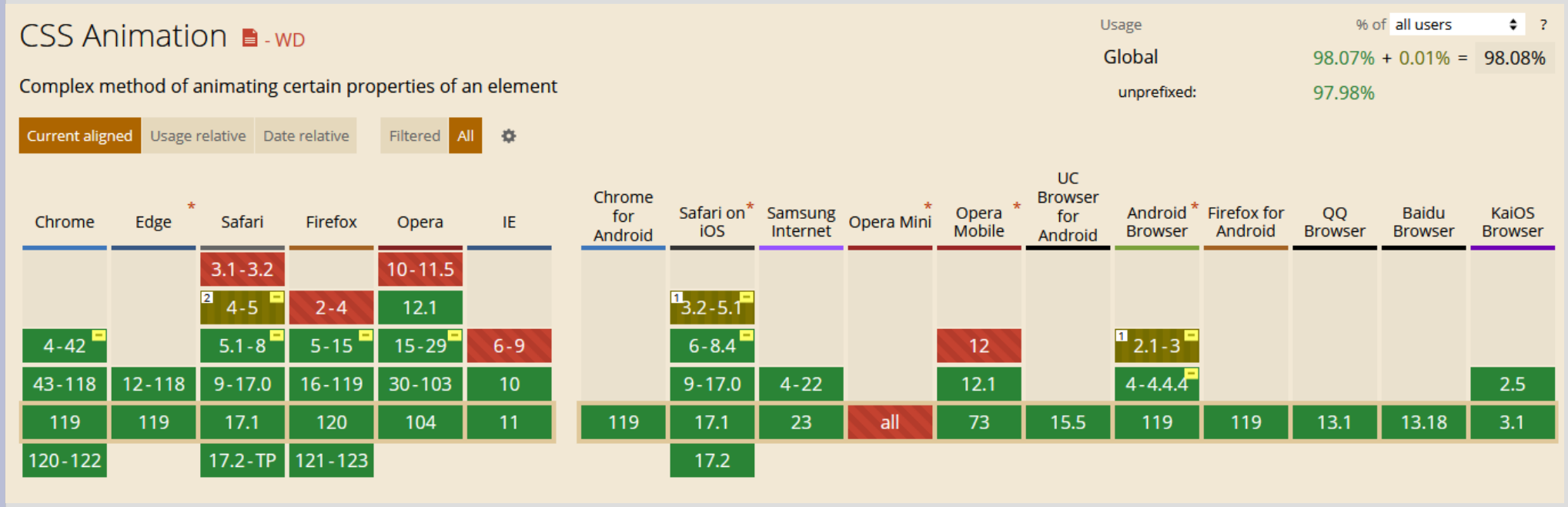
CSS3 - animation

- Les animations avec CSS
- Support par les navigateurs
- Propriétés CSS associées
- Contrôler une animation
- Utilisation de JavaScript
- Outils de test
- Applications (slideshow, sprite,...)

Les animations avec CSS

- Les **transitions** avec CSS ne permettent que des animations simples :
  - limitées à un état de début et de fin pour la définir
  - limitées dans le contrôle (déclenchée par des interactions souris) si on n'utilise pas JavaScript
- Pour avoir un contrôle plus abouti, les animations CSS ont été proposées. On peut définir **autant d'états différents** que nécessaires et on peut préciser la **répétition** de l'animation ainsi que son **état final** et son **délai de déclenchement**.

Support par les navigateurs



Les propriétés CSS3 liées aux **animations** sont supportées depuis de nombreuses versions

https://caniuse.com/#search=css3 animation

@keyframes

- Pour **définir les différents états**, on utilise des **keyframes** (traduit par images-clés ou états-clés)
- Les **étapes** de l'animation vont être identifiées par un **pourcentage** ou l'un des mots-clés **from** qui équivaut à 0 % et **to** qui équivaut à 100 %
- **Pour chaque étape**, on définit les **règles** qui subissent une **modification**
- On donne un **nom** à cette **@keyframes** pour pouvoir **l'associer avec une animation**

Exemple 1 (HTML)

