possible d'écrire un nombre décimal pour une animation plus rapide qu'une seconde, par exemple **0.5 s** ou de l'exprimer en millisecondes (**500 ms**),
- `timing-function` : "modèle d'interpolation" (pas de panique, on verra ça dans une future partie !),
- `delay` : délai avant déclenchement,
- `iteration-count` : nombre de fois que l'on souhaite voir l'animation après son premier déclenchement (ici `infinite`, pour la jouer continuellement),
- `direction` : sens de l'animation qui sera jouée.

  Les valeurs possibles sont :

  - `normal` : l'animation est jouée dans le sens normal à chaque cycle. C'est le réglage par défaut.
  - `running` pour aller dans le sens décrit dans la fonction `@keyframes`
  - `reverse` pour jouer l'animation dans le sens inverse, c'est-à-dire commencer par la keyframe **100%** et terminer à la keyframe **0%**
  - `alternate` pour jouer l'animation en alternant `running` et `reverse` (ce mode sera visible si vous jouez l'animation plus d'une seule fois dans le paramètre `iteration-count`).

  `name` : nom de l'animation qui doit correspondre au nom de la fonction `@keyframes` que vous avez créée.

Remarque

À noter que vous pouvez donner plusieurs instructions dans la fonction `@keyframes` et qu'il est tout à fait possible de déplacer, tourner et changer la couleur de fond de notre élément cible en simultané.

Voici un exemple[1] qui montre une application avec plusieurs éléments que nous venons de voir.

Un paramètre `timing-function` est présent dans les transitions et les animations. Ce paramètre permet de définir comment va se dérouler l'animation.

Par défaut, le paramètre vaut toujours "*ease*", ce qui signifie que la vitesse de la transition augmente au milieu de celle-ci puis ralentit à la fin.

Il existe cependant d'autres timing-functions que vous pouvez utiliser :

- `ease-in` permet d'avoir une vitesse lente au début, mais qui augmente exponentiellement
- `ease-out` permet d'avoir une vitesse rapide au début, mais qui décroît exponentiellement
- `ease` est un mélange avec un pic de vitesse en milieu de parcours
- `ease-in-out` est le mélange de `ease-in` et `ease-out` : son utilisation est déconseillée pour les animations de moins d'une seconde (l'œil humain n'a simplement pas le temps de voir toutes les variations de vitesse dans ce cas)

Exemple

Voici un exemple[2] qui vous permettra de vous rendre compte de la différence entre les timing-functions pour une animation très simple de déplacement d'un carré !

Complément

Vous pouvez également configurer vous-même vos préférences en indiquant comme `timing-function` "*cubic-bezier*".

On entre alors sur le terrain des mathématiques, avec une fonction qui accepte 4 paramètres (*x1, y1, x2, y2*) pour paramétrer très finement votre animation.

Il existe plusieurs outils pour créer une courbe de Bézier de manière visuelle sans directement entrer des nombres. https://cubic-bezier.com/[3] en est un exemple.

1 Exemple Repl.it animation css
2 Exemple Repl.it timing-functions
3 Outil création courbes de bezier

La propriété transform

L'attribut transform applique une transformation 2D ou 3D à un élément. Cette propriété permet de faire pivoter, tourner, déplacer, d'appliquer des homothéties sur des éléments.

Exemple :

```
transform : rotate(20deg);
transform : translate(20px, 10px);
transform : scale(1, 2);
transform : skew(45deg, 62deg);
transform : translate(100px) rotate(20deg) skew(45deg, 62deg) scale(1, 2);
```

La propriété de Transformation peut prendre pour les transformations 2D ou 3D une/des valeurs suivantes :

none: aucune transformation, valeur par défaut.

