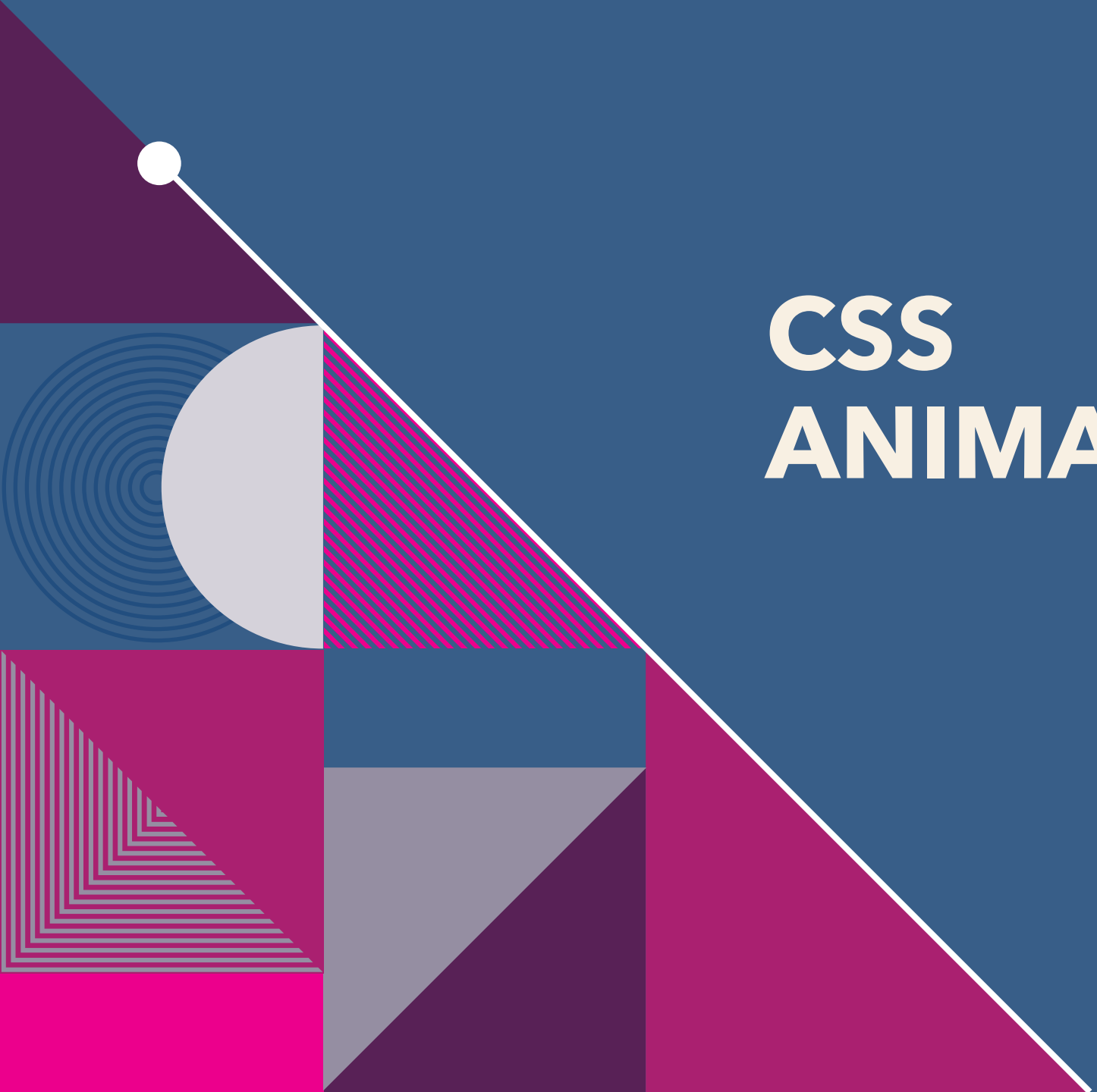


CSS
ANIMATIONS CSS

CSS : ANIMATIONS INTRODUCTION

Pourquoi utiliser des animations dans un projet web ?

