CSS Avancées (Animations)

> **Note importante : Ce document est conçu comme un guide de référence, mais n'est pas exhaustif. La capacité à rechercher des informations complémentaires et à résoudre des problèmes de manière autonome est une compétence essentielle pour tout développeur. Si vous rencontrez des difficultés ou si certaines étapes ne fonctionnent pas comme prévu, n'hésitez pas à consulter d'autres ressources (documentation officielle, tutoriels YouTube, forums spécialisés comme Stack Overflow). Apprendre à trouver l'information manquante et à débugger par vous-même est probablement la compétence la plus précieuse que vous développerez**

Introduction aux animations CSS avec @keyframes

Les animations CSS permettent de créer des effets visuels plus complexes et plus dynamiques que les simples transitions. Contrairement aux transitions qui se limitent à un changement d'état entre deux points, les animations CSS permettent de définir plusieurs étapes intermédiaires et offrent un contrôle précis sur chaque phase de l'animation.

La règle @keyframes

La règle `@keyframes` est au cœur du système d'animation CSS. Elle permet de définir les étapes (ou images clés) d'une séquence d'animation.

```
@keyframes nomDeLAnimation {
  0% {
```

```
  /* Styles au début de l'animation */
  opacity: 0;
  transform: translateY(-20px);
 }

 50% {
  /* Styles à la moitié de l'animation */
  opacity: 0.5;
  transform: translateY(10px);
 }

 100% {
  /* Styles à la fin de l'animation */
  opacity: 1;
  transform: translateY(0);
 }
}
```

Points importants sur @keyframes

- Le nom de l'animation défini après `@keyframes` sera utilisé pour référencer cette animation dans les propriétés d'un élément
- Les pourcentages (0%, 50%, 100%) représentent la progression temporelle de l'animation
- On peut également utiliser les mots-clés `from` (équivalent à 0%) et `to` (équivalent à 100%)
- Vous pouvez définir autant d'étapes intermédiaires que nécessaire (20%, 33%, 75%, etc.)
- Chaque étape peut modifier plusieurs propriétés CSS simultanément

```
@keyframes rotation {
 from {
  transform: rotate(0deg);
 }

 to {
  transform: rotate(360deg);
```

```
  }
}

.element-rotatif {
  animation: rotation 2s linear infinite;
}
```

Les propriétés d'animation

Une fois l'animation définie avec `@keyframes`, il faut l'appliquer à un élément en utilisant les propriétés suivantes :

1. animation-name

Cette propriété spécifie le nom de l'animation à utiliser (le nom défini dans la règle @keyframes).

```
.element {
  animation-name: nomDeLAnimation;
}
```

2. animation-duration

Définit la durée totale d'un cycle complet de l'animation.

```
.element {
  animation-name: nomDeLAnimation;
  animation-duration: 2s; /* 2 secondes */
}
```

3. animation-timing-function

Définit comment l'animation progresse au cours du temps en spécifiant une courbe d'accélération.

```
.element {
  animation-name: nomDeLAnimation;
  animation-duration: 2s;
```

```
  animation-timing-function: ease-in-out;
}
```

Les valeurs possibles sont similaires à celles des transitions :

- `linear` : vitesse constante
- `ease` : accélération douce (par défaut)
- `ease-in` : lent au début, puis accélère
- `ease-out` : rapide au début, puis ralentit
- `ease-in-out` : lent au début, rapide au milieu, lent à la fin
- `cubic-bezier(n,n,n,n)` : courbe de Bézier personnalisée
- `steps(n, start|end)` : progression par étapes discrètes

4. animation-delay

Spécifie un délai avant le démarrage de l'animation.

```
.element {
  animation-name: nomDeLAnimation;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-delay: 0.5s; /* Attend 0.5 secondes avant de démarrer */
}
```

5. animation-iteration-count

Définit combien de fois l'animation doit être jouée.

```
.element {
  animation-name: nomDeLAnimation;
  animation-duration: 2s;
  animation-iteration-count: 3; /* Joue l'animation 3 fois */
}
```

Valeurs possibles :

- Nombre entier positif (1, 2, 5, etc.)
- `infinite` : répète l'animation indéfiniment

6. animation-direction

Définit si l'animation doit jouer en avant, en arrière ou en alternant les deux directions.

```
.element {
  animation-name: nomDeLAnimation;
  animation-duration: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

Valeurs possibles :

- `normal` : joue l'animation normalement (de 0% à 100%)
- `reverse` : joue l'animation en sens inverse (de 100% à 0%)
- `alternate` : alterne entre normal et reverse à chaque itération
- `alternate-reverse` : comme alternate mais commence par reverse

7. animation-fill-mode

Détermine comment les styles sont appliqués avant et après l'exécution de l'animation.

