Partie 1

Animation d'un cheval au galop avec `requestAnimationFrame`

TD - *Completer les commentaires 1. à 9. du scipt ci-dessous :*

```
<body>
  <h1>Animation - Cheval au galop</h1>
  <img id="cheval" src="cheval_0.png" alt="Cheval au galop">
  <div id="info">
    Frame affichée : <span id="compteur">0</span> / 4
  </div>
  <script>
    //   1. ..
    const frames = [
      "images/cheval_0.png",   // phase appui
      "images/cheval_1.png",   // envol avant
      "images/cheval_2.png",   // suspension
      "images/cheval_3.png",   // réception
      "images/cheval_4.png",
    ];
    //   2. ..
    const img      = document.getElementById("cheval");
    const compteur = document.getElementById("compteur");

    let index    = 0;   // indice courant dans le tableau
    let tick     = 0;   // compteur de callbacks

    const VITESSE = 8;  // changer de frame toutes les 8 fps
                        // (~60fps/8  7-8 images par seconde)
    //   3. ...
    function animer() {
      tick++;
      //   4. ...
      if (tick % VITESSE === 0) {
        img.src = frames[index];           // mise à jour de l'
   image
        //   5. ...
        compteur.textContent = index;
        //   6. ...
        index = (index + 1) % frames.length;
      }

      //   8. ...
      requestAnimationFrame(animer);
    }
    //   9. ...
    requestAnimationFrame(animer);
  </script>
```

Partie 2

COURS

Niveau : 1ère NSI
Prérequis : HTML/CSS (mise en page, centrage), JavaScript (tableaux, fonctions, modulo)
Durée estimée : 1h —

2.1 1. Objectif

Réaliser une animation fluide d'un cheval au galop en combinant :

- **HTML/CSS** : afficher et centrer une image à l'écran
- **JavaScript** : gérer une liste d'images et boucler dessus avec le modulo
- **`requestAnimationFrame`** : synchroniser l'animation avec le rafraîchissement de l'écran

---

2.2 2. Principe de l'animation par défilement d'images (sprite)

Un cheval au galop peut être animé en affichant successivement plusieurs images (*frames*), chacune représentant une phase du mouvement.

```
frame 0 → frame 1 → frame 2 → frame 3 → frame 0 → ...
```

On stocke les chemins vers ces images dans un **tableau JavaScript**, et on affiche l'une après l'autre en boucle.

Partie 3

Les méthodes de callback

3.1 Autoréférence

Dans l'exemple suivant, la fonction `bip` fait appel à elle-même, dans le corps de sa propre fonction. C'est de l'autoréférence.

Le programme ci-dessous devrait établir un décompte de 2000 à 0, mais en réalité, celui-ci va entrainer un blocage du système : (**ne pas tester**)

```
<html>
    <div id="bip" class="display"></div>
<script >
```

```
4       let counter = 2000;
5      function bip () {
6          document.getElementById("bip").innerHTML = counter + "
   secondes restantes";
7          counter = counter - 1;
8          bip();
9      }
10     bip();
11 </script>
```

La pile d'appel est alors :

pile d'appel
main()
bip()
bip()
...

En javascript, pour réaliser des appels de fonction à intervalle de temps régulier, il faut utiliser un mécanisme de *callback*, unique solution pour ne pas bloquer une exécution lorsque celle-ci est trop *couteuse*.

Pour eviter le problème de blocage du navigateur, on utilise des méthodes avec fonction de rappel (aussi appelée *callback* en anglais). La fonction de callback est une fonction passée dans une autre fonction en tant qu'argument, qui est ensuite invoquée en temps régulier. On utilise pour cela les méthodes `setInterval`, `setTimeout`et `requestAnimationFrame()`

3.2 Les méthodes `setTimeout` et `setInterval`

Les méthodes `setTimeout` et `setInterval` : Ce sont des processus indépendants qui, quand ils sont lancés par une instruction, ne bloquent pas l'affichage du reste de la page ni les actions de l'utilisateur. Elles permettent de placer dans la pile des prochaines fonctions à exécuter une fonction particulière. Javascript exécute cette pile fonction après fonction dans l'ordre de la pile.

- `SetTimeout` indique un délai avant exécution
- `setInterval` déclenche une opération à intervalles réguliers
- `ClearTimeout` interrompt le décompte de setTimeout et de setInterval

Syntax : `let intervalID = setInterval(function ou bloc de code, [delay (en ms)])` ;

3.3 Méthode `requestAnimationFrame()`

C'est aussi une méthode de *callback* pour realiser une animation en javascript.

