

TP TIC (J S)

Matière : Technologie de l'information et de la communication
Classe : 4 SI
Enseignant : Ilahi Néjib

Durée : 2 heures
Date : Janvier 08
A.S :2007/2008



TP N°4 (Introduction au J S):

Objectifs :

- Savoir insérer du code JS dans une page Web.
- Savoir accéder aux objets formant une page Web en écrivant de code JS.

I- Qu'est ce que le Javascript :

Javascript est un langage de scripts qui incorporé aux balises Html, permet d'améliorer la présentation et l'interactivité des pages Web.

Javascript est donc une extension du code Html des pages Web. Les scripts, *qui s'ajoutent ici aux balises Html*, peuvent en quelque sorte être comparés aux macros d'un traitement de texte.

Ces scripts vont être gérés et exécutés par le browser lui-même sans devoir faire appel aux ressources du serveur. Ces instructions seront donc traitées en direct et surtout sans retard par le navigateur.

Javascript a été initialement développé par Netscape et s'appelait alors LiveScript. Adopté à la fin de l'année 1995, par la firme Sun (qui a aussi développé Java), il prit alors son nom de Javascript.

Javascript n'est donc pas propre aux navigateurs de Netscape (bien que cette firme en soit un fervent défenseur). Microsoft l'a d'ailleurs aussi adopté à partir de son Internet Explorer 3. On le retrouve, de façon améliorée, dans Explorer 4.

II- Comment Insérer un code Javascript :

Pour écrire et tester des codes Javascript il faut un navigateur Web qui reconnaît Javascript (IE, Netscape, Mozilla Firefox,...), un éditeur de traitement de texte (Notepad++, bloc note, Word pad...) et **une solide reconnaissance du langage HTML.**

Le code du javascript a la forme suivante :

```
<script language="javascript">
. Instruction 1 ;
. Instruction 2 ;
. Instruction 3 ;
.
.
.
. Instruction n ;
</script>
```

➡ La question qui se pose où on va insérer ce code JS dans la page HTML ?

- ❖ Il existe 4 manières d'insérer le code Javascript dans une page HTML

1- Insertion pour exécution directe :

C'est-à-dire le code s'exécute automatiquement lors de chargement de la page HTML au niveau du navigateur. Le code javascript est placé dans le corps de la page html c'est-à-dire entre <body>...</body>.

❖ **Exemple1** : Taper le code JS suivant:

```
<html>
<head><title>Exemple1</title></head>
<body>
<center><b>Exemple d'exécution directe du code javascript</b></center><br>

<script language="javascript">
  alert("Ce message est affiché automatiquement");
</script>

</body></html>
```



2- Exécution en fonction d'un événement :

Le code est d'abord lu par le navigateur (exemple : IE), stocké en mémoire pour ne s'exécuter que sur demande. Dans ce cas, le code JS est placé dans le page HTML entre <HEAD></HEAD>, le code s'exécute seulement lors d'un événement généré par intervention de l'utilisateur.

❖ **Exemple2** : Taper le code JS suivant:

```
<html>
<head><title>Exemple2</title>
<script language="javascript">
function affiche(){
  alert("Ce message ne s'affiche que lorsque vous avez cliqué sur le bouton ");}
</script>
</head>
<body>
<form name="f1">
<input type="button" name="evenement" value="cliquer ici" onclick="affiche()">
</form>
</body></html>
```



❖ **Commentaire** : L'appel de la fonction affiche() n'est possible que lorsque **l'événement clic** réalisé par l'utilisateur est déclenché, c'est la raison pour laquelle le langage JS est **un langage événementiel**.

3- Insertion du code JS à l'intérieur d'une balise html :

Certaines balises HTML acceptent de réagir à des événements soit provoqué par l'intervention d'utilisateur soit provoqué par l'intervention du navigateur lui-même (chargement d'une page HTML par exemple). Dans ce cas le code Javascript peut être insérer directement au niveau de la balise comme suit :

❖ **Exemple3** : Taper le code JS suivant:

```
<html><head><title>Exemple3</title></head>

<BODY onLoad="javascript: alert('Bienvenue à cette page') "; onUnload="javascript: alert('Au revoir'); ">
<b><u>Insertion du code JS à l'intérieur d'une balise html</u></b>

</BODY>
</HTML>
```

❖ **Commentaire** :

- onload est un événement qui s'exécute lorsque la page est chargée par le navigateur.
- Unload de même que onload mais s'exécute lorsque l'utilisateur quitte la page



4- Appel du code JS à partir d'un fichier d'extension ".js" :

On peut faire appel aux codes JS écrits dans des fichiers qui portent l'extension ".js" dans notre page HTML. Intérêt de cette procédure est de réutiliser ces codes dans d'autres pages HTML autant de fois que vous voulez.

