FORMATION AFPA

- DEVELOPPEUR LOGICIEL –

(NIVEAU III)

JJP

JAVASCRIPT

INTRODUCTION : Qu'est ce que le JavaScript ?

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs.

Ces scripts vont être gérés et exécutés par le navigateur lui-même sans devoir faire appel aux ressources du serveur.
Langage non compilé mais interprété.

JavaScript n'est pas Java :
Il importe de savoir que **JavaScript** est totalement différent de **Java**.
Bien que les deux soient utilisés pour créer des pages Web évoluées, bien que les deux reprennent le terme Java, nous avons là deux outils informatiques bien différents.

Javascript	Java
Code intégré dans la page Html	Module (applet) distinct de la page Html
Code interprété par le browser au moment de l'exécution	Code source compilé avant son exécution
Codes de programmation simples mais pour des applications limitées	Langage de programmation beaucoup plus complexe mais plus performant
Permet d'accéder aux objets du navigateur	N'accède pas aux objets du navigateur
Confidentialité des codes nulle (code source visible)	Sécurité (code source compilé)

FORMATION AFPA

- DEVELOPPEUR LOGICIEL –

JJP

(NIVEAU III)

JAVASCRIPT

DEROULEMENT DU COURS

I.] Variables

II.] Insérer le JavaScript dans la page HTML

III.] Les fonctions

IV.] Les opérateurs

V.] Les conditions

VI.] Les boucles

VII.] Les tableaux

VIII.] Les événements

IX.] DOM

X.] Les fonctions natives prédéfinies

XI.] Utilisez la console JavaScript des navigateurs

XII.] Réalisation en JavaScript des exercices pseudo-codes
(Avec utilisation de la méthode document.nom_de_la_forme.nom_champ.value)

XIII.] Méthode getElementById() et IDs dans la page HTML5
(Reprendre les exercices réalisés précédemment)

XIV.] "this", className, innerHTML, visibilité, les selects, les cases à cocher

XV.] Les cookies

XVI.] Les expressions régulières

JAVASCRIPT

I.] Variables

Variable non typée : l'affectation type implicitement.
Déclaration par le mot clé *var*.
Variable locale dans les fonctions.
Variable globale hors des fonctions.
Variable déclarée mais non affectée : *undefined*
Null pour noter l'absence de valeur.

Déclaration de variables :

var myVariable = 2;

```
//Déclaration de i, de j et de k.
var i, j, k;

//Affectation de i.
i = 1;

//Déclaration et affectation de prix.
var prix = 0;
```

En **JavaScript**, il n'existe pas de portée lexicale de type bloc comme en **Java** (ou dans de nombreux autres langages). La portée des variables est au niveau de la fonction où elles sont déclarées, c'est-à-dire que toute variable déclarée dans le corps d'une fonction y est utilisable, peu importe où se situe sa déclaration.

La portée lexicale d'une variable locale à une fonction A s'étend aux déclarations de fonctions faites à l'intérieur de A, et ainsi de suite. L'inverse n'est pas vrai.

Les noms de variables ne peuvent pas être les noms suivants (qui sont des noms réservés) :

- abstract
- boolean break byte
- case catch char class
- default do double
- else extends
- false final finally float
- goto
- if, implements, import, in, instanceof, int, interface
- long
- native, new, null

JAVASCRIPT

- package, private, protected, public
- return
- short, static, super, switch, synchronized
- this, throw, throws, transient, true, try
- var, void
- while, with

Nom de variable correct	Nom de variable incorrect	Raison
Variable	Nom de Variable	comporte des espaces
Nom_De_Variable	123	commence par un chiffre
nom_de_variable	toto@mailcity.com	caractère spécial @
nom_de_variable_123	Nom-de-variable	signe - interdit

JAVASCRIPT

II.] Insérer le JavaScript dans la page HTML

```
<script type="text/javascript" src="js/iframe.js"></script>
```

OU BIEN :

```
<script language="JavaScript">
// code JavaScript

</script>
```

OU BIEN :

```
<script type="text/javascript">
// code JavaScript

</script>
```

JAVASCRIPT

III.] Les fonctions

Les fonctions doivent toujours avoir un nom, posséder les parenthèses ouvrantes et fermantes (pour y accueillir les éventuels paramètres) et commencer par une accolade { et se fermer par l'autre accolade }

```
<script language="JavaScript">
        function nom_de_ma_fonction()        {

        }
</script>
```

Appel à cette fonction :

```
<a href="javascript:nom_de_ma_fonction()">appeler ma fonction JS</a>
```

JAVASCRIPT

IV.] Les opérateurs

Tous les opérateurs arithmétiques, de comparaison, conditionnels ou logiques ont la même syntaxe qu'en **C** ou en **Java**.

