# **JS** : Gestion du temps

En JavaScript on va pouvoir temporiser certaines actions ce qui est très pratique dans certain cas. Pour un ordinateur le temps est mesuré par défaut en **milliseconde**.

## LA DATE

Pour manipuler la date on va utiliser la fonction `Date();`

```
var laDate = new Date();
```

Cette fonction nous renverra le jour, le mois, l'année, l'heure et le fuseaux horaire de votre ordinateur au moment ou cette variable est déclarée.

Pour pouvoir récupérer une des informations sur ce format, nous avons des fonctions très simples qui vont pouvoir nous aider :

- **`getFullYear()`** : renvoie l'année sur 4 chiffres ;
- **`getMonth()`** : renvoie le mois (0 à 11) ;
- **`getDate()`** : renvoie le jour du mois (1 à 31) ;
- **`getDay()`** : renvoie le jour de la semaine (0 à 6, la semaine commence le dimanche) ;
- **`getHours()`** : renvoie l'heure (0 à 23) ;
- **`getMinutes()`** : renvoie les minutes (0 à 59) ;
- **`getSeconds()`** : renvoie les secondes (0 à 59) ;
- **`getMilliseconds()`** : renvoie les millisecondes (0 à 999).

```
laDate.getDate(); // Renvoi uniquement le jour de la date
```

Il est possible d'initier cette fonction lors de la déclaration de la variable :

```
var secondes = new Date().getSeconds(); // Renvoi uniquement les secondes
```

**À noter** : Ces fonctions sont toutes selon l'heure locale. Il existes aussi d'autres fonctions et méthodes pour la gestion de la date.

## LES FONCTIONS TEMPORELLES

La fonction **`setTimeout()`** permet de déclencher des instructions au bout d'un temps donné (en millisecondes). On écrit alors :

```
setTimeout(function() {
    // Les instructions à exécuter
}, 2000);
```

Au bout de 2000 millisecondes (soit 2 secondes) on va donc déclencher nos instructions.
On peut également appeler une autre fonction à l'aide de cette méthode :

```
setTimeout(maFonction, 2000);
// la fonction nommée maFonction sera déclenchée au bout de 2 secondes
```

La méthode `setInterval` fonctionne comme `setTimeout` mais elle permet de déclencher des instructions à intervalle régulier d'un temps donné (en millisecondes).
Par exemple :

```
setInterval(function() {
    // Les instructions à exécuter
}, 2000);
```

Il est également possible de déclarer une variable pour nommer notre fonction temporelle :

```
var decompte;

decompte = setTimeout(maFonction, 2000);
```

Cela va nous permettre par la suite d'annuler une action temporelle avec les méthodes associées `clearTimeout` et `clearInterval`. Par exemple :

```
clearTimeout(decompte) ; //On annule l'action temporelle nommée decompte
```

**À noter** : la fonction `setTimeout` est bien plus « stable » que `setInterval`. Mieux vaut se passer de `setInterval` et utiliser `setTimeout` quel que soit le cas d'application, vous obtiendrez des animations beaucoup plus fluides.

Pour cela on va utiliser une fonction qui va lancer un délais avec `setTimeout` en boucle. On aura alors le même effet qu'avec la fonction `setInterval`. Il faudra donc écrire :

```
function interval() {
     setTimeout(interval, 2000);
     // La fonction interval() fait appel à elle-même toute les 2 secondes
}

interval(); // On initie la fonction
```