- matrix() : permet d'appliquer une matrice de transformation à un élément.
- translate() : déplacer horizontalement et verticalement un élément.
- translateX() : déplacer horizontalement un élément.
- translateY() : déplacer verticalement un élément.
- translateZ() : déplacer un élément en profondeur.
- Translate3D() : déplacer sur les trois axes de perspective un élément.
- Scale() : redimensionner horizontalement et verticalement un élément.
- ScaleX() : redimensionner horizontalement un élément.
- ScaleY() : redimensionner verticalement un élément.
- ScaleZ() : redimensionner un élément sur sa profondeur.
- Scale3D() : redimensionner un élément sur les trois axes de perspective.
- rotate() : permet de faire une rotation à un élément.
- rotateX() : faire une rotation à un élément sur l'axe de perspective X.
- rotateY() : faire une rotation à un élément sur l'axe de perspective Y.
- rotate 3D() : faire une rotation à un élément sur les trois axes de perspective.
- skew() : incliner horizontalement et verticalement un élément.
- skewX() : incliner horizontalement un élément.
- skewY() : incliner verticalement un élément.
- perspective() : définit une perspective view pour un élément transformé en 3D.
- inherit : Hérite la propriété de son élément parent.
- initial : remet la propriété à sa valeur par défaut.

Attention

Seuls certains éléments peuvent être convertis. Il n'est pas possible de convertir des éléments dont la mise en page est gérée par une zone en ligne, une colonne de tableau ou un groupe de colonnes de tableau non remplacé.

En résumé

Les transitions et les animations CSS sont similaires à bien des égards, mais elles diffèrent par la complexité des transitions, la façon dont le code CSS interagit avec JavaScript, le fonctionnement des boucles et la méthode de déclenchement de la lecture de l'animation. Les transitions CSS sont généralement les meilleures pour les mouvements simples un à un, tandis que les animations CSS conviennent aux séries de mouvements plus complexes.

VII. Exercice : Appliquez la notion

Question

Écrivez le code qui permet d'animer une * * qui aurait une classe `.boxWithAnimation`.

- `.boxWithAnimation` doit commencer en ayant un fond noir (exprimé en RGB), faire 200 px de large par 200 px de haut, et être complètement ronde.
- `.boxWithAnimation` doit se déplacer de 400 px vers le bas durant l'animation.
- `.boxWithAnimation` doit avoir un fond jaune à mi-parcours de son animation et un fond rouge à la fin de son animation.
- L'animation doit prendre 0.8 s, utiliser la timing-function `ease-in`, démarrer après 0.5 s de délai, se jouer indéfiniment en alternant (c'est-à-dire aller une fois vers le bas, puis revenir à sa position et couleur initiale).

Indice :

Vous pouvez utiliser *transform : translateY(400px)* pour réaliser le déplacement vers le bas.

VIII. Auto-évaluation

A. Exercice final

Exercice 1

Exercice

Parmi les propositions suivantes, lesquelles sont équivalentes à une couleur noire ?

- ☐ black
- ☐ rgb(241, 36, 174)
- ☐ hsl(0, 0%, 0%)
- ☐ slg(0%, 0, 0%)

Exercice

Cochez les affirmations qui sont **vraies.**

- ☐ CSS permet de mettre plusieurs ombres grâce à `box-shadow` sur un élément HTML.
- ☐ Je peux indiquer à `box-shadow` une couleur semi-transparente.
- ☐ Dans `box-shadow: 10px 10px black;`, les deux premières valeurs correspondent aux dimensions de l'ombre.
- ☐ CSS ne me permet pas de décaler mon ombre vers la gauche ni vers le haut.

Exercice

`text-shadow` permet de rajouter une ombre sur un texte et s'utilise de la même manière que `box-shadow`.

- O Vrai
- O Faux

Exercice

Pour arrondir les coins de ma boîte de manière indépendante pour chaque coin, l'ordre des paramètres est le suivant :

- O gauche - droite - bas - haut
- O gauche - haut - droite - bas
- O haut - droite - bas - gauche
- O bas - droite - gauche - haut

Exercice

La propriété CSS `background` me permet de mettre en fond :

- ☐ Une image
- ☐ Une vidéo YouTube avec la fonction `url()`
- ☐ Une couleur unie
- ☐ Un dégradé de couleurs

Exercice

`background-size` permet de gérer l'aspect de l'image de fond. Avec quelle valeur puis-je dire à mon image de prendre toute la place disponible pour ne laisser aucun espace vide ?

