Javascript

Romain Jalabert - 15.02.2022 - IPSSI

Les événements JS avec target

JS peut réagir à des **événements** qui se produisent en HTML.

Un event HTML peut être lié à une action du navigateur (page qui finit de charger) ou à celle d'un utilisateur (clique sur un bouton, input qui change …)

On peut ainsi exécuter du code lorsque ces événements sont détectés. *Par exemple*, lorsque l'on utilise un eventListener sur un élément HTML il est possible de récupérer des informations sur l'événement en question (si l'on clique, la cible du clic ainsi que ses informations avec target).

```
const play = document.querySelector('.btn-play');

play.addEventListener('click', (e) => {console.log(e.target)})

// Retourne <div class="btn-play">
```

SetTimeout() et SetInterval()

La méthode **SetTimeout()** permet d'appeler une fonction après un certain nombre de millisecondes.

ex : const myTimeout = setTimeout(myFunction, 5000);

Ici la fonction myFunction() sera exécutée au bout de 5 secondes. Avec clearTimeout(myTimeout) je stoppe le Timeout ainsi que l'éxecution de la fonction associée.

La méthode **SetInterval()** permet d'appeler une fonction tous les x millisecondes.

ex : const myInterval = setInterval(myFunction, 5000);

Ici la fonction myFunction() sera exécutée toutes les 5 secondes. On utilisera clearInterval() pour stopper la fonction. Avec clearInterval(myInterval), la fonction cessera d'être répétée.

Math.random() et Math.floor()

Math.random() va nous permettre de retourner une valeur aléatoire comprise entre 0 (inclus) et 1 (exclus).

Combinée avec **Math.floor()** on pourra récupérer des entiers aléatoires.

ex : **Math.floor(Math.random() * 6)** > Retourne un entier entre 0 et 5

Exercice : Créer une petite app qui permet de **sélectionner un élève au hasard** dans la classe.

On utilisera les fonctions ci-dessus ainsi que setInterval. Les noms sont contenus dans un tableau.

Créer un jeu en JS pierre/feuille/ciseau (shifumi !)

- 3 choix possibles (pierre, feuille, ciseaux)
- Montrer à la fois le choix du user et de l'ordinateur
- Choisir dans un premier temps le symbole puis appuyer sur Jouer pour lancer la partie
- Si aucun item sélectionné on affiche un message d'erreur
- *Bonus : Rajouter le score du user et du computer (+1 par victoire, 0 dans les autres cas)*

Exercice - Quiz en JS

Réaliser un **QCM** en Javascript. Vous avez la data à disposition, à vous d'afficher les questions comme sur l'exemple.

Une réponse possible par question.

Mario / Dino Game

On va coder un petit jeu JS qui ressemble furieusement au **dino** de Google.

L'idée est d'avoir un personnage (qui pourrait juste être un carré) qui saute pour éviter des obstacles. Il se situe à gauche de notre fenêtre et les obstacles viennent de la droite.

On va comme toujours procéder par étape :

- D'abord créer les personnages
- Puis s'intéresser à leur placement dans le jeu et leur animation (le dino saute et les obstacles se déplacent de droite à gauche)
- Enfin créer les variables js ainsi que les fonctions nécessaires.

  Dans un premier temps "jump".

Mario / Dino Game

Bonus : La présence d'un compteur pour le score !

Bonus + : Un décor avec quelques animations, de nouvelles fonctionnalités ? Prends toi pour Hideo Kojima !

Notions utile :

- **@keyframes animation** en CSS pour les mouvements (https://www.w3schools.com/cssref/css3_pr_animation-keyframes.php)

- les fonctions **SetTimeout** et **SetInterval** qui vont nous être très utiles ! SetInterval notamment pour vérifier si notre personnage est toujours en vie à intervalles très réguliers.

- **getComputedStyle** méthode qui récupère la liste des règles de style CSS d'un élément sous forme d'objet. (Utile pour cerner nos personnages)