Qu'est-ce qu'un opérateur ?

Les opérateurs sont des symboles qui permettent de manipuler des variables, ie d'effectuer des opérations, les évaluer... On distingue plusieurs types d'opérateurs :

les opérateurs de calcul
les opérateurs d'assignation
les opérateurs d'incrémentation
les opérateurs de comparaison
les opérateurs logiques

Les opérateurs de calcul

Les opérateurs de calcul permettent de modifier mathématiquement la valeur d'une variable

Opérateur	Dénomination	Effet	Exemple	Résultat (x vaut 7)
+	addition	Ajoute deux valeurs	x+3	10
-	soustraction	Soustrait deux valeurs	x-3	4
*	multiplication	Multiplie deux valeurs	x*3	21
/	division	Divise deux valeurs	x/3	2.3333333
%	modulo	Reste de la division entière	x%3	1
=	affectation	Affecte une valeur à une variable	x=3	Met la valeur 3 dans la variable x

Les opérateurs d'assignation

Ces opérateurs permettent de simplifier des opérations telles qu'ajouter une valeur dans une variable et stocker le résultat dans la variable. Une telle opération s'écrirait habituellement de la façon suivante par exemple : x=x+2 Avec les opérateurs d'assignation il est possible d'écrire cette opération sous la forme suivante : x+=2 Ainsi, si la valeur de x était 7 avant opération, elle sera de 9 après.
Les autres opérateurs du même type sont les suivants :

Opérateur	Effet	Exemple
+=	addition deux valeurs et stocke le résultat dans la variable (à	x+=2 équivaut à x=x+2
-=	soustrait deux valeurs et stocke le résultat dans la variable	x-=2 équivaut à x=x-2
*=	multiplie deux valeurs et stocke le résultat dans la variable	x*=2 équivaut à x=x*2
/=	divise deux valeurs et stocke le résultat dans la variable	x/=2 équivaut à x=x/2

JAVASCRIPT

Les opérateurs d'incrémentation

Opérateur	Dénomination	Effet	Syntaxe	Résultat (x vaut 7)
++	Incrémentation	Augmente d'une unité la variable	x++	8
--	Décrémentation	Diminue d'une unité la variable	x--	6

Ce type d'opérateur permet de facilement augmenter ou diminuer d'une unité une variable. Ces opérateurs sont très utiles pour des structures telles que des boucles, qui ont besoin d'un compteur (variable qui augmente de un en un). Un opérateur de type x++ permet de remplacer des notations telles que x=x+1ou bien x+=1

Les opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat (x vaut 7)
== A ne pas confondre avec le signe d'affectation (=)	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	x==3	1 si x est égal à 3, sinon 0
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	x<3	1 si x est inférieur à 3, sinon 0
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	x<=3	1 si x est inférieur à 3, sinon 0
>	opérateur de	Vérifie qu'une variable est	x>3	1 si x est supérieur à
	supériorité stricte	strictement supérieure à une valeur		3, sinon
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	x>=3	1 si x est supérieur ou égal à 3, sinon 0
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	x !=3	1 si x est différent de 3, sinon 0

Les opérateurs logiques (booléens)

Ce type d'opérateur permet de vérifier si plusieurs conditions sont vraies :

Opérateur	Dénomination	Effet	Syntaxe
||	OU logique	Vérifie qu'une des conditions est réalisée	((condition1)||(condition2))
&&	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1)&&(condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

JAVASCRIPT

Les priorités

Lorsque l'on associe plusieurs opérateurs, il faut que le navigateur sache dans quel ordre les traiter, voici donc dans l'ordre décroissant les priorités de tous les opérateurs :

()	parenthèses
++ -- !	opérateurs unaires
* / %	mult,div,modulo
+ -	addition,soustraction
<< >>	décalage de bits
< > <= >=	opérateurs relationnels
== !=	égalité
&	ET binaire
^	OU exclusif binaire(XOR)
|	OU binaire
&&	ET logique
||	OU logique
= += -= *= /= %= ^=	Affectations diverses

Cela signifie que , dans une expression complexe, où apparaissent plusieurs opérateurs, JavaScript les interprètera en tenant compte de leur priorité. Si plusieurs priorités se trouvent dans la même expression et sont de même niveau ce sera l'opérateur le plus à gauche qui sera effectué en premier.
Grâce aux parenthèses qui sont toujours prioritaires, vous pouvez définir vous-même l'ordre de calcul des opérateurs. Donc utilisez le plus souvent les parenthèses qui rendront votre programme plus clair, plus lisible.