Exemple : Voir exemple livre page 83

III- Les objets et leur hiérarchie

Javascript va diviser une page en objets et surtout va vous permettre d'accéder à ces objets, d'en retirer des informations et de les manipuler.

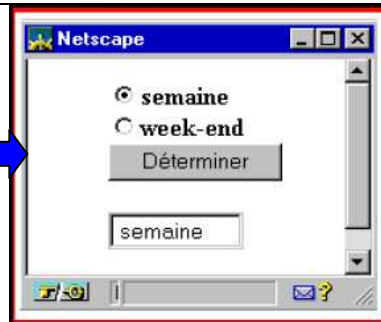
Voyons d'abord une illustration des différents objets qu'une page peut contenir.

Vous avez chargé la page suivante :

Javascript va diviser cette page en objets et surtout va vous permettre d'accéder à ces objets, d'en retirer des informations et de les manipuler. Voyons d'abord une illustration des différents objets qu'une page peut contenir. Vous avez chargé la page suivante :

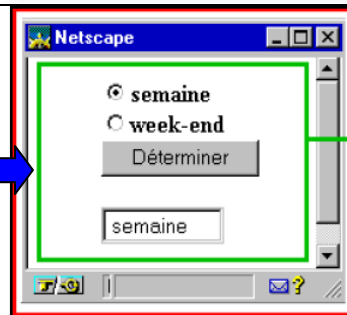


Cette page s'affiche dans une fenêtre. C'est **l'objet fenêtre**.



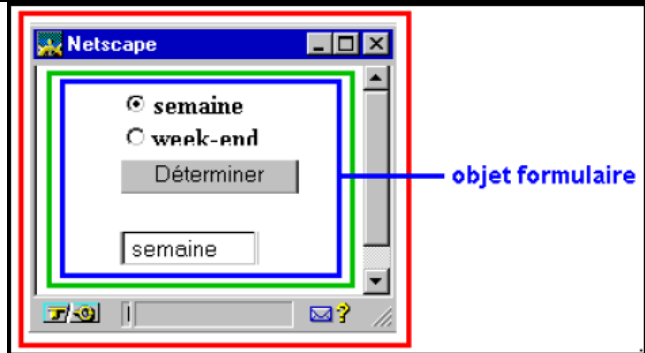
objet fenêtre

Dans cette fenêtre, il y a un document Html. C'est l'objet document. Autrement dit (et c'est là que l'on voit apparaître la notion de la hiérarchie des objets Javascript), l'objet fenêtre contient l'objet **document**.

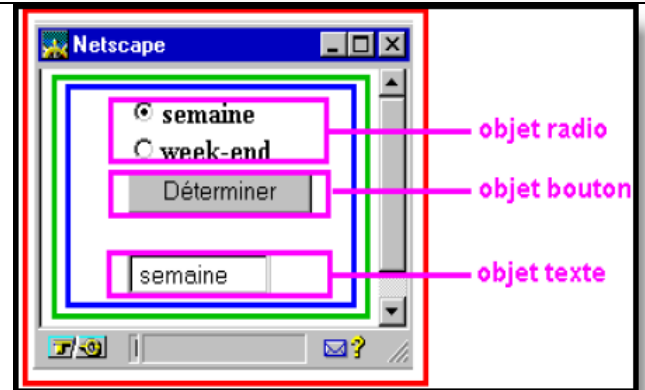


objet document

Dans ce document, on trouve un formulaire au sens Html. C'est l'objet **formulaire**. Autrement dit, l'objet fenêtre contient un objet document qui lui contient un objet formulaire.



Dans ce document, on trouve trois objets. Des boutons radio, un bouton classique et une zone de texte. Ce sont respectivement l'objet **radio**, l'objet **bouton**, l'objet **texte**. Autrement dit l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet radio, l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet bouton et l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet texte.



La hiérarchie de l'objet bouton de cet exemple est donc

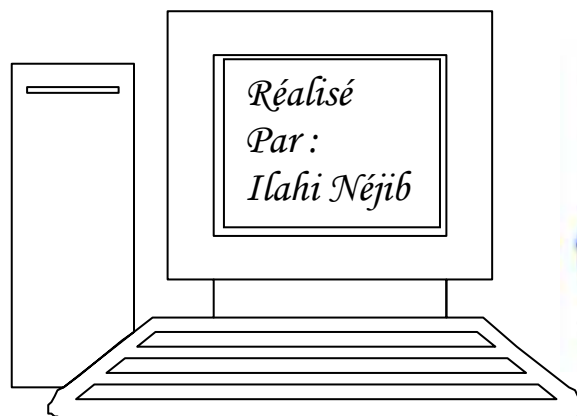
windows **document** **formulaire** **bouton**

Pour accéder à un objet (vous l'avez peut-être déjà deviné), il faudra donner le chemin complet de l'objet en allant du contenant le plus extérieur à l'objet à l'objet référencé.