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Animations avec CSS</title>
        <link type="text/css" rel="stylesheet" href="animation.css" />
    </head>
    <body>
        <header>
            <img class="logo" alt="Logo" src="html5.png" />
        </header>
        <main>
            <h1>Les animations avec CSS</h1>
            <section class="objet">
                <h3>CSS3</h3>
                <p>Animation</p>
            </section>
        </main>
    </body>
</html>
```

Exemple 1 (CSS)

```css
.objet {
    background-color: aquamarine;
    color: orange;
    font-size:2em;
    width: 200px;
    height: 200px;
    border: 3px solid black;
    border-radius: 50%;
    text-align: center;
}

.logo{
    animation-name: logoHTML5;
    animation-duration: 4s;
}

@keyframes logoHTML5 {
    from {
        transform: rotateY(0);
    }
    to {
        transform: rotateY(1turn);  /* équivalent à rotateY(360deg); */
    }
}
```

animation-name

- Pour **appliquer une animation** à un élément, il suffit de lui définir la propriété **animation-name** en précisant le **nom de la @keyframes** à utiliser
- Puis on indique la **durée** avec la propriété **animation-duration**
- La propriété animation-name peut prendre **comme valeur** un ou plusieurs noms d'animation ou bien encore la valeur none pour, par exemple, annuler un héritage

animation-duration

- La propriété **animation-duration** permet de définir la durée de l’animation
- Elle n'accepte que des **durées positives**, exprimées en secondes ou millisecondes
- Contrairement à une transition, **une animation débute lors du chargement de la page** et non pas uniquement lors d'un changement d'état

animation-timing-fonction

- Comme dans les transitions, il est possible de **contrôler la vitesse et l'accélération** de l'animation.
- On utilise la propriété **animation-timing-function**
- Les **valeurs** sont identiques : ease (par défaut), linear, ease-in, ease-out, ease-in-out ou encore cubic-bezier(n,n,n,n) (courbe de Bézier spéciale)
- Mais dans les animations, on peut la définir de manière **globale** ou modifiée **individuellement** pour chaque keyframe

animation-delay

- Il est possible de définir un **départ retardé** de l'animation avec la propriété **animation-delay** qui prend comme valeur une durée.
- Si la **valeur** est **positive**, alors l'animation débutera après n secondes
- Si la **valeur** est **négative**, l'animation débutera instantanément mais à partir de la position qu'elle aurait dû avoir après n secondes

Exemple 2

```css
.objet {
    background-color: aquamarine;
    color: orange;
    font-size:2em;
    width: 200px;
    height: 200px;
    border: 3px solid black;
    border-radius: 50%;
    text-align: center;

    animation-name: roule;
    animation-duration: 4s;
    animation-delay: 2s;
}

@keyframes roule {
    0% {transform: translateX(0) rotate(0);}
    20% {transform: translateX(-50px) rotate(-45deg);}
    100% {transform: translateX(800px) rotate(720deg);}
}

.logo{
    animation-name: logoHTML5;
    animation-duration: 4s;
}

@keyframes logoHTML5 {
    from {transform: rotateY(0);}
    to {transform: rotateY(1turn);  /* équivalent à rotateY(360deg); */}
}
```

animation-iteration-count

- La propriété **animation-iteration-count** définit le nombre de répétitions de l'animation.
- C'est une chose impossible à réaliser avec les transitions.
- Il existe même le mot-clé **infinite** qui permet de répéter l'animation sans fin
- Si un **délai** a été défini par animation-delay, il ne sera pris en compte que lors de la première itération

Exemple 3

```css
.objet {
    background-color: aquamarine;
    color: orange;
    font-size:2em;
    width: 200px;
    height: 200px;
    border: 3px solid black;
    border-radius: 50%;
    text-align: center;

    animation-name: roule;
    animation-duration: 4s;
    animation-iteration-count: 2;
}

@keyframes roule {
    0% {transform: translateX(0) rotate(0);}
    20% {transform: translateX(-50px) rotate(-45deg);}
    100% {transform: translateX(800px) rotate(720deg);}
}

.logo{
    animation-name: logoHTML5;
    animation-duration: 4s;
    animation-iteration-count: infinite;
}