JAVASCRIPT

V.] Les conditions

Qu'est-ce qu'une structure conditionnelle?

On appelle structure conditionnelle les instructions qui permettent de tester si une condition est vraie ou non, ce qui permet de donner de l'interactivité à vos scripts par exemple.

L'instruction if

L'instruction if est la structure de test la plus basique, on la retrouve dans tous les langages (avec une syntaxe différente...). Elle permet d'exécuter une série d'instruction si une condition est réalisée.
La syntaxe de cette expression est la suivante :

```
if (condition testée)      {
        liste d'instructions
}
```

- la condition doit être entre des parenthèses
- il est possible de définir plusieurs conditions à remplir avec les opérateurs ET et OU (&& et ||) par exemple :
- if ((condition1)&&(condition2)) teste si les deux conditions sont vraies
- if ((condition1)||(condition2))exécutera les instructions si l'une ou l'autre des deux conditions est vraie
- s'il n'y a qu'une instruction, les accolades ne sont pas nécessaires.

L'instruction if ... else

L'instruction if dans sa forme basique ne permet de tester qu'une condition, or la plupart du temps on aimerait pouvoir choisir les instructions à exécuter en cas de non-réalisation de la condition...
L'expression if ... else permet d'exécuter une autre série d'instruction en cas de non-réalisation de la condition. La syntaxe de cette expression est la suivante :

```
if (condition testée)      {
        liste d'instructions
} else  {
        autre série d'instructions
}
```

Il est possible de faire un test avec la structure suivante :
(condition) **?** instruction si vrai : instruction si faux
la condition doit être entre des parenthèses
lorsque la condition est vraie, l'instruction de gauche est exécutée
lorsque la condition est fausse, l'instruction de droite est exécutée

Les instructions if…else peuvent être imbriquées

JAVASCRIPT

VI.] Les boucles

Les boucles sont des structures qui permettent d'exécuter plusieurs fois la même série d'instructions jusqu'à ce qu'une condition ne soit plus réalisée.

La façon la plus commune de faire une boucle est de créer un compteur (une variable qui s'incrémente, c'est-à-dire qui augmente de 1 à chaque tour de boucle) et de faire arrêter la boucle lorsque le compteur dépasse une certaine valeur.

La boucle for

L'instruction for permet d'exécuter plusieurs fois la même série d'instructions : c'est une boucle ! Dans sa syntaxe, il faut préciser le nom de la variable qui sert de compteur (éventuellement sa valeur de départ), la condition sur la variable pour laquelle la boucle s'arrête (basiquement une condition qui teste si la valeur du compteur dépasse une limite) et enfin une instruction qui incrémente (ou décrémente) le compteur.

```
for(compteur; condition; modification du compteur)      {
        liste d'instructions
}
```

Par exemple :

```
for(i=1; i<6; i++)          {
        alert(i)
}
```

Cette boucle affiche 5 fois la valeur de i, c'est-à-dire 1, 2, 3, 4 et 5 Elle commence à i = 1, vérifie que i est bien inférieur à 6, etc... jusqu'à atteindre la valeur i=6, pour laquelle la condition ne sera plus réalisée, la boucle s'interrompra et le programme continuera son cours.

Quelques remarques :
- il faudra toujours vérifier que la boucle a bien une condition de sortie (le compteur s'incrémente correctement)
- une instruction alert(i); dans une boucle est un bon moyen pour vérifier la valeur du compteur pas à pas !
- il faut bien compter le nombre de fois que l'on veut faire exécuter la boucle :

for (i=0;i<10;i++)	exécute 10 fois la boucle (i de 0 à 9)
for (i=0;i<=10;i++)	exécute 11 fois la boucle (i de 0 à 10)
for (i=1;i<10;i++)	exécute 9 fois la boucle (i de 1 à 9)
for (i=1;i<=10;i++)	exécute 10 fois la boucle (i de 1 à 10)

L'instruction while

L'instruction while représente un autre moyen d'exécuter plusieurs fois la même série d'instructions.

```
while (condition testée) {
        liste d'instructions
}
```

Cette instruction exécute la liste d'instructions tant que (while signifie tant que) la condition est réalisée.
La condition de sortie pouvant être n'importe quelle structure conditionnelle, les risques de boucle infinie (boucle dont la condition est toujours vraie) existent, c'est-à-dire qu'elle risque de provoquer un plantage du navigateur !