Soit par exemple pour le bouton radio "semaine" : `(window).document.form.radio[0]`.

Nous avons mis l'objet window entre parenthèses car comme il occupe la première place dans la hiérarchie, il est repris par défaut par Javascript et devient donc facultatif.

Et enfin pour les puristes, Javascript n'est pas à proprement parler un langage orienté objet tel que C++ ou Java. On dira plutôt que Javascript est un langage basé sur les objets.



TP TIC (J S)

Matière : Technologie de l'information et de la communication
Classe : 4 SI
Enseignant : Ilahi Néjib

Durée : 2 heures
Date : Janvier 08
A.S :2007/2008



TP N°5 (Afficher du texte):

Objectifs :

- Savoir afficher du texte et des messages.
- Savoir la notion d'une méthode.

I- Les boîtes de dialogue :

Une boîte de dialogue est une fenêtre qui s'affiche au premier plan suite à un événement, et qui permet

- Soit d'avertir l'utilisateur
- Soit le confronter à un choix
- Soit lui demander de compléter un champ pour récupérer une information

I-1 : La méthode alert():

Alert() est une méthode de l'objet Window. Pour se conformer à la notation classique **nom_de_l'objet.nom_de_la_propriété**, on aurait pu noter window.alert(). Window venant en tête des objets Javascript, celui-ci est repris par défaut par l'interpréteur et devient en quelque sorte facultatif.

Si vous souhaitez que votre texte de la fenêtre alert() s'inscrive sur plusieurs lignes, il faudra utiliser le caractère spécial /n pour créer une nouvelle ligne.

Voici sa syntaxe :

```
alert(nom_de_la_variable);
```

```
alert('Chaîne de caractères');
```

```
alert('Chaîne de caractères' + nom_de_la_variable);
```

❖ Exemple1 : Taper le code suivant

```
<html><head><title>Exemple1</title>
<script language="javascript">
  function msg(){
    alert("Attention!");}
</script></head>
<body>
<center><b><u>La méthode Alert()</u></b></center><br>
<form name= "f1">
<input type="button" value="Cliquer ici !!! " onclick="msg()" >
</form></body></html>
```



Résultat du code

Remarque : La chaîne de caractère peut (et doit dans certains cas) contenir des caractères marqués d'un antislash (\). Par exemple, si vous voulez écrire :

Message d'alerte :
Au feu!!

Il faudra écrire le script suivant :

```
alert('Message d\'alerte \nAu feu!!!');
```



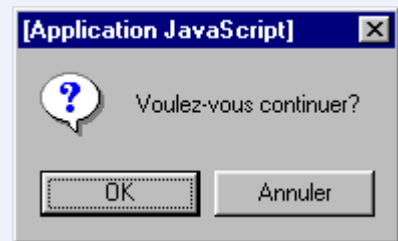
I-2 : La méthode Confirm():

La méthode *confirm()* est similaire à la méthode *alert()*, si ce n'est qu'elle permet un choix entre "OK" et "Annuler". Lorsque l'utilisateur appuie sur "OK" la méthode renvoie la valeur *true*. Elle renvoie *false* dans le cas contraire...Elle admet comme *alert()* un seul paramètre: une chaîne de caractères...

Sa syntaxe est : `confirm('Chaîne de caractères');`

❖ **Exemple2 :**

```
<html><head><title>Exemple2</title>
<script language="javascript">
  function choix(){
    ok= Confirm("Voulez vous continuer?");
    if(ok)
      {alert("Vous aimez JS !!! ");}
    else{ alert("Vous n\'aimez pas JS !!! ");}
  }
</script></head>
<body>
<center><b><u>La méthode confirm()</u></b></center><br>
<form name= "f1">
<input type="button" value="Cliquer ici !!! " onclick="choix()" >
</form></body></html>
```



Résultat du code

I-3 : La méthode prompt():

La méthode *prompt* est un peu plus évoluée que les deux précédentes puisqu'elle fournit un moyen simple de récupérer une information provenant de l'utilisateur, on parle alors de boîte d'invite. La méthode *prompt()* requiert deux arguments :