- Améliorer l'expérience utilisateur : Les interfaces sont plus intuitives et agréables
- Permettent d'attirer l'attention de l'utilisateur là où on le souhaite
- Feedback visuel : Indiquent les interactions possibles (ex : animation au survol)

Animations les plus courantes :

- Transitions : Permettent d'adoucir les changements d'état (boutons, liens)
- Animations : Effets continus ou répétés (bannière, icones)
- Micro-animations : petites animations pour améliorer l'expérience utilisateur (barre de chargement, notifications).

CSS : ANIMATIONS BONNES PRATIQUES

- **Simplicité** : Éviter les animations trop complexes ou distrayantes.
- **Performance** : Optimiser les animations pour qu'elles ne ralentissent pas le site.
- **Accessibilité** : S'assurer que les animations n'affectent pas négativement les utilisateurs souffrant de troubles de la vision ou de mouvements.

CSS : ANIMATIONS
TYPES D'ANIMATIONS

Transitions CSS

- Permettent de créer des animations simples en changeant les propriétés CSS sur un événement (hover, focus).
- Exemples : Changement de couleur, déplacement, redimensionnement.

```css
.example {
    transition: all 0.3s ease;
}

.example:hover {
    background-color: blue;
    transform: scale(1.1);
}
```

CSS : ANIMATIONS
TYPES D'ANIMATIONS

Animations avec Keyframes

- Utilisation de la propriété « @keyframes » pour définir des étapes d'une animation.
- Permettent des animations plus complexes

```
@keyframes example-animation {
    from {
        opacity: 0;
        transform: translateY(-10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

.example_animation {
    animation: example-animation 0.5s ease-out;
}
```

CSS : ANIMATIONS
TYPES D'ANIMATIONS

Micro intéractions

- Petites animations visant à améliorer l'interaction utilisateur.
- Utilisées pour fournir un feedback immédiat à une action de l'utilisateur.

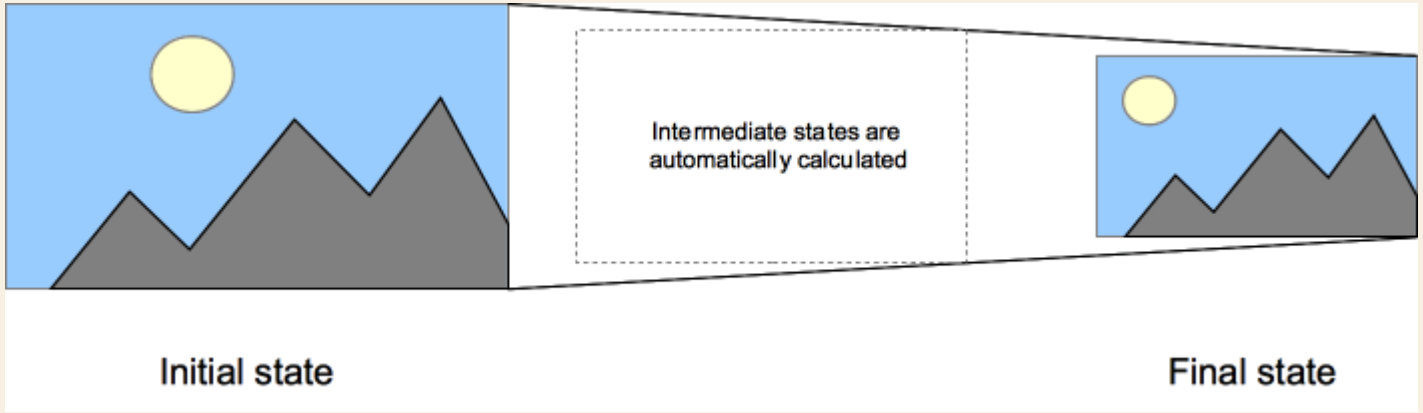
```
@keyframes example-animation {
    from {
        opacity: 0;
        transform: translateY(-10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

.example_animation {
    animation: example-animation 0.5s ease-out;
}
```

CSS : ANIMATIONS TRANSITIONS CSS

Les transitions CSS permettent de contrôler la vitesse d'animation lorsque les propriétés CSS sont modifiées.

Les animations qui utilisent des transitions entre deux états sont souvent appelées transitions implicites car l'état initial et l'état final sont définis implicitement par le navigateur.