- O cover
- O contain
- O fill
- O expand

Exercice

Pour réaliser un dégradé de couleur, j'utilise `linear-gradient()`. Combien de paramètres prend cette fonction au minimum pour fonctionner ?

- O 3, la direction du dégradé, la couleur de départ et la couleur d'arrivée
- O 5, la direction du dégradé, la couleur de départ, la position souhaitée de la couleur de départ, la couleur d'arrivée, la position souhaitée de la couleur d'arrivée
- O 2, la couleur de départ et la couleur d'arrivée

Exercice

Une transition peut s'opérer sur n'importe quelle propriété CSS.

- O Vrai
- O Faux

Exercice

Pour mes transitions et animations, j'exprime généralement les paramètres `durée` et `délai`.

- ☐ en secondes, par exemple **1 s**
- ☐ en millisecondes, par exemple **1000 ms**
- ☐ en nanosecondes, par exemple **1000000000 nanos**
- ☐ en minutes, **0,0166667 min**

Exercice

Cochez les affirmations qui sont **vraies**

- ☐ CSS Transition me permet de jouer la transition sur plusieurs propriétés simultanément
- ☐ Une transition se déclenche avec une pseudo-classe comme :hover ou sur un changement de classe CSS
- ☐ Transition et animation référencent la même chose en CSS avec des noms différents
- ☐ La timing-function sert à déterminer le temps d'attente avant le déclenchement d'une transition / animation
- ☐ Dans une fonction @keyframes, je peux indistinctement écrire *0% {}* et *from {}* pour désigner l'état de départ de mon animation
- ☐ Il faut absolument que je mette le plus d'effets possibles sur ma page pour me démarquer des autres