- le texte d'invite
- la chaîne de caractères par défaut dans le champ de saisie

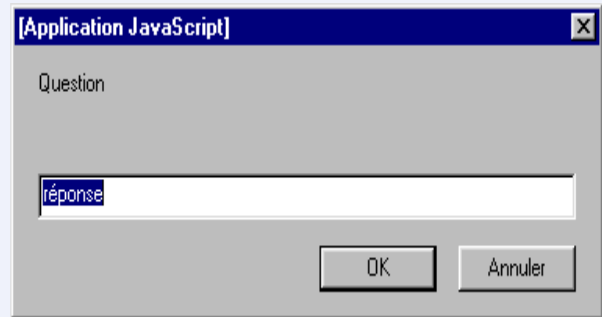
Sa syntaxe est donc la suivante :

```
prompt('Posez ici votre question','chaîne par défaut');
```

Cette boîte d'invite retourne la valeur de la chaîne saisie par l'utilisateur, elle retourne la valeur *null* si jamais aucun texte n'est saisi...

❖ Exemple3 :

```
<html><head><title>Exemple3</title>
<script language="javascript">
  function choix(){
    ch= prompt("Question","réponse");
    if(ch != null)
      alert(ch) ;
  }
</script></head>
<body>
<center><b><u>La méthode
prompt()</u></b></center><br>
<form name= "f1">
<input type="button" value="Cliquer ici !!! "
onclick="choix()" >
</form></body></html>
```



Résultat du code

II : Qu'appelle-t-on une méthode ?

Une méthode est une fonction associée à un objet, c'est-à-dire une action que l'on peut faire exécuter à un objet. Les méthodes des objets du navigateur sont des fonctions définies à l'avance par les normes HTML, on ne peut donc pas les modifier, il est toutefois possible de créer une méthode personnelle pour un objet que l'on a créé soi-même. Prenons par exemple une page HTML, elle est composée d'un objet appelé *document*. L'objet *document* a par exemple la méthode *write()* qui lui est associée et qui permet de modifier le contenu de la page HTML en affichant du texte. Une méthode s'appelle un peu comme une propriété, c'est-à-dire de la manière suivante :

```
window.objet1.objet2.methode()
```

Dans le cas de la méthode *write()*, l'appel se fait comme suit :

```
window.document.write()
```

Dans le cas de la méthode *alert()*, l'appel se fait comme suit :

`window.alert()` et puisque l'objet `windows` occupe la 1^{ère} place dans la hiérarchie il suffit d'écrire `alert()` tout court.



II -1 : La méthode Write :

Une La méthode *write()* de l'objet *document* permet de modifier de façon dynamique le contenu d'une page HTML. Voici la syntaxe de la méthode *write()* :

```
window.document.write(expression1, expression2, ...)
```

Cette méthode permet d'écrire le résultat des expressions passées en paramètre dans le document dans lequel elle est utilisée. Il est ainsi possible d'utiliser la méthode *write()* de différentes façons :

- soit en passant directement le texte en paramètres :
document.write("bonjour");
qui aura pour effet de concaténer la chaîne 'bonjour' à l'endroit où est placé le script

- soit en passant le texte par l'intermédiaire d'une variable :

```
Chaine='bonjour';  
document.write(Chaine);
```

Ce qui aura pour effet de concaténer la chaîne 'bonjour' (contenue dans la variable *Chaine*) à l'endroit où est placé le script

- soit en utilisant les deux :
- ```
Chaine='bonjour';
document.write('je vous passe le ' + Chaine);
```

Ce qui aura pour effet de concaténer la chaîne 'bonjour' (contenue dans la variable *Chaine*) à la suite de la chaîne de caractère 'je vous passe le' dans la page HTML

- soit en insérant directement une expression, qui sera évaluée dans un premier temps et dont le résultat sera ensuite affiché:
- ```
Chaine='La racine carrée de 2 vaut :';  
document.write(Chaine+Math.sqrt(2));
```

Il est notamment possible d'utiliser des balises HTML à l'intérieur même de la méthode *write* :

```
document.write('<font color="#FF0000">Bonjour</font>');
```

II -2: La méthode *writeln* :

La méthode *writeln()* fonctionne exactement comme la méthode *write()* à la seule différence qu'elle ajoute un retour chariot à la fin de la chaîne.

Or un retour chariot (en HTML) est ignoré par le navigateur (Rappel: un retour à la ligne se fait avec la balise `
`).