JAVASCRIPT

VII.] Les tableaux

Introduction à la notion de tableau

Les variables de Javascript ne permettent de stocker qu'une seule donnée à la fois.
Or, étant donné qu'il est souvent utile de manipuler de nombreuses données, le concept de variable se révéle parfois insuffisant, car il devient difficile de gérer un grand nombre de variable distinctes.

Pour y remédier Javascript propose une structure de données permettant de stocker l'ensemble de ces données dans une "variable commune" : le tableau.

Un tableau, en Javascript, est donc une variable pouvant contenir plusieurs données indépendantes, indexées par un numéro, appelé *indice*.
L'indice d'un tableau est ainsi l'élément permettant d'accéder aux données qui y sont stockées.

Tableaux multidimensionnels

Lorsque le tableau est composé uniquement de variables, on parle de tableau monodimensionnel (ou unidimensionnel).
Voici une manière de représenter un tableau unidimensionnel :

Indice	0	1	2	3
Donnée	donnée 1	donnée 2	donnée 3	donnée 4

Remarquez que le premier élément d'un tableau porte toujours l'indice 0 !
Dans un tableau à n éléments, le n^{ième} élément porte ainsi l'indice n-1.

Lorsqu'un tableau contient lui-même d'autres tableaux on parle alors de tableaux multidimensionnels.
Voici une représentation d'un exemple de tableau multidimensionnel :

<table>
<tr><td>0</td><td colspan="3">1</td><td>2</td><td colspan="2">3</td></tr>
<tr><td>donnée 1 (variable)</td><td colspan="3">donnée 2 (tableau)<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>donnée 1</td><td>donnée 2</td><td>donnée 3</td></tr></table></td><td>donnée 3 (variable)</td><td colspan="2">donnée 4 (tableau)<table><tr><td>0</td><td>1</td></tr><tr><td>donnée 1</td><td>donnée 2</td></tr></table></td></tr>
</table>

JAVASCRIPT

Création de tableau

Le langage Javascript fournit plusieurs façons de créer un tableau :

```
var MonTableau = [];
```

```
var MonTableau = new Array();
```

La première méthode est conseillé.

Il est possible d'initialiser des valeurs a la creation

```
var MonTableau = ["donnée 1", "donnée 2"];
```

```
var MonTableau = new Array("donnée 1", "donnée 2");
```

Accès aux données

L'accès aux éléments du tableau se fait en écrivant le nom du tableau suivi de crochets contenant l'indice de l'élément.

```
var MonTableau = ["Teebo",
   "Eaulive",
   "Asevere",
   "Kalamit",
   "Serge",
   "Chat_Teigne",
   "BmV"];

alert("Le 4ième élément est "+MonTableau[3]);

// Affichera "Le 4ième élément est Kalamit"
```

JAVASCRIPT

Manipulation de tableaux

Le langage Javascript propose l'objet Array, comportant de nombreuses méthodes permettant de manipuler les tableaux, c'est-à-dire d'insérer, supprimer, ou extraire des éléments dans le tableau et également de trier les élément du tableau.

Tableaux associatifs

Il est possible d'utiliser des indices personnalisés pour indexer des valeurs, on parle alors de tableau associatif. Un tableau associatif n'est autre qu' un objet javascript ou json

Indice	"Paul"	"André"	"Pierre"	"Jean-François"
Donnée	16	22	12	25

Affectation de valeurs

Pour créer un tableau associatif, il suffit de déclarer un objet par l'intermédiaire d'une variable, puis pour chaque indices on écrire le nom de l' indice suivit de ":" on affecter une donnée et on terminer par une virgule.

```
var tableau_associa={"valeur1":10,"valeur2":55,"valeur3":30"};
```

on peut simplifier la lisibilité en faisant un saut de ligne âpres la virgule.

```
var tableau_associa={
"valeur1":10,
"valeur2":55,
"valeur3":30
};
```

il est aussi possible d'affecter des valeur après la création de l'objet

```
var tableau_associa={}
tableau_associa.valeur1=10,
tableau_associa.valeur2=55,
tableau_associa.valeur3=30
};
```

JAVASCRIPT

Accès aux données

L'accès aux éléments du tableau associative se fait en écrivant
le nom du tableau suivi du nom de l'indice de l'élément.

```
var tableau_associa={
"valeur1":10,
"valeur2":55,
"valeur3":30
};

alert(tableau_associa.valeur2);

// Affichera "55"
```

JAVASCRIPT

VIII.] Les événements

Qu'appelle-t-on un événement?