@keyframes logoHTML5 {
    from {transform: rotateY(0);}
    to {transform: rotateY(1turn);  /* équivalent à rotateY(360deg); */}
}
```

animation-direction

- La propriété **animation-direction** permet de définir si la répétition de l'animation se déroule normalement ou à l’envers ou si elle alterne
- Les valeurs possibles pour cette propriété sont :
  - **normal** : par défaut, on joue normalement l’animation de 0 % à 100 %
  - **reverse** : à l’envers, de 100 % à 0 %
  - **alternate** : d’abord normal, ensuite reverse
  - **alternate-reverse** : d’abord reverse, ensuite normal

Exemple 4

```
.objet {
    background-color: aquamarine;
    color: orange;
    font-size:2em;
    width: 200px;
    height: 200px;
    border: 3px solid black;
    border-radius: 50%;
    text-align: center;

    animation-name: roule;
    animation-duration: 4s;
    animation-delay: 2s;
    animation-iteration-count: 2;
    animation-direction: alternate;
    animation-fill-mode: both;
}

@keyframes roule {
    0% {transform: translateX(400px) rotate(0);}
    20% {transform: translateX(-50px) rotate(-405deg);}
    100% {transform: translateX(800px) rotate(720deg);}
}
```

animation-fill-mode

- **Par défaut**, une animation ne modifie pas les propriétés sur lesquelles elle s'applique avant le début (durant la animation-delay) et après la fin de celle-ci.
- La propriété **animation-fill-mode** permet de modifier ce comportement. Les valeurs possibles sont :
  - **none** : comportement par défaut
  - **backwards** : applique les propriétés définies dans l'état 0% avant le début de l'animation, y compris le délai
  - **forwards** : applique les propriétés définies dans l'état final de l'animation une fois l'animation terminée (NB : selon la direction de l’animation : 100% ou 0%)
  - **both** : combinaison de forwards et backwards

Exemple 5

```
.objet {
    background-color: aquamarine;
    color: orange;
    font-size:2em;
    width: 200px;
    height: 200px;
    border: 3px solid black;
    border-radius: 50%;
    text-align: center;

    animation-name: roule;
    animation-duration: 4s;
    animation-iteration-count: 2;
    animation-delay: 2s;
    animation-fill-mode: none;
}

@keyframes roule {
    0% {transform: translateX(400px) rotate(0);}
    20% {transform: translateX(-50px) rotate(-405deg);}
    100% {transform: translateX(800px) rotate(720deg);}
}
```

Contrôler une animation (animation-play-state)

- Par défaut, une animation sur un élément HTML démarre dès qu'on a défini dans la propriété **animation-name** la valeur du **@keyframes** (éventuellement après un certain délai)
- On peut déjà contrôler avec la propriété **animation-play-state** qui peut prendre deux valeurs : **paused** ou **running** (par défaut)
- Pour un meilleur contrôle, il faudra utiliser **JavaScript** et gérer les événements

Exemple 6

```
.logo{
    animation-name: logoHTML5;
    animation-duration: 4s;
    animation-iteration-count: infinite;