❖ Exemple 4:

```
<HTML>  
<head><title>Exemple4</title></head>  
<BODY>  
<H1>Ceci est du Html</H1>  
<SCRIPT LANGUAGE="Javascript">  
<  
document.write("<H1>Et ceci du Javascript</H1>");  
>  
</SCRIPT>  
</BODY>  
</HTML>
```

Ce qui donnera comme résultat :

Ceci est du Html

Et ceci du Javascript



TP TIC (J S)

Matière : Technologie de l'information et de la communication
Classe : 4 SI
Enseignant : Ilahi Néjib

Durée : 2 heures
Date : Janvier 08
A.S :2007/2008



TP N°6 (Variables & opérateurs en JS):

Objectifs :

- Savoir déclarer et manipuler des variables en JS.
- Savoir les opérateurs utilisés en JS.

I- Les variables :

Une variable est un objet repéré par son nom, pouvant contenir des données, qui pourront être modifiées lors de l'exécution du programme.

II-La déclaration de variables

Le Javascript étant très souple au niveau de l'écriture (à double tranchant car il laisse passer des erreurs...), la déclaration des variables peut se faire de deux façons:

- soit de façon explicite, en faisant précéder la variable du mot clé *var* qui permet d'indiquer de façon rigoureuse qu'il s'agit d'une variable :

```
var chaine= "bonjour"
```

- soit de façon implicite, en laissant le navigateur déterminer qu'il s'agit d'une déclaration de variable. Pour déclarer implicitement une variable, il suffit d'écrire le nom de la variable suivie du caractère = et de la valeur à affecter :

```
chaine= "bonjour"
```

Même si une déclaration implicite est tout à fait reconnue par le navigateur, il est plus rigoureux de déclarer les variables de façon explicite avec le mot *var*.

Voici un exemple dans lequel deux variables sont déclarées :

```
<html><head><SCRIPT language="Javascript">
  function f(){
  var MaVariable;
  var MaVariable2 = 3;
  MaVariable = 2;
  document.write(MaVariable*MaVariable2);}</SCRIPT>
  <body>
  <form name="f1">
  <input type="button" value="cliquer ici" onclick="f()"></form>
  </body></html>
```

III-Portée (visibilité) des variables

Selon l'endroit où une variable est déclarée, celle-ci pourra être accessible (visible) de partout dans le script ou bien uniquement dans une portion confinée du code, on parle de « portée » d'une variable.

Lorsqu'une variable est déclarée sans le mot clé *var*, c'est-à-dire **de façon implicite**, elle est accessible de partout dans le script (n'importe quelle fonction du script peut faire appel à cette variable). On parle alors de **variable globale**

La portée d'une variable déclarée **de façon explicite** (précédée du mot-clé *var*), dépend de l'endroit où elle est déclarée.

- Une variable déclarée au début du script, avant toute fonction, sera globale. Elle peut être utilisée n'importe où dans le script .
- Une variable déclarée explicitement dans une fonction aura une portée limitée à cette seule fonction, c'est-à-dire qu'elle est inutilisable ailleurs. On parle alors de « **variable locale** ».

Voici deux exemples permettant de l'illustrer :

```
<html><head>
<body>
<SCRIPT language="Javascript">
var a = 12;
var b = 4;

function MultipliePar2(b) {
    var a = b * 2;
    return a;
}
</SCRIPT><body>
<script language="javascript">
document.write("Le double de ",b," est ",MultipliePar2(b));
document.write("<br>");
document.write("La valeur de a est ",a);
</script>
</body></html>
```

Dans l'exemple ci-dessus, la variable *a* est déclarée explicitement en début de script, ainsi que dans la fonction. Voici ce qu'affiche ce script :

```
Le double de 4 est 8
La valeur de a est 12
```

Voici un autre exemple dans lequel *a* est déclarée implicitement dans la fonction :

```
<html><head>
<body>
<SCRIPT language="Javascript">
var a = 12;
var b = 4;
```

```
function MultipliePar2(b) {
    a = b * 2;
    return a;
}
</SCRIPT>
<body><script language="javascript">
document.write("Le double de ",b," est ",MultipliePar2(b));
document.write("<br>");
document.write("La valeur de a est ",a);
</script></body></html>
```

Voici ce qu'affiche ce script :

```
Le double de 4 est 8
La valeur de a est 8
```

IV-Les types de données dans les variables

En Javascript il n'est pas nécessaire de déclarer le type des variables, contrairement à des langages évolués tels que le langage C ou le Java pour lesquels il faut préciser s'il s'agit d'entier (*int*), de nombre à virgule flottante (*float*) ou de caractères (*char*).

En fait, le Javascript n'autorise la manipulation que de 4 types de données:

- des **nombres**: entiers ou à virgules
- des **chaînes de caractères** (string): une suite de caractères
- des **booléens**: des variables permettant de vérifier une condition. Les booléens peuvent prendre deux états :
 - *true* : si le résultat est vrai ;
 - *false* : lors d'un résultat faux.
- des **variables de type null**: un mot caractéristique pour indiquer que la variable ne contient aucune donnée.