Les évènements sont des actions de l'utilisateur qui vont pouvoir donner lieu à une interactivité.
L'événement clic de souris est le seul que le html gère. Grâce à JavaScript il est possible d'associer des fonctions, des méthodes à des événements tels que le passage de la souris au-dessus d'une zone, le changement d'une valeur, ...

Ce sont les gestionnaires d'événements qui permettent d'associer une action à un événement. La syntaxe d'un gestionnaire d'événement est la suivante :

onevenement="Action_JavaScript_ou_Fonction();"

Les gestionnaires d'événements sont associés à des objets, et leur code s'insère dans la balise de ceux-ci.

Liste de quelques événements

Evénement	Gestionnaire d'événement	Description
Click	onclick	l'utilisateur clique sur l'élément associé à l'événement
Load	onload	le navigateur de l'utilisateur charge la page en cours
Unload	onunload	le navigateur de l'utilisateur quitte la page en cours
MouseOver	onmouseover	l'utilisateur positionne le curseur de la souris au-dessus d'un élément
MouseOut	onmouseout	le curseur de la souris quitte un élément
Focus	onfocus	l'utilisateur donne le focus à un élément, c'est-à-dire que cet élément est sélectionné comme étant l'élément actif
Blur	onblur	l'élément perd le focus, c'est-à-dire que l'utilisateur clique hors de cet élément, celui-ci n'est alors plus sélectionné comme étant l'élément actif
Change	onchange	l'utilisateur modifie le contenu d'un champ de données
Select	onselect	l'utilisateur sélectionne un texte (ou une partie d'un texte) dans un champ de type "text" ou "textarea"
Submit	onsubmit	Se produit lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire (le bouton qui permet d'envoyer le formulaire)

JAVASCRIPT

Quelques objets auxquels on peut associer des événements

Chaque événement ne peut pas être associé à n'importe quel objet. Un événement onChange ne peut pas s'appliquer à un lien hypertexte

Objet	Evénements associables
Lien hypertexte	onclick, onmouseover, onmouseout
Page du navigateur	onload, onunload
Bouton, Case à cocher, Bouton radio, Bouton	onclick
Liste de sélection d'un formulaire	onblur, onchange, onfocus
Bouton Submit	onsubmit
Champ de texte et zone de texte	onblur, onchange, onfocus, onselect