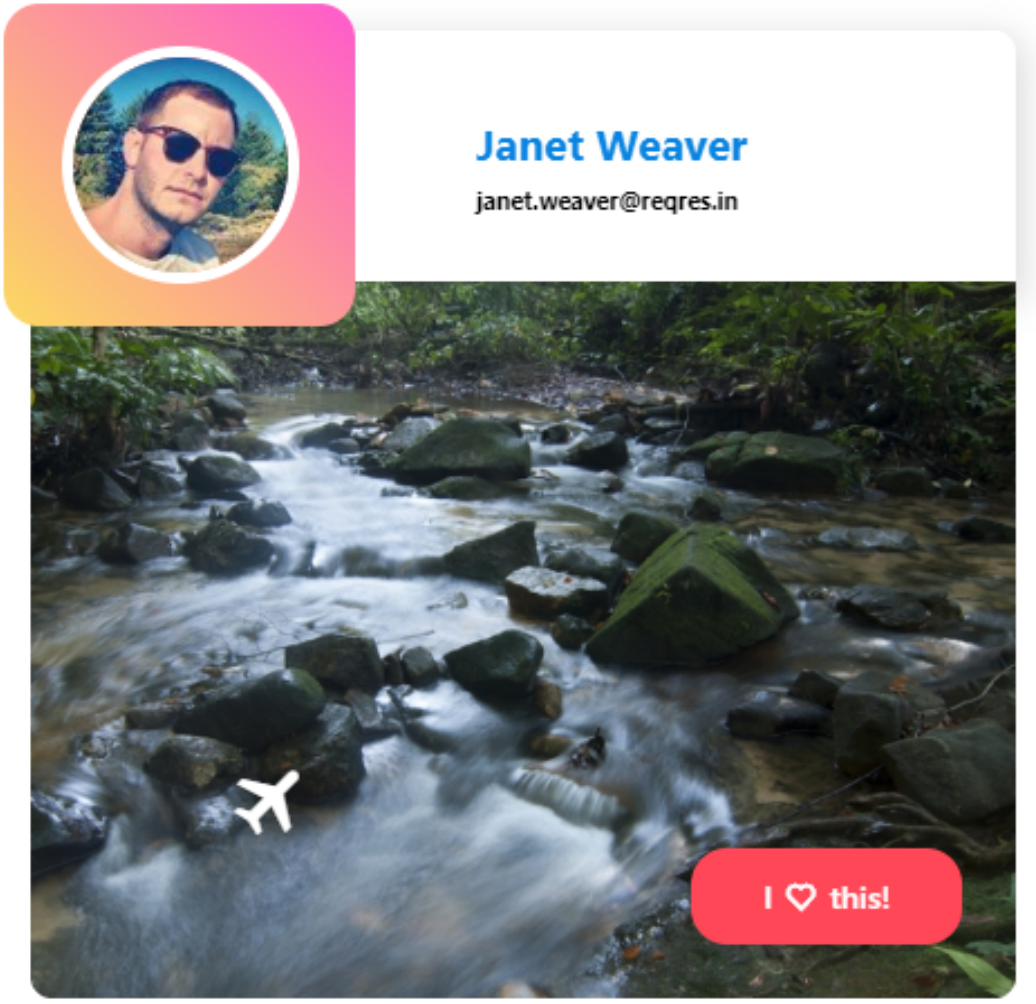
B. Exercice : Défi

Dans le cadre de la refonte de son interface, un client souhaite améliorer l'interface des profils des utilisateurs sur son site Internet.

Vous allez devoir mettre en place le CSS d'une mini carte de profil, comme nous pourrions trouver dans un réseau social tel qu'Instagram.

Cette carte comportera une photo de profil, un nom et une adresse e-mail factice. Une zone affichera également la photo de votre endroit préféré.

Voici le résultat final de ce que vous devrez obtenir :



Voici également le code source HTML qui vous servira de base :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>repl.it</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
    <link href="https://cdn.jsdelivr.net/npm/remixicon@2.3.0/fonts/remixicon.css" rel="stylesheet">
  </head>
  <body>
    <div class="card">
      <div class="my-profile">
        <div class="profile-pic-wrapper">
          <img id="profile-pic" src="https://randomuser.me/api/portraits/women/75.jpg" alt="profile-pic">
        </div>
        <div class="profile-info">
          <h2 id="profile-name">Janet Weaver</h2>
          <h5 id="profile-email">janet.weaver@reqres.in</h5>
        </div>
      </div>
```

```
      <div class="favorite-place">
        <div class="plane"><i class="ri-plane-line"></i></div>
        <button class="like-button">I <i class="ri-heart-line"></i> this!</button>
      </div>
    </div>
  </body>
</html>
```

Question

Enfin, voici les spécifications que vous allez devoir appliquer :

- `.my-profile` doit avoir tous ses coins arrondis à 12 px, ainsi qu'une ombre portée.
- `.profile-pic-wrapper` doit avoir tous ses coins arrondis à 20 px et un fond en dégradé. Ce dégradé sera tourné de 45 degrés, partira de la couleur #FBDA61 pour arriver à la couleur #FF5ACD.
- `#profile-pic` doit être arrondie complètement et avoir une bordure blanche (exprimée en HSL) de 6 px.
- `#profile-name` doit être écrit en bleu (exprimé en RGB).
- `.favorite-place` : vous pouvez mettre l'image de votre endroit préféré (à défaut, vous pouvez utiliser cette image : http://placeimg.com/640/480/nature). L'image devra être alignée sur le haut et couvrir toute la * *. Les coins bas-droite et bas-gauche seront arrondis à 12 px, les autres ne doivent pas être arrondis.
- `.plane` doit balayer l'écran depuis le coin en bas à gauche de l'image de votre endroit préféré, jusqu'au coin en haut à droite (pour cela, écrivez la propriété `animation` et la fonction `@keyframes`).
- `.like-button` doit devenir plus gros (utilisez *transform : scale(1.1, 1.1)*) au passage de la souris pour montrer à l'utilisateur qu'il peut liker l'image !

Ces consignes sont également présentes sous forme de commentaires dans le fichier **style.css** ci-dessous, qui vous servira de base.