❖ **Exemple** : Faire les exemples de livre pages (84-86).

V- Qu'est-ce qu'un opérateur?

Les opérateurs sont des symboles qui permettent de manipuler des variables, c'est-à-dire effectuer des opérations, les évaluer, ...

On distingue plusieurs types d'opérateurs:

- les opérateurs de calcul
- les opérateurs d'affectation
- les opérateurs d'incrémentatation
- les opérateurs de comparaison
- les opérateurs logiques



V-1 :Les opérateurs de calcul

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

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
+	opérateur d'addition	Ajoute deux valeurs	$x+3$	10
-	opérateur de soustraction	Soustrait deux valeurs	$x-3$	4
*	opérateur de multiplication	Multiplie deux valeurs	$x*3$	21
/	plus: opérateur de division	Divise deux valeurs	$x/3$	2.3333333
=	opérateur d'affectation	Affecte une valeur à une variable	$x=3$	Met la valeur 3 dans la variable x
%	opérateur modulo	Retourne le reste de la division entière de l'opérande de gauche par celle de droite	$x \% 2$	1

V-2 :Les opérateurs d'affectation

Ces opérateurs permettent de simplifier des opérations telles que *ajouter une valeur dans une variable et stocker le résultat dans la variable*. Une telle opérations 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 opérateurs de ce type sont les suivants:

Opérateur	Effet
+=	ajoute l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche.
-=	soustrait l'opérande de droite à l'opérande de gauche et stocke le résultat dans l'opérande de gauche.
*=	multiplie l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche.
/=	divise l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche.
%=	calcule le reste de la division entière de l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche.

Les opérateurs d'incrémentatation

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 lourdes telles que $x=x+1$ ou bien $x+=1$

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
++	Incrémentatation	Augmente d'une unité la variable	$x++$	8
--	Décrémentatation	Diminue d'une unité la variable	$x--$	6

V-3 :Les opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
== A ne pas confondre avec le signe d'affectation (=)!!	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	x==3	Retourne <i>True</i> si X est égal à 3, sinon <i>False</i>
===	opérateur d'identité	Vérifie l'identité de valeur et de type de deux valeurs	a===b	Retourne <i>True</i> si a est égal à b et est de même type, sinon <i>False</i>
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	x!=3	Retourne 1 si X est différent de 3, sinon 0
!==	opérateur de non identité	Vérifie la non identité de valeur et de type de deux valeurs, c'est-à-dire si les deux valeurs n'ont pas la même valeur ou bien sont de types différents.	a!==b	Retourne <i>True</i> si a est différent de b ou bien est de type différent, sinon <i>False</i>
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	x<3	Retourne <i>True</i> si X est inférieur à 3, sinon <i>False</i>
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	x<=3	Retourne <i>True</i> si X est inférieur ou égale à 3, sinon <i>False</i>
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	x>3	Retourne <i>True</i> si X est supérieur à 3, sinon <i>False</i>
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	x>=3	Retourne <i>True</i> si X est supérieur ou égal à 3, sinon <i>False</i>

V-4 :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	((expression1) (expression2))
&&	ET logique	Vérifie que toutes les conditions sont réalisées	((expression1)&&(expression2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)



TP TIC (J S)

Matière : Technologie de l'information et de la communication
Classe : 4 SI
Enseignant : Ilahi Néjib

Durée : 2 heures
Date : Janvier 08
A.S :2007/2008



TP N°7 (Les structures de contrôle & les fonctions en JS):

Objectifs :

- Savoir utiliser les différentes structures de contrôle.
- Savoir Déclarer une fonction en JS.

I- La structure conditionnelle :

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

La syntaxe de cette expression est la suivante :

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

- **Exemple** : `if (x==2) document.write("X vaut 2");`

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 réalisée) {  
    //liste d'instructions  
}  
else {  
    //autre série d'instructions  
}
```

II- La boucle For :

L'instruction *for* permet d'exécuter plusieurs fois la même série d'instructions: c'est une boucle!

Dans sa syntaxe, il suffit de préciser le nom de la variable qui sert de compteur (et 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.

La syntaxe de cette expression est la suivante :

```
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, 4,5
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'interrompt et le programme continuera son cours.

- il faudra toujours vérifier que la boucle a bien une condition de sortie (i.e le compteur s'incrémente correctement)
- une instruction *Alert(i)*; dans votre 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)

III- La boucle While :

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

La syntaxe de cette expression est la suivante :

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

Cette instruction exécute la liste d'instructions **tant que** (*while* est un mot anglais qui signifie *tant que*) la condition est réalisée.

