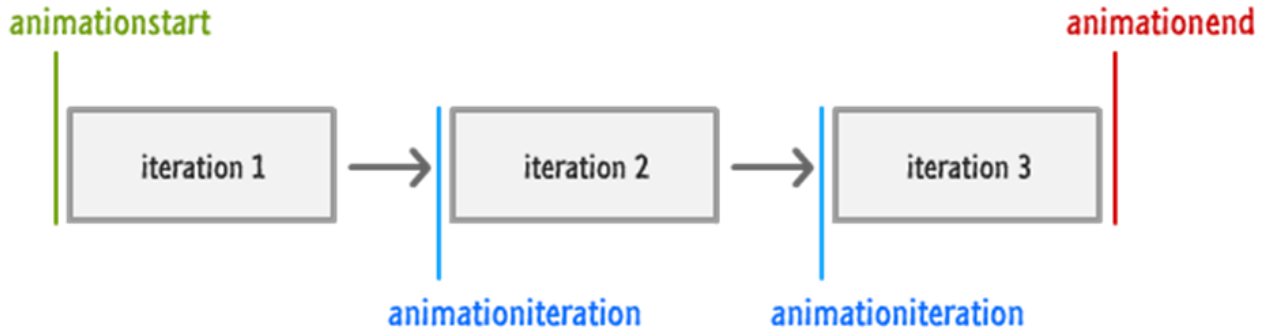
Semaine 12 - AnimationEvent et requestAnimationFrame

Les événements des animations CSS

Il est possible d'avoir plus de contrôle sur les animations CSS, ainsi que des informations utiles à leur propos, en utilisant les événements liés aux animations. Ces événements, représentés par l'objet **AnimationEvent**, peuvent être utilisés pour détecter **quand une animation démarre**, **quand elle se termine** ou quand elle **commence un nouveau cycle** (« *animation-iteration-count* »). Chaque événement inclut minimalement comme information: l'instant auquel il survient ainsi que le nom de l'animation qui a déclenché l'événement.

Il existe donc **trois** événements de type **AnimationEvent**, qui sont:

- **animationstart** : cet événement est diffusé dès que l'animation démarre. Si un délai a été spécifié avec la propriété «*animation-delay*», alors cet événement se déclenche seulement lorsque le délai est expiré;
- **animationiteration** : cet événement est diffusé **à la fin de chaque itération** d'une animation, sauf si un événement « animationend » est diffusé en même temps, soit lors de la dernière itération. Cela signifie que l'événement « *animationiteration* » ne se produit pas pour les animations ne jouant qu'une seule fois (*animation-iteration-count : 1*) qui est la valeur par défaut;
- **animationend** : Cet événement est distribué lorsque l'animation se termine. **MAIS**, il va de soi, qu'il n'est jamais diffusé si la propriété «*animation-iteration-count*» est réglée à **infinite**.



Par ailleurs, tous les événements de type **AnimationEvent** possèdent deux propriétés spécifiques qui peuvent nous renseigner sur l'événement produit :

- **animationName**: renvoie une chaîne contenant la valeur de la propriété CSS animation-name (la règle @keyframes) associée à l'animation ;
- **elapsedTime** : renvoie un nombre flottant donnant la quantité de temps, en secondes, depuis que l'animation est en cours d'exécution. Pour un événement "*animationstart*", la propriété *elapsedTime* est évidemment égale à 0,0.

Dans le code qui suit, on récupère un élément « div » de la page HTML et on lui attribue une animation par programmation qui joue en alternance 5 fois. À l'aide de la gestion des événements d'animation, on y affiche le temps écoulé après chaque itération de l'animation et aussi la fin de celle-ci.

```
<body>
    <div> </div>
    <!-- Scripts Javascript -->
    <script>
        //Récupérer la division
        let laDiv = document.querySelector("div");
```

```
        //Attribuer l'animation deplacer-div
        laDiv.style.animation = "deplacer-div 1s 5 alternate";

        //Attribution des gestionnaire d'événement de l'animation
        laDiv.addEventListener("animationiteration", afficherInfos);
        laDiv.addEventListener("animationend", afficherInfos);

        //Fonction appelée par les événements distribués
        function afficherInfos(event) {
            if(event.type == "animationiteration"){
                laDiv.innerText = event.elapsedTime + " sec(s) écoulé(es)";
            }else{
                laDiv.innerText = "Fin de l'animation";
                //Enlever les gestionnaires d'événement
                laDiv.removeEventListener("animationiteration", afficherInfos);
                laDiv.removeEventListener("animationend", afficherInfos);
            }
        }
    </script>
</body>
```

Attention! Notons que les conteneurs parents écoutent par défaut la fin des animations attribuées à leurs éléments enfants, ce qui peut engendrer des conflits si on souhaite réagir uniquement à la fin d'une animation attribuée au conteneur parent et que ses nœuds enfants sont également animés. Dans ce cas, l'analyse de la cible (event.target) ou de l'animation concernée (event.animationName) dans la fonction de rappel exécutée sera très importante.