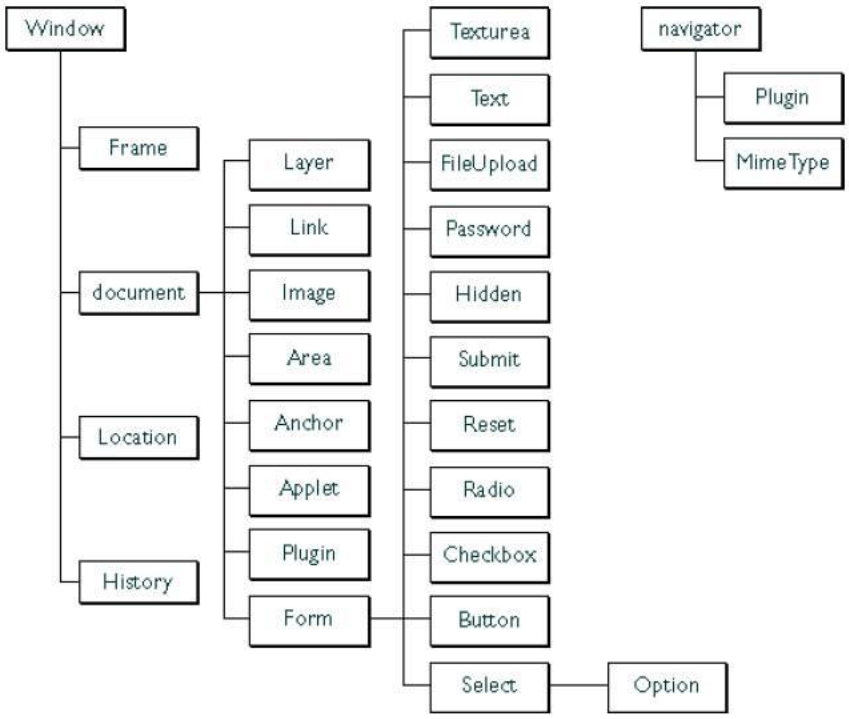
JAVASCRIPT

IX.] DOM

Définition :
Le Document Object Model est un standard du W3C qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents HTML3 et XML4.

Le document peut ensuite être traité et les résultats de ces traitements peuvent être réincorporés dans le document tel qu'il sera présenté.

Arborescence :
DOM permet de construire une arborescence de la structure d'un document et de ses éléments.
DOM est utilisé pour pouvoir modifier facilement des documents **XML** ou accéder au contenu des pages web.



JAVASCRIPT

X.] Les fonctions natives prédéfinies

Je vous prie de voir les 3 sites suivants :

http://www.toutjavascript.com/reference/

http://www.w3schools.com/jsref/

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux

XI.] Utilisez la console JavaScript des navigateurs

Je vous prie de voir le site suivant :

http://www.alsacreations.com/astuce/lire/1436-console-javascript.html

XII.] Réalisation en JavaScript des exercices pseudo-codes

(Avec utilisation de la méthode document.nom_de_la_forme.nom_champ.value)

XIII.] Méthode getElementById() et IDs dans la page HTML5

(Reprendre les exercices réalisés précédemment)

XIV.] "this", className, innerHTML, visibilité, les selects, les cases à cocher

JAVASCRIPT

XV.] Les cookies

Les cookies ont été inventés par Netscape afin de donner une "mémoire" aux serveurs et navigateurs Web. Le protocole HTTP, qui gère le transfert des pages Web vers le navigateur ainsi que les demandes de pages du navigateur vers le serveur, est dit *state-less* (sans état) : cela signifie qu'une fois la page envoyée vers le navigateur, il n'a aucun moyen d'en garder une trace. Vous pourrez donc venir deux, trois, cent fois sur la page, le serveur considérera toujours qu'il s'agit de votre première visite.

Cela peut être gênant à plusieurs titres : le serveur ne peut pas se souvenir si vous êtes authentifié à une page protégée, n'est pas capable de conserver vos préférences utilisateur, etc. En résumé, il ne peut se souvenir de rien ! De plus, lorsque la personnalisation a été créée, cela est vite devenu un problème majeur.

Les cookies ont été inventés pour remédier à ces problèmes. Il existe d'autres solutions pour les contourner, mais les cookies sont très simples à maintenir et très souples d'emploi.

Fonctionnement des cookies

Un cookie n'est rien d'autre qu'un petit fichier texte stocké par le navigateur. Il contient certaines données :

1. Une paire nom / valeur contenant les informations.
2. Une date d'expiration au-delà de laquelle il n'est plus valide.
3. Un domaine et une arborescence qui indiquent quel répertoire de quel serveur y aura accès.

Dès que vous demandez une page Web à laquelle le cookie peut être envoyé, celui-ci est ajouté dans l'en-tête HTTP. Les programmes côté serveur peuvent alors le lire et décider, par exemple, si vous avez le droit de voir la page ou si vous voulez que les liens soient jaunes avec un fond vert.

Ainsi, chaque fois que vous visitez la page d'où vient le cookie, les informations vous concernant sont disponibles. C'est parfois bien pratique, mais cela peut aussi nuire à votre vie privée. Heureusement, la plupart des navigateurs vous permettent de gérer les cookies (vous pouvez donc supprimer ceux provenant des sites publicitaires, par exemple).

Les cookies peuvent aussi être lus par JavaScript. Ils sont principalement destinés à conserver vos préférences.

Date d'expiration

Chaque cookie doit avoir une date d'expiration au-delà de laquelle il ne sera plus valide et supprimé. Si vous ne renseignez pas cette date, le cookie sera supprimé à la fermeture du navigateur. Cette date est exprimée au format UTC (Greenwich).