Remarque : L'instruction Répéter...Jusqu'à en JS à pour syntaxe :

```
do{
    liste d'instructions
} while (condition réalisée) ;
```

La condition de sortie pouvant être n'importe quelle structure conditionnelle, les risques de boucle infinie (boucle dont la condition est toujours vraie) sont grands, c'est-à-dire qu'elle risque de provoquer un plantage du navigateur!

III- L'instruction Switch...case :

L'instruction *switch* permet de faire plusieurs tests de valeurs sur le contenu d'une même variable. Ce branchement conditionnel simplifie beaucoup le test de plusieurs valeurs d'une variable, car cette opération aurait été compliquée (mais possible) avec des *if* imbriqués. Sa syntaxe est la suivante :

```
switch (Variable) {  
    case Valeur1:  
        Liste d'instructions;  
        break;  
    case Valeur2:  
        Liste d'instructions;  
        break;  
    case ValeurX:  
        Liste d'instructions;  
        break;  
    default:  
        Liste d'instructions;  
        break;  
}
```



- Remarque : Il est essentiel de terminer chaque bloc d'instruction par l'instruction *break* !
- *Applications : Faire les applications des pages 89-94.*

IV- Les fonctions en JS :

1. Définition d'une fonction :

Une fonction est un groupe de ligne(s) de code de programmation destiné à exécuter une tâche bien spécifique et que l'on pourra, si besoin est, utiliser à plusieurs reprises. De plus, l'usage des fonctions améliorera grandement la lisibilité de votre script.

En Javascript, il existe deux types de fonctions :

- les fonctions propres à Javascript. On les appelle des "méthodes". Elles sont associées à un objet bien particulier comme c'était le cas de la méthode `Alert()` avec l'objet `window`.
- les fonctions écrites par vous-même pour les besoins de votre script. C'est à celles-là que nous nous intéressons maintenant.

2. Déclaration d'une fonction :

Pour déclarer ou définir une fonction, on utilise le mot (réservé) `function`.

La syntaxe d'une déclaration de fonction est la suivante :

```
function nom_de_la_fonction(arguments) {  
    ... code des instructions ...  
}
```

3. Application : page 95

TP TIC (J S)

Matière : Technologie de l'information et de la communication
Classe : 4 SI
Enseignant : Ilahi Néjib

Durée : 2 heures
Date : Janvier 08
A.S :2007/2008



TP N°8 (Les formulaires et la gestion des événements en JS):

Objectifs :

- *Savoir utiliser les formulaires en JS.*
- *Savoir gérer les événements en JS.*

I- Les événements en JS :

Les événements sont des actions de l'utilisateur, qui vont pouvoir donner lieu à une interactivité. L'événement par excellence est le clic de souris, car c'est le seul que le HTML gère. Grâce au 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();"
```

Lorsqu'il est utilisé dans un lien hypertexte, par exemple, la syntaxe sera la suivante :

