Javascript

Les dates et le temps

Les dates en javascript

- Javascript dispose du constructeur Date pour travailler avec les dates
- Comparer deux dates permet de connaître le temps écoulés entre deux instants dans un programme
- De manière générale en programmation, le nombre de millisecondes écoulées depuis le 1er janvier 1970 sert de base de calcul

Les dates en javascript

- Pour afficher la date au moment où le script est exécuté, on utilise new Date()
- Pour afficher le nombre de millisecondes écoulées depuis le 1er janvier 1970 au moment de l'exécution du script, on utilise Date.now()
- Pour calculer le temps écoulés entre deux dates, il suffit de soustraire la date la plus récente à la plus ancienne

Comparer deux dates

```
var premiereDate = Date.now();
// Du code est exécuté entre les deux dates
var secondeDate = Date.now();
var tempsEcoule = secondeDate - premiereDate;
// tempsEcoule contient le nombre de millisecondes de différences entre nos deux dates
```

Convertir une date

- Pour comparer facilement les dates, il est possible de convertir une date en millisecondes en utilisant Date.parse(“date”)
- Exemple :

```
var timestamp = Date.parse("26 Oct 2018");
document.write(timestamp); // Affichera 1540504800000
```

Convertir une date

- Inversement, il est possible à l'aide des méthodes natives de l'objet Date d'obtenir le mois,le jour, l'année etc… d'un temps en millisecondes
- Exemple :

```
var timestamp = Date.parse("26 Oct 2018");
var maDate = new Date(timestamp);
document.write(maDate.getMonth()); // Affichera le chiffre 9 , Octobre = 9eme mois de l'année en javascript, 0 = janvier
```

Convertir une date

- Voici certaines méthodes de traitement de date :
- 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)

Fonctions temporelles

- Javascript est un langage de programmation qui offre nativement une gestion du temps
- Il est ainsi possible d'exécuter du code au bout de x millisecondes/secondes ou à intervalle régulier

setTimeout

- La fonction setTimeout permet d'exécuter du code au bout d'un certain temps
- Cette fonction accepte en paramètre une fonction ou une fonction anonyme (un morceau de code écrit directement)
- Le paramètre spécifiant le temps avant execution s'écrit toujours en millisecondes, par ex 5 secondes s'écrira 5000

setTimeout

- Exemple 1 : Appel d'une fonction au bout de 3 secondes

```
setTimeout(maFonction, 3000); // maFonction sera exécutée au bout de 3 secondes
```

- Exemple 2 : Avec une fonction anonyme

```
setTimeout(function() {
    alert("3 secondes se sont écoulées");
}, 3000);
```

setInterval

- La fonction setInterval permet d’exécuter du code à intervalle régulier
- Malheureusement, cette fonction pose de nombreux problèmes
- Il est donc recommandé d’utiliser setTimeout en obligeant la fonction exécutée par setTimeout à s’appeler elle même

setTimeout : auto appel

- Voici un exemple de boucle utilisant setTimeout

```
function boucle()
{
    document.write("2 secondes se sont écoulées <br>");
    setTimeout(boucle, 2000);
}
boucle();
```

clearTimeout : Annuler l’action temporelle

- Il peut être utile d’annuler une action temporelle, surtout dans le cas d’une boucle
- On utilisera la fonction clearTimeout

clearTimeout : Annuler l’action temporelle

```
var temps_passe=0;
function boucle()
{
temps_passe+=2;
document.write("2 secondes se sont écoulées <br>");
    if(temps_passe<20)
    {
        setTimeout(boucle, 2000);
    }
    else
    {
        clearTimeout();
    }
}
boucle();
```

clearTimeout : Annuler l’action temporelle

- Par défaut, clearTimeout() annule le dernier appel à la fonction setTimeout
- Si on gère plusieurs setTimeout en même temps, il faut pouvoir annuler un appel spécifique
- Pour se faire, on placera nos appels à setTimeout dans des variables

clearTimeout : Annuler l’action temporelle

- Exemple :

```
var timer1 = setTimeout(function() {
    alert("timer1");
},1500);
var timer2 = setTimeout(function() {
    alert("timer2");
},1000);
clearTimeout(timer1);
```