ANNEXE TP1

OBJET WINDOW

Il s'agit – comme son nom l'indique – de la fenêtre du browser en cours d'exécution. Tous les éléments de la fenêtre sont des propriétés ou des méthodes de window.

L'objet window étant intégré dans le langage, il n'est pas nécessaire de créer une instance de cette classe.

▪ Propriétés

Toutes ces propriétés correspondent à des éléments de la fenêtre ouverte. Elles permettent de changer quelques détails sympathiques dans le visuel d'une page web. Pour y accéder, on suit la syntaxe habituelle d'un objet :

window.propriété;

Il existe 6 propriétés qui correspondent chacune à des éléments ou des caractéristiques de la fenêtre. Voici leur liste :

- defaultStatus : le texte par défaut affiché dans la barre d'état.
- status : le texte à afficher dans la barre d'état, prioritaire par rapport à defaultStatus.
- name : le nom de la fenêtre
- screenTop : ordonnée du point supérieur gauche de la fenêtre.
- screenLeft : abscisse du point supérieur gauche de la fenêtre.
- closed : booléen indiquant si la fenêtre est fermée.

▪ Méthodes

▪ Généralités

L'objet window possède de nombreuses méthodes, dont certaines qui ont déjà été vues précédemment – les boîtes de dialogue -, et qui peuvent offrir quelques atouts à une page web. La plupart renvoient une valeur à la variable qui appelle la fonction. Voici leur syntaxe :

variable = window.méthode();

Certaines propriétés ne nécessitent pas de préciser le suffixe « window.» pour fonctionner. C'est notamment le cas des boites de dialogue.

- **Liste des méthodes**

Le nombre des méthodes étant élevé, elles sont présentées sous forme d'une liste que voici :

- **alert('texte')** : boite de message avec un bouton unique.
- **prompt('texte', 'valeur_défaut')** : boite d'invite avec un texte informatif et une zone de texte avec une valeur par défaut facultative.
- **confirm('texte') :** boite de confirmation avec un texte informatif et deux boutons OK et Annuler.
- **print('texte') :** afficher le texte dans la page.
- **open('URL', 'nom', options)** : ouvre une page pop-up avec les caractéristiques données en paramètres.
- **close() :** ferme la fenêtre.
- **focus() :** donne le focus à la page.
- **blur() :** enlève le focus à la page.
- **moveBy(x,y) :** déplacement relatif du coin supérieur gauche de la fenêtre.
- **moveTo(x,y) :** déplacement absolu du coin supérieur gauche de la fenêtre.
- **resizeBy(x,y) :** redimensionnement relatif de la fenêtre.
- **resizeTo(x,y) :** redimensionnement absolu de la fenêtre.
- **scrollBy(x,y) :** scroll relatif.
- **scrollTo(x,y) :** scroll absolu.
- **setTimeOut('fonction', temps)** : déclenche une minuterie en millisecondes.
- **clearTimeout('timer')** : suspend la minuterie.
- **stopTimeOut('timer') :** arrête une minuterie.
- **setInterval(code, délai)** : exécute le code – sous forme de String - passé en paramètre à chaque fois que le délai est écoulé.
- **clearInterval(timer)** : arrête la minuterie définie avec setInterval().
- **stop()** : arrête le chargement de la page.
- **home()** : ouvre la page de démarrage de l'internaute

▪ Ouverture et fermeture d'une fenêtre popup

La méthode javascript pour ouvrir une nouvelle fenêtre est **window.open** :
open ("URL", "nom de la fenêtre", "options")

cette méthode peut avoir plusieurs arguments, les voici:

- **L'url de la nouvelle fenêtre** - Simplement l'url qui sera chargée dans la nouvelle fenêtre.
- **Le nom de la nouvelle fenêtre** - Le nom de la nouvelle fenêtre, c'est très utile pour que deux fenêtres puissent communiquer entre elles.
- **Les options** - Différentes options permettant de configurer la nouvelle fenêtre, tel que sa taille, sa position...

il existe différentes options, certaines sont désactivée dans les navigateurs pour des raisons de sécurité.
voici les options généralement activées:
with: la largeur de la fenêtre en px