    animation-play-state: running;
}

.logo:hover {
    animation-play-state: paused;
}

@keyframes logoHTML5 {
    from {transform: rotateY(0);}
    to {transform: rotateY(1turn);  /* équivalent à rotateY(360deg); */}
}
```

animation

- Il existe aussi la propriété **animation** qui est la **notation raccourcie** reprenant toutes les autres propriétés, dans l'ordre suivant :
  - animation-name
  - animation-duration
  - animation-timing-function
  - animation-delay
  - animation-iteration-count
  - animation-direction
  - animation-fill-mode
  - animation-play-state

animation

- Bien entendu, on n’est pas obligé de préciser toutes les valeurs. Si une valeur n’est pas indiquée, la valeur par défaut sera utilisée.
- Même si c’est une bonne pratique de passer les propriétés dans l’ordre, cela n’a pas vraiment d’importance. Il faut juste veiller à placer **d’abord la durée** (animation-duration) **et ensuite le délai** (animation-delay) puisqu’elles ont la même unité
- Ne pas oublier le **nom** (animation-name) qui est nécessaire pour jouer l’animation

JavaScript et animations

- On dispose de 3 événements JavaScript pour contrôler le déroulement d'une animation :
  - **animationstart** : se produit quand une animation démarre
  - **animationend** : se produit quand une animation se termine
  - **animationiteration** : se produit quand une animation est répétée (si animation-iteration-count > 1)

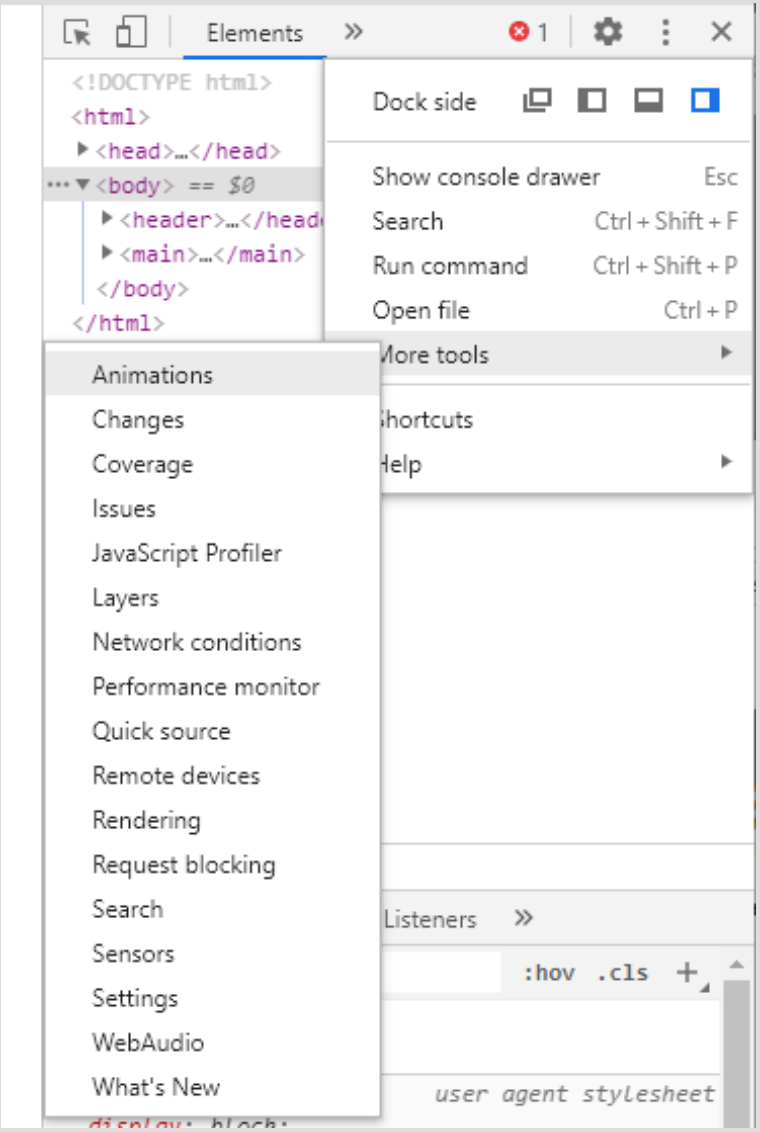
Exemple 7 - CSS

```css
main h1 {
    /*animation-name: deplaceTitre;*/
    animation-duration: 3s;
    animation-delay: 1s;
    animation-iteration-count: 3;
    animation-fill-mode: forwards;
}