CSS : ANIMATIONS TRANSITIONS CSS

Les transitions CSS vous permettent de choisir :

- les propriétés à animer en les listant explicitement
- le début de l'animation
- la durée de l'animation
- la façon dont la transition s'exécutera

Attention, toutes les propriétés ne peuvent pas être animées (voir : https://developer.mozilla.org/fr/docs/Web/CSS/CSS_animated_properties)

CSS : ANIMATIONS TRANSITIONS CSS

Voici les propriétés à définir :

- « transition-property » : Permet de définir les propriétés CSS que l'on souhaite animer.
- « transition-duration » : Durée de la transition
- « transition-timing-function » : Définit une fonction qui décrit la façon dont les valeurs intermédiaires sont calculées. Ex : 'ease', 'linear', 'ease-in', 'ease-out', 'ease-in-out' (voir fonctions de temporisation : https://developer.mozilla.org/fr/docs/Web/CSS/easing-function)
- « transition-delay » : Cette propriété indique le temps à attendre entre le moment où la propriété est modifiée et le début de la transition.
- La super propriété « transition » reprend ces différentes propriétés :

```
div {
    transition: <property> <duration> <timing-function> <delay>;
  }
```

CSS : ANIMATIONS
TRANSITIONS CSS : EXEMPLES

- **Création d'un bouton avec transition de background-color**

```
<style>
        .button {
            background-color: #008CBA;
            color: white;
            padding: 15px 32px;
            text-align: center;
            font-size: 16px;
            border: none;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }

        .button:hover {
            background-color: #005f6b;
        }
    </style>
    <button class="button">Hover me</button>
```

CSS : ANIMATIONS
TRANSITIONS CSS : EXEMPLES

- **Transition sur la taille d'un élément**

```
<style>
        .box2 {
            width: 100px;
            height: 100px;
            background-color: red;
            transition: transform 0.3s ease-in-out;
        }

        .box2:hover {
            transform: scale(1.5);
        }
    </style>
    <div class="box2"></div>
```

CSS : ANIMATIONS
TRANSITIONS CSS : MISE EN PRATIQUE

- **Exercice 1 : Créer un bouton avec une transition de couleur**
  - Créer un fichier HTML avec un bouton
  - Ajouter une transition de couleur au survol du bouton
- **Exercice 2 : Créer une carte avec une transition de taille**
  - Créer une carte avec du texte et une image
  - Ajouter une transition pour agrandir la carte au survol
- **Exercice 3 : Créer une liste de navigation avec transitions**
  - Créer une liste de navigation.
  - Ajouter des transitions pour changer la couleur des liens et souligner les éléments au survol.

CSS : ANIMATIONS
TRANSITIONS CSS : RESSOURCES

MDN Web Docs : CSS Transitions
https://developer.mozilla.org/fr/docs/Web/CSS/CSS_transitions/Using_CSS_transitions

CSS-Tricks : Guide des Transitions CSS :
https://css-tricks.com/an-interactive-guide-to-css-transitions/

CSS : ANIMATIONS
ANIMATIONS AVEC KEYRAMES

La règle « @keyframes » permet de définir les étapes qui composent la séquence d'une animation CSS. Cela permet de contrôler une animation plus finement que ce qu'on pourrait obtenir avec les transitions.

Il est possible de manipuler la règle « @keyframes » via JavaScript et le CSSOM, notamment avec l'interface « CSSKeyframesRule ».

Les keyframes permettent ainsi de créer des animations complexes.

CSS : ANIMATIONS KEYRAMES : SYNTAXE

La règle d'animation « @keyframes » est toujours suivie de l'identifiant de l'animation créée.

On va ensuite définir les états des différentes étapes de l'animation.

```
@keyframes animation-name {
    from { /* styles initiaux */ }
    to { /* styles finaux */ }
}

/* ou */

@keyframes animation-name {
    0% { /* styles initiaux */ }
    100% { /* styles finaux */ }
}
```

CSS : ANIMATIONS KEYRAMES : PROPRIÉTÉS

- **animation-name** : Le nom de l'animation.
- **animation-duration** : La durée de l'animation.
- **animation-timing-function** : La fonction de timing.
- **animation-delay** : Le délai avant le début de l'animation.
- **animation-iteration-count** : Le nombre de fois que l'animation doit être jouée.
- **animation-direction** : La direction de l'animation (normal, reverse, alternate, alternate-reverse).
- **animation-fill-mode** : Le style appliqué à l'élément avant et après l'animation (none, forwards, backwards, both).
- **animation-play-state** : Le statut de l'animation (running, paused).