- **height**: la hauteur de la fenêtre en px
- **left**: la position de la fenêtre à partir de la droite en px
- **top**: la position de la fenêtre à partir du haut en px
- **menubar**: affichage du menu, yes pour oui, no pour non
- **toolbar**: affichage de la barre d'outils, yes pour oui, no pour non
- **scrollbars**: affichage de la barre de défilement, yes pour oui, no pour non

Exemple :

```
<html>
    <head>
        <title>Ouvrir une pop-up en javascript</title>
        <script type="text/javascript">
             function open_infos()
               {
            fen1= window.open('pageb.html','nom_de_ma_popup','menubar=no,
                  scrollbars=no, top=100, left=100, width=300,height=200');
                }
         </script>
    </head>
    <body>
   <a href="#null" onclick="javascript:open_infos();">Ouvrir la Pop-Up</a>
    </body>
</html>
```

La méthode **close()** adressée à une variable fenêtre permet de la fermer.

▪ SetTimeout et setInterval: Les délais en JavaScript

Les méthodes *setTimeout* et *setInterval* sont des méthodes de l'objet Window. On peut donc écrire *setTimeout* ou *window.setTimeout*.

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.

SetTimeout indique un délai avant exécution

La méthode (de l'objet Window) **setTimeout()** déclenche un miniteur, un compte à rebours au bout duquel un expresion JS passée en 1er paramètre sera exécutée; le temps d'attente est passé en 2ème paramètre.Autyrement dit elle définit une action à exécuter et un délai avant son exécution. Elle retourne un identifieur pour le processus.

• **Syntaxe :**

```
var TimeoutID = setTimeout("instructions", délai en millisecondes).
```

- ✔ instructions: c'est une chaîne de caractères contenant la formule de calcul ou le nom de la fonction appelée à l'issue du délai. Cette action n'est exécutée qu'une seule fois.
- ✔ `délai en millisecondes` : est une valeur numérique entière, exprimée en millisecondes.
- ✔ TimeoutID est un identificateur qui est utilisé seulement pour annuler l'évaluation avec la méthode "clearTimeout()".
- ✔ **Exemple:**

```
function mafonction() { ...}

setTimeout(mafonction, 5000);

La fonction sera exécutée après 5 secondes (5 000 millisecondes).
```

• La méthode **clearTimeout**(TimeoutID) annule le minuteur avant sa fin. Surtout utilisé pour arrêter des appels récursifs d'une fonction passée dans setTimeout()

Exemple :

```
<HTML>
<HEAD>
<script>
var w, id
function ouvrir() {
w=open("","Essai","width=50,height=100,resizable=1,scrollbars=1")
```

```
}
function essai() {
w.focus();
w.document.write("bonjour<BR>")
id= setTimeout ("essai()",1000)
}
function arreter() {
clearTimeout(id);
 w.focus();
 }
 </script></HEAD>
 <BODY>
 <form>
<INPUT type =Button value="Démarrer" onClick="ouvrir();essai()">
 <INPUT type =Button value="Arrêter" onClick="if (w != null) arreter()">
 <INPUT type =Button value="Fermer" onClick="if (w != null) w.close()">
 </form>
</BODY>
</HTML>
```

setInterval déclenche une opération à intervalles réguliers

Similaire à *setTimeout*, elle déclenche répétitivement la même action à intervalles réguliers.

```
setInterval("instructions", délai)
```

Exemple 1 :

```
setInterval("alert(('bip')", 10000);
```

Affiche un message toutes les dix secondes.

L'action sera recommencée jusqu'à ce que l'on quitte la page ou que *clearInterval* soit exécutée.

Exemple 2 :

```
<button onclick="start()">Lancer le décompte</button>
<div id="bip" class="display"></div>
<script>
var counter = 10;
var intervalId = null;
function finish()
{ clearInterval(intervalId);
 document.getElementById("bip").innerHTML = "TERMINE!"; }
function bip()
{ counter--;
```

```
if(counter == 0)
finish(); else {
document.getElementById("bip").innerHTML = counter + " secondes
restantes"; } }
function start(){
intervalId = setInterval(bip, 1000);
 }
</script>
```