La méthode requestAnimationFrame() de l'interface Window indique au navigateur qu'on souhaite exécuter une animation. Elle demande au navigateur d'appeler une fonction de rappel (callback en anglais) fournie par l'utilisateur avant le prochain rafraîchissement.

Dans cet exemple, issu du site MDN, un élément est animé pour 2 secondes (2000 millisecondes). L'élément se déplace à une vitesse de 0.1px/ms vers la droite. Sa position relative (en pixels CSS) peut donc être calculée en fonction du temps écoulé entre le début de l'animation (en millisecondes) et 0.1 * ecoule. La position finale de l'élément est située 200px (0.1 * 2000) à droite de sa position initiale.

```
<html>
    <div id="bip" class="display"></div>
<script >
      let counter = 2000;
     function bip () {
         document.getElementById("bip").innerHTML = counter + "
   secondes restantes";
         counter = counter - 1;
         requestAnimationFrame(bip);
             }
    requestAnimationFrame(bip);
</script>
```

Partie 4

Exemple complet : cheval au galop

4.1 Partie HTML

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Cheval au galop</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <img id="cheval" src="cheval_0.png" alt="Cheval au galop">
  <script src="animation.js"></script>
</body>
</html>
```

Points clés : - Un seul élément `<img>` avec l'identifiant `cheval`. - On ne crée **pas** plusieurs balises `<img>` : c'est JavaScript qui change la source.

4.2 CSS — Centrer l'image à l'écran

```
body {
  margin: 0;
  height: 100vh;
```

```
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #87CEEB; /* ciel bleu */
}

#cheval {
  width: 300px;
}
```

Rappel Flexbox :

Propriété	Rôle
`display: flex`	Active le mode Flexbox sur le `<body>`
`justify-content: center`	Centre **horizontalement**
`align-items: center`	Centre **verticalement**
`height: 100vh`	Le body occupe toute la hauteur de la fenêtre

4.3 JavaScript — L'animation

4.3.1 Le tableau des images

```
const frames = [
  "cheval_0.png",
  "cheval_1.png",
  "cheval_2.png",
  "cheval_3.png",
  "cheval_4.png"
];
```

On regroupe tous les chemins dans un tableau. Modifier l'animation revient simplement à modifier ce tableau.

4.3.2 Le modulo pour boucler

Pour parcourir le tableau en boucle infinie, on utilise l'opérateur **modulo** (%) :

```
let index = 0;

// À chaque étape :
index = (index + 1) % frames.length;
```

index	(index + 1) % 5
0	1
1	2
4	**0** ⟵ repart au début

Propriété clé : `n % longueur` reste toujours dans `[0, longueur - 1]`.

4.3.3 `requestAnimationFrame`

`requestAnimationFrame(callback)` demande au navigateur d'appeler `callback` **avant le prochain affichage à l'écran** (environ 60 fois par seconde).

C'est préférable à `setInterval` car : - synchronisé avec l'écran (pas de saccades) - mis en pause automatiquement si l'onglet est inactif (économie de ressources)

4.3.4 Contrôler la vitesse avec un compteur

60 appels par seconde, c'est trop rapide pour une animation de cheval. On n'avance dans le tableau qu'**une fois tous les N appels** :

```
const img    = document.getElementById("cheval");
const frames = ["cheval_0.png","cheval_1.png","cheval_2.png",
                "cheval_3.png","cheval_4.png","cheval_5.png"];

let index    = 0;
let compteur = 0;
const VITESSE = 8; // changer une frame toutes les 8 images écran

function animer() {
  compteur++;

  if (compteur % VITESSE === 0) {
    img.src = frames[index];
    index = (index + 1) % frames.length;
  }

  requestAnimationFrame(animer); // relance la boucle
}

requestAnimationFrame(animer); // démarre l'animation
```

4.4 Schéma de fonctionnement

```
requestAnimationFrame(animer)


   animer() appelée

   compteur++

```

```
compteur % VITESSE === 0 ?

OUI    img.src = frames[index]
         index = (index+1) % frames.length

          requestAnimationFrame(animer)

                (boucle infinie)
```

4.5 Code complet (en un seul fichier)

Voir le fichier **`cheval_animation.html`** fourni avec ce cours.

Partie 5

Résumé des notions mobilisées

Notion	Utilisation
Tableau JS	Stocker les références des images
Modulo `%`	Boucler sur le tableau indéfiniment
`document.getElementById`	Accéder à l'élément `<img>`
`requestAnimationFrame`	Synchroniser l'animation avec l'écran
Flexbox CSS	Centrer l'image dans la page
`img.src`	Changer dynamiquement l'image affichée