CSS : ANIMATIONS KEYRAMES : EXEMPLES

```css
@keyframes slidein {
    from {
        transform: translateX(-100%);
    }
    to {
        transform: translateX(0);
    }
}

.animated-element {
    animation-name: slidein;
    animation-duration: 2s;
    animation-timing-function: ease-in-out;
    animation-iteration-count: infinite;
}
```

CSS : ANIMATIONS KEYRAMES : EXEMPLES

- **Animation de déplacement** : faire glisser un élément de gauche à droite

```
    <style>
        @keyframes slidein {
            from {
                transform: translateX(-100%);
            }
            to {
                transform: translateX(0);
            }
        }

        .slider {
            width: 100px;
            height: 100px;
            background-color: red;
            animation: slidein 2s ease-in-out infinite;
        }
    </style>
    <div class="slider"></div>
```

CSS : ANIMATIONS KEYRAMES : EXEMPLES

- **Animation de rotation** : faire tourner un élément

```
<style>
    @keyframes rotate {
        from {
            transform: rotate(0deg);
        }
        to {
            transform: rotate(360deg);
        }
    }

    .rotator {
        width: 100px;
        height: 100px;
        background-color: blue;
        animation: rotate 2s linear infinite;
    }
</style>
<div class="rotator"></div>
```

CSS : ANIMATIONS
KEYFRAMES : MISE EN PRATIQUE

- **Exercice 1 : Créer une animation de clignotement**
  - Créer une animation où un texte clignote en changeant de couleur.
- **Exercice 2 : Créer une animation de rebond**
  - Créer une animation où un élément rebondit de haut en bas.
- **Exercice 3 : Créer une animation de chargement**
  - Créer une animation pour un indicateur de chargement (loader).

CSS : ANIMATIONS
KEYFRAMES : RESSOURCES

MDN Web Docs : CSS Animations
https://developer.mozilla.org/fr/docs/Web/CSS/animation

CSS-Tricks : Guide des Animations CSS :
https://css-tricks.com/almanac/properties/a/animation/

CSS : ANIMATIONS
TECHNIQUES AVANCÉES

Combinaison de Transitions et Keyframes

- Utiliser des transitions pour les changements d'état simples et des keyframes pour des animations plus complexes.
- Exemple : Animer la couleur avec une transition et la position avec des keyframes

```
    <style>
        .combo-element {
            background-color: red;
            transition: background-color 0.5s ease;
            animation: move 2s infinite alternate;
        }

        .combo-element:hover {
            background-color: blue;
        }

        @keyframes move {
            0% {
                transform: translateX(0);
            }

            100% {
                transform: translateX(100px);
            }
        }
    </style>
    <p class="combo-element">Lorem ipsum dolor sit amet consectetur, adipisicing elit.
Libero ut error, dolor reiciendis dicta ratione minus voluptates sed cupiditate ea.</p>
```

CSS : ANIMATIONS
TECHNIQUES AVANCÉES

Animations multi-étapes avec les Keyframes

- Définir plusieurs étapes dans une animation pour créer des mouvements complexes.
- Exemple :

```
<h3 class="multi-step-element">Keyframes multi-étapes</h3>
    <style>
        @keyframes multi-step {
            0% {
                transform: translateY(0);
            }

            25% {
                transform: translateY(-50px);
            }

            50% {
                transform: translateY(0);
            }

            75% {
                transform: translateY(50px);
            }

            100% {
                transform: translateY(0);
            }
        }

        .multi-step-element {
            animation: multi-step 4s ease-in-out infinite;
        }
```