Dans l'exemple qui suit, nous avons une section incluant une balise animée. La section est également animée avec un délai d'exécution. Un gestionnaire d'événement "*animationend*" est attribué à la section, nous devons donc nous assurer que l'événement distribué concerne bien la fin de l'animation de la section. Les différentes portions de codes sont les suivante:

```
<!-- Structure HTML-->
<section class="animer-couleur">
    <div class="animer-mot">ANIMATION-INTRO</div>
</section>
```

```
/*CODE CSS*/
.animer-mot::before {
    /*Animation par étape*/
    animation: animer-mot 3s steps(15, end) forwards;
}

/*L'animation est appliquée avec un délai de 3.5s*/
.animer-couleur {
    animation: animer-couleur 750ms ease-out 3.5s;
}
```

```
/*** Scripts Javascript ***/

let laSection = document.querySelector("section");
laSection.addEventListener("animationend", demarrerApplication);


//Fonction appelée à la fin de l'animation de la section
function demarrerApplication(event) {
    if(event.animationName =="animer-couleur"){
        //Code quand l'animation de la section est terminée
    }
}
```

Quelques références utiles

- Mozilla (Septembre 2022). HTMLElement: animationstart event. Repéré à : https://developer.mozilla.org/fr/docs/Web/API/HTMLElement/animationstart_event
- Mozilla (Septembre 2022). HTMLElement: animationiteration event. Repéré à : https://developer.mozilla.org/fr/docs/Web/API/HTMLElement/animationiteration_event
- Mozilla (Septembre 2022). HTMLElement: animationend event. Repéré à : https://developer.mozilla.org/fr/docs/Web/API/HTMLElement/animationend_event
- Mozilla (Septembre 2022). Utiliser les animations CSS | Utiliser les événements liés aux animations. Repéré à : https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Animations/Using_CSS_animations#utiliser_les_%C3%A9v%C3%A9nements_li%C3%A9s_aux_animations

Animer les éléments du DOM avec JavaScript

Vous avez déjà vu, dans le cours **TIM-1J1 Animation et interactivité en jeu**, que JavaScript dispose de **deux fonctions temporelles classiques** qui permettent d'influencer le **délai d'exécution** de nos scripts pour permettre, par exemple, la création d'une animation. Il s'agit des méthodes **setTimeout()** et **setInterval()**.

La méthode **setTimeout()** permet de déclencher un code au bout d'un temps donné, tandis que la méthode **setInterval()** va déclencher un code selon un **intervalle régulier** qui aura été spécifié.

Cette note de cours couvre une **nouvelle spécification**, issue de HTML5, l'API **RequestAnimationFrame** qui permet d'optimiser l'affichage de plusieurs éléments animés en **synchronisant ces derniers avec le moteur graphique du navigateur**.

Utiliser l'API requestAnimationFrame

L'usage de **requestAnimationFrame()** est proche de **setInterval()**, mais a **plusieurs avantages** :

- Le navigateur **optimise les traitements** pour garantir un rafraîchissement régulier, généralement à 60 images par seconde (60 ips), ce qui rend les animations plus **fluides**;
    - **requestAnimationFrame()** est la **version moderne** de setInterval() . Il exécute un bloc de code spécifié avant que le navigateur n'effectue un nouveau rafraîchissement tout en prenant en compte le temps de calcul nécessaire, ce qui permet à une animation d'être exécutée à une fréquence d'images appropriée quel que soit l'environnement dans lequel elle est exécutée. **L'animation sera donc synchronisée avec le rafraichissement de l'écran** et sera plus fluide . La fréquence moyenne de rafraîchissement est de 60 fois par seconde (60 ips), soit à tous les 16,666666 millisecondes (1000/60);
    - **setInterval() ne se synchronise pas correctement avec l'actualisation de l'affichage**, ce qui peut souvent provoquer un scintillement important. Notons aussi que setInterval() fonctionne uniquement avec des nombres entiers, alors 1000/60 sera égal à 17 millisecondes au lieu de 16,666666 millisecondes, ce qui explique également certains sauts d'images.
- Le navigateur **interrompt l'animation** lorsque l'onglet n'est plus visible (perd le focus) afin de réduire la consommation CPU (plus optimal pour la durée de vie de la batterie d'un périphérique mobile).

Animer avec l'API requestAnimationFrame

L'utilisation de la méthode **requestAnimationFrame()**, présente des similitudes avec setInterval() et setTimeout(), sauf que le délai d'exécution n'est pas précisé, et que la fonction de rappel (« ***callback*** ») **doit être appelée de nouveau à chaque exécution**, pour obtenir un effet d'animation.

- **window.requestAnimationFrame(laFonctionAExecuter);**
- **ou, requestAnimationFrame(laFonctionAExecuter)**

À noter que l'usage du préfixe **window** n'est pas obligatoire, on le mentionne ici pour préciser que la méthode requestAnimationFrame appartient à l'objet global « ***Window*** » .