@keyframes deplaceTitre {
    0%   { transform: translateX(-50px); }
    50%  { transform: translateX(250px); }
    100% { transform: translateX(0px);   }
}
```

Exemple 7 – JavaScript & HTML

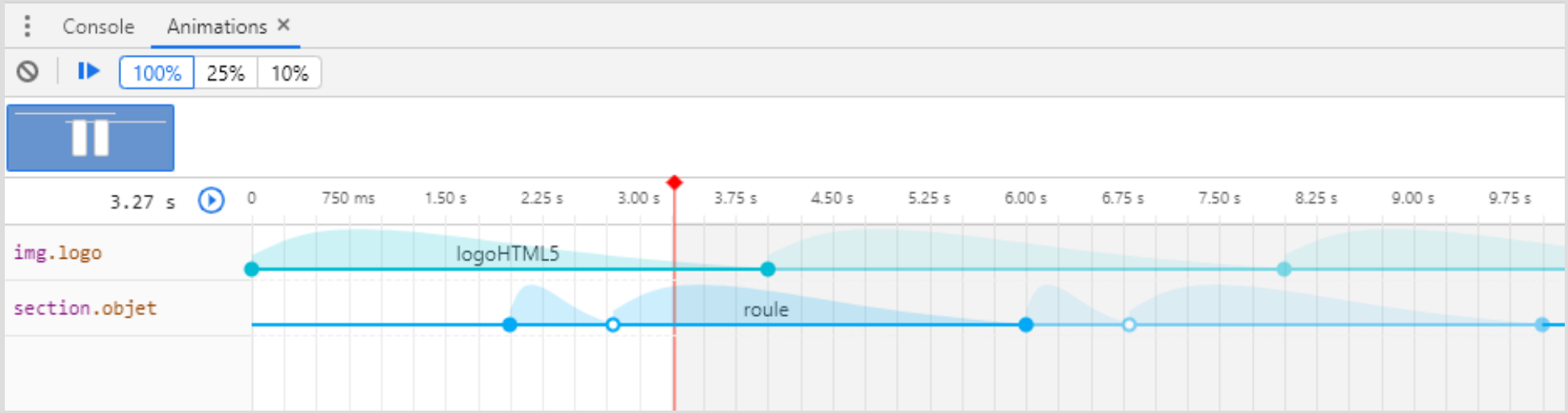
```
<main>
    <h1>Les animations avec CSS</h1>
    <section class="objet">
        <h3>CSS3</h3>
        <p>Animation</p>
    </section>
    <br />
    <button onclick="anime();">Animer le titre</button>
</main>

<script>
    var leTitre = document.querySelector("main h1");

    leTitre.addEventListener("animationstart", demarre, false);
    leTitre.addEventListener("animationend", fini, false);
    leTitre.addEventListener("animationiteration", compte, false);

    function anime() {
        console.log("Animation CSS");
        leTitre.style.animationName = "deplaceTitre";
    }

    function demarre(e) {
        var monAnimation = e.animationName;
        console.log("Animation "+ monAnimation +" démarre");
    }

    function fini() {
        console.log("Animation terminée");
        leTitre.style.animationName = "";
    }