CSS : ANIMATIONS
TECHNIQUES AVANCÉES

Animations de plusieurs propriétés

- Animer plusieurs propriétés simultanément pour créer des effets dynamiques.

```
    <style>
        .combo-element {
            background-color: red;
            transition: background-color 0.5s ease;
            animation: move 2s infinite alternate;
        }

        .combo-element:hover {
            background-color: blue;
        }

        @keyframes move {
            0% {
                transform: translateX(0);
            }

            100% {
                transform: translateX(100px);
            }
        }
    </style>
    <p class="combo-element">Lorem ipsum dolor sit amet consectetur, adipisicing elit.
Libero ut error, dolor reiciendis dicta ratione minus voluptates sed cupiditate ea.</p>
```

CSS : ANIMATIONS
TECHNIQUES AVANCÉES

Contrôle des animations avec JavaScript

Nous pouvons utiliser JavaScript pour déclencher, arrêter ou modifier les animations.

L'exemple suivant est là pour illustrer, mais nous pourrons revenir dessus lorsque nous verrons JavaScript.

```
<style>
        .controlled-element {
            width: 100px;
            height: 100px;
            background-color: purple;
            animation: controlled-animation 2s infinite;
            animation-play-state: paused;
        }

        @keyframes controlled-animation {
            from {
                transform: translateX(0);
            }

            to {
                transform: translateX(200px);
            }
        }
    </style>
```

```
<div class="controlled-element" id="animateMe"></div>
    <button onclick="startAnimation()">Start</button>
    <button onclick="stopAnimation()">Stop</button>
```

```
<script>
        function startAnimation() {
            document.getElementById('animateMe').style.animationPlayState = 'running';
        }

        function stopAnimation() {
            document.getElementById('animateMe').style.animationPlayState = 'paused';
        }
    </script>
```

CSS : ANIMATIONS
BONNES PRATIQUES

Performance des Animations

- Utiliser « transform » et « opacity » pour des animations plus fluides et performantes.
- Éviter d'animer les propriétés qui déclenchent des recalculs de layout (comme « width », « height », « margin »).

Accessibilité

- Fournir des options pour désactiver les animations pour les utilisateurs souffrant de troubles (épilepsie).
- Utiliser les media queries pour détecter les préférences utilisateur.

```
@media (prefers-reduced-motion: reduce) {
    .animated-element {
        animation: none;
        transition: none;
    }
}
```

CSS : ANIMATIONS
BONNES PRATIQUES

Testing et Debugging

- Utiliser les outils de développement du navigateur pour inspecter et déboguer les animations.
- Vérifier l'impact des animations sur la performance du site.

Expérience Utilisateur

- Garder les animations simples et intuitives.
- Utiliser les animations pour améliorer l'expérience utilisateur, pas pour distraire.

Cohérence et identité visuelle

- Maintenir une cohérence dans les animations pour renforcer l'identité visuelle du site.
- Suivre les lignes directrices de la marque pour les styles d'animation.

CSS : ANIMATIONS
TECHNIQUES AVANCÉES : RESSOURCES

MDN Web Docs : Performance animations CSS

- https://developer.mozilla.org/en-US/docs/Web/Performance/CSS_JavaScript_animation_performance
- https://developer.mozilla.org/fr/docs/Learn/Performance/CSS

A11Y Project : Animations et Accessibilité

- https://www.a11yproject.com/posts/design-accessible-animation/

CSS : ANIMATIONS
TECHNIQUES AVANCÉES :TD

Exercice 1 : Améliorer une animation existante pour la performance

- Identifier une animation inefficace et la réécrire pour utiliser « transform » et « opacity ».

```
.inefficient-animation {
    width: 100px;
    height: 100px;
    background-color: orange;
    animation: inefficient 2s infinite;
}

@keyframes inefficient {
    0% { width: 100px; }
    50% { width: 200px; }
    100% { width: 100px; }
}
```

CSS : ANIMATIONS
TECHNIQUES AVANCÉES :TD

Exercice 2 : Ajouter des options d'accessibilité pour une animation

- Utiliser les media queries pour désactiver les animations si l'utilisateur préfère réduire le mouvement.

```css
.accessible-animation {
    width: 100px;
    height: 100px;
    background-color: yellow;
    animation: bounce 2s infinite;
}

@keyframes bounce {
    0%, 100% { transform: translateY(0); }
    50% { transform: translateY(-50px); }
}
```