L'exemple ci-dessous fait simplement tourner un élément HTML à chaque itération:

```
let elementHTML = document.querySelector("div");
let degreRotation = 0;

//Première requête pour l'animation
requestAnimationFrame(tournerElement);
```

```
function tournerElement() {
    //On incrémente le degré de rotation
    degreRotation++;

    //On applique la transformation
    elementHTML.style.transform = `rotate(${degreRotation}deg)`;

    //Prochaine requête pour l'animation
    requestAnimationFrame(tournerElement);
}
```

Annuler un effet d'animation

Pour annuler un effet d'animation avec requestAnimationFrame(), on doit utiliser la méthode **cancelAnimationFrame()**, qui appartiennent aussi à l'objet global « Window » :

- **window.cancelAnimationFrame(requeteID);**
- ou, **cancelAnimationFrame(requeteID)**

Tout comme clearInterval() ou clearTimeout(), cette méthode prend en argument l'**identifiant numérique** de la requête à annuler. Cet identifiant (qui est un simple nombre entier) est **retourné** à chaque appel de la méthode requestAnimationFrame(). **Pour interrompre une animation, il faut donc s'assurer d'enregistrer cet identifiant numérique dans une variable**.

Dans l'exemple qui suit, on interrompt l'animation de l'élément HTML qui tourne, lorsqu'on clique sur ce dernier :

```
let elementHTML = document.querySelector("div");
let degreRotation = 0;
let requeteID = 0;
//Première requête pour l'animation
requestAnimationFrame(tournerElement);

function tournerElement() {
    //...même code que ci-haut...SAUF avec l'ajout ci-bas:

    //Prochaine requête pour l'animation
    requeteID = requestAnimationFrame(tournerElement);
}

//Arrêt de l'animation si on clique sur l'élément
elementHTML.addEventListener("click", arreterAnimation);

function arreterAnimation(event){
        cancelAnimationFrame(requeteID);
}
```

Toutefois, pour basculer entre arrêter ou repartir l'animation au clic sur l'élément, le code serait toutefois le suivant :

```
//Première requête d'animation
requeteID = requestAnimationFrame(tournerElement);

//Arrêter ou démarrer l'animation si on clique sur l'élément
elementHTML.addEventListener("click", arreterOuPartirAnimation);

function arreterOuDemarrerAnimation(event) {
    if (requeteID == 0) {
        //Partir une nouvelle requête pour l'animation
        requeteID = requestAnimationFrame(tournerElement);

    } else {
        //Arrêter l'animation et gérer la valeur de requeteID
        cancelAnimationFrame(requeteAnimID);
        requeteID = 0;

    }
}
```

Temps dans une fonction de rappel

La fonction de rappel appelée par requestAnimationFrame(), peut être utilisée pour animer quasiment n'importe quelle propriété CSS. Il faut noter que **cette fonction de rappel reçoit une information en paramètre**, qui représente le **temps** en millisecondes à laquelle la fonction de rappel est appelée.

requestAnimationFrame(laFonctionAExecuter);

function laFonctionAExecuter(tempsActuel) { //code

}

Ce **temps** est **calculé à partir du début de la navigation de la page**. Il est exprimé en nombre flottant et est mesuré en **millisecondes**. En matière d'animation, la prise en compte du temps d'exécution peut nous permettre d'animer des éléments selon des cadences différentes, ou d'arrêter une animation après un certain temps écoulé.

Dans l'exemple qui suit, on fait tourner l'élément HTML uniquement pendant environ 5 secondes (5000 millisecondes). Dans le code qui suit, examiner la variable **tempsInitial** :

```
let elementHTML = document.querySelector("div");
let degreRotation = 0;
//Pour le décompte du temps écoulé - null au début,
//sera enregistré à la première requête d'animation
let tempInitial = null;

function tournerElement(tempsActuel) {
    //On incrémente le degré de rotation
```

```
    degreRotation++

    //On applique la transformation
    elementHTML.style.transform = `rotate(${degreRotation}deg)`;

    //On enregistre D'ABORD le temps initial, si c'est la première requête
    if (tempInitial == null) {
        tempInitial = tempsActuel;
    }
    let secondesEcoulees = Math.round((tempsActuel - tempInitial) / 1000);

    if (secondesEcoulees <= 5) {
        //Prochaine requête pour l'animation
        requeteAnimID = requestAnimationFrame(ajouterGoutteEau);
    }

}
```

Notons toutefois qu'il serait aussi justifié d'utiliser simplement une fonction **setTimeout()** pour arrêter l'exécution des requêtes d'animation sans avoir besoin de calculer à chaque fois le temps écoulé… Dans ce cas toutefois, l'enregistrement du numéro de requête d'animation en cours (**requeteID**) serait **obligatoire**.

Quelques références utiles

- Mozilla (Mars 2023). Window.requestAnimationFrame() - Reference Web API | MDN. Repéré à: https://developer.mozilla.org/fr/docs/Web/API/Window/requestAnimationFrame
- Mozilla (Septembre 2022). Window.cancelAnimationFrame() - Reference Web API | MDN. Repéré à: https://developer.mozilla.org/fr/docs/Web/API/Window/cancelAnimationFrame
- ToutJavaScript (Mars 2020). Méthode : window.requestAnimationFrame(). Repéré à : https://www.toutjavascript.com/reference/ref-window.requestanimationframe.php