```
.element {
  animation-name: nomDeLAnimation;
  animation-duration: 2s;
  animation-fill-mode: forwards;
}
```

Valeurs possibles :

- `none` : l'élément conserve son style original avant et après l'animation
- `forwards` : l'élément conserve les styles définis par la dernière image clé après la fin de l'animation
- `backwards` : applique les styles de la première image clé pendant la période définie par animation-delay
- `both` : combine les effets de forwards et backwards

8. animation-play-state

Permet de mettre en pause ou de reprendre une animation.

```
.element {
  animation-name: nomDeLAnimation;
  animation-duration: 2s;
  animation-play-state: running; /* ou paused */
}

.element:hover {
  animation-play-state: paused; /* Met en pause l'animation au survol */
}
```

Syntaxe raccourcie

Toutes ces propriétés peuvent être combinées en une seule déclaration avec la propriété raccourcie `animation`.

```
.element {
  /* animation: name duration timing-function delay iteration-count direction fill-mode play-state */
  animation: nomDeLAnimation 2s ease-in-out 0.5s infinite alternate forwards running;
}
```

L'ordre des valeurs n'est pas strict, mais il est recommandé de suivre l'ordre indiqué pour une meilleure lisibilité.

Animations multiples

Il est possible d'appliquer plusieurs animations à un même élément en les séparant par des virgules.

```
.element {
  animation:
    rotation 2s linear infinite,
    changementCouleur 5s ease-in-out alternate infinite;
}
```

```css
@keyframes rotation {
  from { transform: rotate(0deg); }
  to { transform: rotate(360deg); }
}

@keyframes changementCouleur {
  0% { background-color: red; }
  50% { background-color: blue; }
  100% { background-color: green; }
}
```

Exemples d'animations courantes

Animation de pulsation

```css
@keyframes pulse {
  0% {
    transform: scale(1);
  }
  50% {
    transform: scale(1.1);
  }
  100% {
    transform: scale(1);
  }
}

.bouton-pulse {
  animation: pulse 2s ease-in-out infinite;
}
```

Animation de rebond

```css
@keyframes bounce {
  0%, 20%, 50%, 80%, 100% {
    transform: translateY(0);
```

```
  }
  40% {
    transform: translateY(-30px);
  }
  60% {
    transform: translateY(-15px);
  }
}

.element-rebond {
  animation: bounce 2s infinite;
}
```

Animation de fondu enchaîné

```
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

.element-apparition {
  animation: fadeIn 1s ease-out forwards;
}
```

Performances et optimisation

Pour des animations fluides, privilégiez l'animation des propriétés qui n'entraînent pas de reflow (recalcul complet de la mise en page) :

- **Propriétés peu coûteuses** : `transform`, `opacity`, `filter`
- **Propriétés coûteuses à éviter** : `width`, `height`, `margin`, `padding`, `top`, `left`, etc.

Utilisez `will-change` avec parcimonie pour indiquer au navigateur qu'une propriété va être animée :

```css
.element {
  will-change: transform, opacity;
}
```

Compatibilité et préfixes

Les animations CSS sont bien supportées par tous les navigateurs modernes, mais pour assurer une compatibilité maximale, vous pouvez utiliser des préfixes vendeurs :

```css
@-webkit-keyframes nomDeLAnimation { /* ... */ }
@keyframes nomDeLAnimation { /* ... */ }

.element {
  -webkit-animation: nomDeLAnimation 2s;
  animation: nomDeLAnimation 2s;
}
```

Cependant, l'utilisation d'outils comme Autoprefixer est recommandée pour gérer automatiquement ces préfixes.

Interactions avec les media queries

Les animations peuvent être adaptées en fonction de la taille de l'écran ou des préférences utilisateur :

```css
@media (max-width: 768px) {
  .element {
    animation-duration: 1s; /* Animation plus rapide sur mobile */
  }
}

@media (prefers-reduced-motion: reduce) {
  .element {
    animation: none; /* Désactive l'animation si l'utilisateur préfère réduire les mouvements */
```

```
  }
}
```

> **Protection intellectuelle :** Ce cours, est une création originale de **MANJAL Younes** et est protégé par le droit d'auteur conformément au Code de la propriété intellectuelle français (article L.111-1). Toute reproduction, modification, diffusion ou utilisation, même partielle, sans autorisation écrite préalable est strictement interdite. L'intelligence artificielle a été utilisée uniquement comme outil d'assistance pour la correction orthographique et la reformulation de certaines phrases, sous ma supervision et validation. Le contenu pédagogique et intellectuel reste entièrement ma création personnelle.