    function compte() {
        console.log("L'animation recommence");
    }
</script>
```

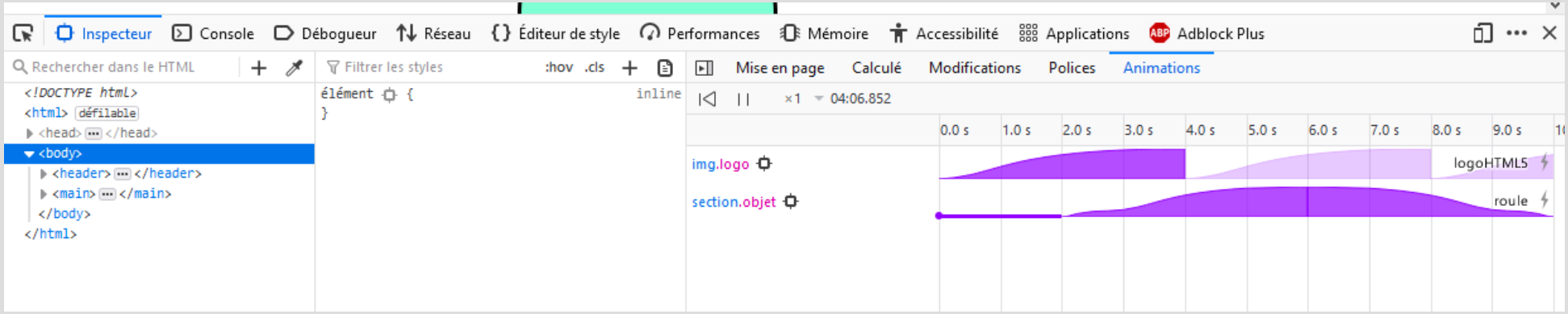
Outils dans les navigateurs



- Chrome, Firefox,… ont des **outils de test** d'animation (Webdev Tools – F12)
- Dans **Chrome**, il faut activer le panneau Animations dans More Tools
- Dans **Firefox**, il est installé dans l'Inspecteur dans un onglet Animations

Outils Chrome - Firefox





Applications

- **Slideshow** : avec une animation, on fait défiler des images automatiquement
- **Sprite** : avec une animation image par image, on peut simuler le mouvement d'un objet ou d'un personnage à la manière d'un dessin animé

NB : pour réaliser le passage d'une image à l'autre, il faut progresser **par paliers** au lieu d'avancer de manière continue, on emploie alors la fonction **steps(nombre)** comme animation-timing-function

Slideshow – HTML

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Animations avec CSS - Slideshow</title>
        <link type="text/css" rel="stylesheet" href="slideshow.css" />
    </head>
    <body>
        <header>
            <img class="logo" alt="Logo" src="html5.png" />
        </header>
        <main>
            <h1>Les animations avec CSS : Slideshow</h1>
            <section id="slideshow">
                <ul id="contenu">
                    <li><img src="images/image1.jpg" alt="Panda" /></li>
                    <li><img src="images/image2.jpg" alt="Poussin" /></li>
                    <li><img src="images/image3.jpg" alt="Chaton" /></li>
                    <li><img src="images/image4.jpg" alt="Ecureuil" /></li>
                </ul>
            </section>
        </main>
    </body>
</html>
```

Slideshow – CSS

```css
#slideshow {
    position: relative;
    width: 360px;
    height: 360px;
    margin: 20px auto;
    overflow: hidden;
}

@keyframes slider {
      0.0% { left: 0px; }
     12.5% { left: 0px; }
     25.0% { left: -360px; }
     37.5% { left: -360px; }
     50.0% { left: -720px; }
     62.5% { left: -720px; }
     75.0% { left: -1080px; }
     87.5% { left: -1080px; }
    100.0% { left: 0px; }
}
```

```css
#contenu li {
    display: inline;
}

#contenu {
    position: absolute;
    top: 0px;
    left: 0px;
    width: 1440px;
    margin: 0px;
    padding: 0px;

    animation-name: slider;
    animation-duration: 12s;
    animation-iteration-count: infinite;
    animation-timing-function: linear;
}
```

Sprite – HTML

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Animations avec CSS - Sprites</title>
        <link type="text/css" rel="stylesheet" href="sprite.css" />
    </head>
    <body>
        <main>
            <h1>Les animations avec CSS : Sprites</h1>
            <section id="sprite">
                <img src="images/bugJongle.png" alt="Jongleur" />
            </section>
            <section id="oiseau">
                <img src="images/oiseau.png" alt="Oiseau" />
            </section>
            <section id="course">
                <img src="images/homme.png" alt="Coureur" />
            </section>
        </main>
    </body>
</html>
```

Sprite – CSS

```css
#sprite {
    width: 200px;
    overflow: hidden;
}

#sprite img {
    position: relative;
    animation: jongleur 2s infinite steps(11);
}

@keyframes jongleur{
    from { left:0; }
    to { left:-2200px; }
}

#oiseau {
    width: 183px;
    overflow: hidden;
}

#oiseau img {
    position: relative;
    animation: vol 2s infinite steps(14);
}

@keyframes vol {
    from { left:0; }
    to { left:-2562px; }
}
```

```css
#course {
    width: 128px;
    overflow: hidden;
    animation: avance 4s infinite linear;
}

#course img {
    position: relative;
    animation: coureur 1s infinite steps(8);
}

@keyframes coureur {
    from { left:0; }
    to { left:-1024px; }
}

@keyframes avance {
    from {transform: translateX(-200px);}
    to {transform: translateX(1200px);}
}
```