JAVASCRIPT

Les cookies peuvent être créés, lus et supprimés par JavaScript. Ils sont accessibles à travers la propriété document.cookie. Vous pouvez traiter cette propriété comme une chaîne (bien que cela n'en soit pas réellement une). Vous n'avez accès qu'au couple nom / valeur.

Pour créer un cookie sur ce domaine avec une paire nom / valeur valant 'ppkcookie1=testcookie' et expirant le 28 février 2010, il faut faire :

document.cookie = 'ppkcookie1=testcookie; expires=Sun, 28 Feb 2010 00:00:00 UTC; path=/'

1. D'abord, la paire nom / valeur ('ppkcookie1=testcookie').
2. Puis un point-virgule et un espace.
3. La date d'expiration dans un format correct (expires=Sun, 28 Feb 2010 00:00:00 UTC).
4. De nouveau un point-virgule et un espace.
5. Le domaine (path=/).

La syntaxe est très stricte, ne la changez pas (bien sûr, les scripts vous permettront de gérer ces valeurs) !

3 fonctions utiles :

```
function createCookie(name,value,days) {
	var expires= "";
	if (days) {
		var date = new Date();
		date.setTime(date.getTime()+(days*24*60*60*1000));
		expires = "; expires="+date.toGMTString();
	}
	else expires = "";
	document.cookie = name+"="+value+expires+"; path=/";
}

function readCookie(name) {
	var nameEQ = name + "=";
	var ca = document.cookie.split(';');
	for(var i=0;i < ca.length;i++) {
		var c = ca[i];
		while (c.charAt(0)==' ') c = c.substring(1,c.length);
		if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length,c.length);
	}
	return null;
}
```

JAVASCRIPT

```
function eraseCookie(name) {
        createCookie(name,"",-1);
}
```

Exemple d'appels :

```
var myStrCookie= "1:2|5:7";
createCookie("afpalab ", myStrCookie, 1);
var myCntCookie= readCookie("afpalab");
eraseCookie("afpalab ");
```

JAVASCRIPT

XVI.] Les expressions régulières

Syntaxe

reg=new RegExp(String motif[, String type])

Description

Objet Expression régulière
Le paramètre motif décrit le format de chaîne à trouver.
Le paramètre type décrit le type d'expression régulière.
Si type vaut "g", l'expression sera analysée globalement sur l'ensemble de la chaîne.
Si type vaut "i", l'expression sera analysée indifféremment sur les majuscules ou les minuscules.
type peut donc valoir "", "g", "i" ou "gi"

Il existe deux syntaxes équivalentes pour créer une expression régulière :

```
var reg=new RegExp("[0-9]+","g")
var reg=/[0-9]+/g
```

Les expressions régulières sont le plus souvent utilisées dans les méthodes match(), replace() et split() de l'objet String.

Propriété

$1..$9 (Contenu de l'expression parenthésée 1 à 9)

Méthodes

compile() (Modifie le motif d'une expression régulière)
exec() (Retourne la première sous-chaîne correspondant au motif)
test() (Teste l'expression régulière sur une chaîne)

JAVASCRIPT

Exemple Surligne un mot

Code source

```
<SCRIPT language="JavaScript">

        var chaine="Les chiens et les chiennes, les chats et les oiseaux";
        var reg=new RegExp("(chien)", "g");
        document.write("Chaîne d'origine : " + chaine + "<BR>");
        document.write("Chaîne traitée : " + chaine.replace(reg,"<mark>$1</mark>") + "<BR>");

</SCRIPT>
```

Résultat

Chaîne d'origine : Les chiens et les chiennes, les chats et les oiseaux
Chaîne traitée : Les chiens et les chiennes, les chats et les oiseaux

Explication

Cet exemple montre comment surligner une occurrence d'un mot.
Le motif (chien) de l'expression régulière permet de trouver tous les mots (chien).
Dans l'appel à replace(), le second paramètre indique comment remplacer chien.
Le symbole $1 représente la première expression entre parenthèse du motif.
Donc, "<mark>$1</mark>" remplace toutes les occurrences chien par "<mark>chien</mark>"

Exemple d'une fonction qui permet de tester si la chaine saisie ne contient que des lettres :

```
function isPureAlphabet(sStr)
{
        var reg = new RegExp('[a-z]+', 'gi');
        return(reg.test(sStr));
}
```

JAVASCRIPT

PUIS MISE EN PRATIQUE AVEC LE PROJET FIL ROUGE

- en mode test, et non en mode version finale -

- Voir avec le formateur avant de commencer -