```
/*
--- Rappel des consignes ---

- .my-profile doit avoir tous ses coins arrondis à 12px
- .my-profile doit avoir une ombre portée
- .profile-pic-wrapper doit avoir tous ses coins arrondis à 20px
- .profile-pic-wrapper doit avoir un fond en dégradé
  ce dégradé sera tourné de 45 degrés, partira de la couleur #FBDA61 pour arriver à la couleur #FF5ACD.
- #profile-pic doit être arrondie complètement et avoir une bordure blanche (exprimée en HSL) de 6px
- #profile-name doit être écrit en bleu (exprimé en RGB)
- .favorite-place doit avoir l'image de fond suivante (http://placeimg.com/640/480/nature) alignée sur le haut et qui couvre toute la <div>
- .favorite-place doit avoir ses coins bas-droite et bas-gauche arrondis à 12px, les autres ne doivent pas être arrondis.
- .plane doit balayer l'écran depuis le coin en bas à gauche de l'image de votre endroit préféré, jusqu'au coin en haut à droite (pour cela, écrivez la propriété animation et la fonction @keyframes)
- .like-button doit devenir plus gros (utilisez transform : scale(1.1, 1.1)) au passage de la souris pour montrer à l'utilisateur qu'il peut liker l'image !
*/

@keyframes wheeeee {

}

.my-profile {

}

.profile-pic-wrapper {
```

```
26
27 }
28
29 #profile-pic {
30
31 }
32
33 #profile-name {
34
35 }
36
37 .favorite-place {
38
39 }
40
41 .plane {
42   position: absolute;
43   color: white;
44   animation: 5s ease 0.1s infinite running wheeeee;
45   font-size: 3rem;
46   transform: rotate(45deg);
47 }
48
49 .like-button {
50   transition: transform 0.5s ease;
51 }
52
53 .like-button:hover,
54 .like-button:focus {
55 }
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
```

```
84
85
86
87
88 /* Ce code ne fait pas partie de l'exercice et sert juste pour l'affichage de cet exemple */
89
90 * {
91   font-family: system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI",
92     "Roboto", "Oxygen", "Ubuntu", "Cantarell", "Fira Sans",
93     "Droid Sans", "Helvetica Neue", sans-serif;
94 }
95
96 .card {
97   width: 550px;
98   margin: 80px auto;
99 }
100
101 .my-profile {
102   display: flex;
103   align-items: center;
104   width: 100%;
105   height: 150px;
106 }
107
108 .profile-info {
109   margin-left: 2rem;
110   display: flex;
111   flex-direction: column;
112   align-items: flex-start;
113   justify-content: center;
114   height: 100%;
115   flex-grow: 2;
116   padding: 0 20px;
117 }
118
119 h2, h5 {
120   margin: 4px 0;
121 }
122
123 .profile-pic-wrapper {
124   display: flex;
125   align-items: center;
126   justify-content: center;
127   flex-grow: 1;
128   height: calc(100% + 30px);
129   transform: translateX(-15px);
130   z-index: 2;
131 }
132
133 #profile-pic {
134   width: 120px;
135   height: 120px;
136 }
137
138 .favorite-place {
139   position: relative;
140   width: 100%;
141   height: 400px;
```

```
  transform: translateY(-10px);
}

.plane {
  position: absolute;
  color: white;
  font-size: 3rem;
  transform: rotate(45deg);
}

.like-button {
  position: absolute;
  bottom: 30px;
  right: 30px;
  padding: 15px 40px;
  border: 0;
  border-radius: 20px;
  background: #ff4757;
  color: #fff;
  font-size: 1.1rem;
  font-weight: 600;
  display: flex;
  align-items: center;
  justify-content: center;
  cursor: pointer;
}

.like-button > i {
  margin: 0 7px;
}
```

Note importante : du code est déjà présent sur l'exemple, il n'est pas nécessaire de le modifier, il sert pour l'affichage.