```
<A href="URL" "onEvenement='Action_Javascript_ou_Fonction() ;'">Lien</a>
```

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

II- Liste des événements:

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

Événement	Description
Abort (onAbort)	Cet événement a lieu lorsque l'utilisateur interrompt le chargement de l'image
Blur (onBlur)	Se produit lorsque 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)	Se produit lorsque l'utilisateur modifie le contenu d'un champ de données.
Click (onClick)	Se produit lorsque l'utilisateur clique sur l'élément associé à l'événement.
dblclick	Se produit lorsque l'utilisateur double-clique sur l'élément associé à l'événement

(onDblclick)	(un lien hypertexte ou un élément de formulaire). Cet événement n'est supporté que par les versions de Javascript 1.2 et supérieures
dragdrop (onDragdrop)	Se produit lorsque l'utilisateur effectue un <i>glisser-déposer</i> sur la fenêtre du navigateur. Cet événement n'est supporté que par les versions de Javascript 1.2 et supérieures
error (onError)	Se déclenche lorsqu'une erreur apparaît durant le chargement de la page. Cet événement fait partie du Javascript 1.1.
Focus (onFocus)	Se produit lorsque l'utilisateur donne le focus à un élément, c'est-à-dire que cet élément est sélectionné comme étant l'élément actif
keydown (onKeyDown)	Se produit lorsque l'utilisateur appuie sur une touche de son clavier. Cet événement n'est supporté que par les versions de Javascript 1.2 et supérieures
keypress (onKeyPress)	Se produit lorsque l'utilisateur maintient une touche de son clavier enfoncée. Cet événement n'est supporté que par les versions de Javascript 1.2 et supérieures
keyup (onKeyUp)	Se produit lorsque l'utilisateur relâche une touche de son clavier préalablement enfoncée. Cet événement n'est supporté que par les versions de Javascript 1.2 et supérieures
Load (onLoad)	Se produit lorsque le navigateur de l'utilisateur charge la page en cours
MouseOver (onMouseOver)	Se produit lorsque l'utilisateur positionne le curseur de la souris au-dessus d'un élément
MouseOut (onMouseOut)	Se produit lorsque le curseur de la souris quitte un élément. Cet événement fait partie du Javascript 1.1.
Reset (onReset)	Se produit lorsque l'utilisateur efface les données d'un formulaire à l'aide du bouton Reset.
Resize (onResize)	Se produit lorsque l'utilisateur redimensionne la fenêtre du navigateur
Select (onSelect)	Se produit lorsque 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)
Unload (onUnload)	Se produit lorsque le navigateur de l'utilisateur quitte la page en cours

III- Association des événements aux objets

Chaque événement ne peut pas être associé à n'importe quel objet. Il est évident par exemple qu'un événement *OnChange* ne pourra pas s'appliquer à un lien hypertexte. Voici un tableau récapitulant les objets auxquels peuvent être associés chaque événement :

Événements	Objets concernés
abort	Image
blur	Button, Checkbox, FileUpload, Password, Radio, Reset, Select, Submit, Text, TextArea, window
change	FileUpload, Select, Submit, Text, TextArea
click	Button, document, Checkbox, Link, Radio, Reset, Select, Submit
dblclick	document, Link
dragdrop	window
error	Image, window
focus	Button, Checkbox, FileUpload, Password, Radio, Reset, Select, Submit, Text,

	TextArea, window
keydown	document, Image, Link, TextArea
keypress	document, Image, Link, TextArea
keyup	document, Image, Link, TextArea
load	Image, window
mousedown	Button, document, Link
mousemove	Aucun spécifiquement
mouseout	Layer, Link
mouseover	Area, Link
mouseup	Button, document, Link
move	window
reset	form
resize	window
select	text, Textarea
submit	Form
unload	window

IV- Les événements et les formulaires:

Avec Javascript, les formulaires Html prennent une toute autre dimension. N'oublions pas qu'en Javascript, on peut accéder à chaque élément d'un formulaire pour, par exemple, y aller lire ou écrire une valeur, noter un choix auquel on pourra associer un gestionnaire d'événement... Tous ces éléments renforceront grandement les capacités interactives de vos pages.

❖ **Exemple1 (Zone de texte)** : Taper le code JS suivant:

La zone de texte est l'élément d'entrée/sortie par excellence de Javascript. La syntaxe Html est <INPUT TYPE="text" NAME="nom" SIZE=x MAXLENGTH=y> pour un champ de saisie d'une seule ligne, de longueur x et de longueur maximale de y.

L'objet text possède trois propriétés :

- ❖ **name** indique le nom du contrôle par lequel on pourra accéder.
- ❖ **Defaultvalue** indique la valeur par défaut qui sera affichée dans la zone de texte.
- ❖ **Value** indique la valeur en cours de la zone de texte. Soit celle tapée par l'utilisateur ou si celui-ci n'a rien tapé, la valeur par défaut.

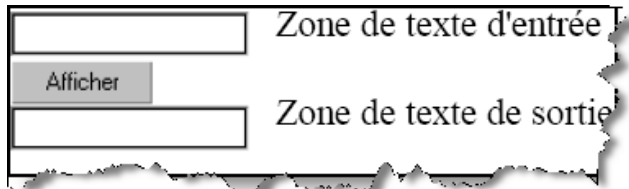
```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function controle(form1) {
var test = document.form1.input.value;
alert("Vous avez tapé : " + test);
}
</SCRIPT>
</HEAD><BODY>
<FORM NAME="form1">
<INPUT TYPE="text" NAME="input" VALUE=""><BR>
<INPUT TYPE="button" NAME="bouton" VALUE="Contrôler"
onClick="controle(form1)">
</FORM>
</BODY>
</HTML>
```



- ❖ **Constatation** : Lorsqu'on clique sur le bouton "contrôler", Javascript appelle la fonction controle() à laquelle on passe le formulaire dont le nom est form1 comme argument. Cette fonction controle() définie dans les balises <HEAD> prend sous la variable test, la valeur de la zone de texte. Pour accéder à cette valeur, on note le chemin complet de celle-ci. Soit dans le document présent, il y a l'objet formulaire appelé form1 qui contient le contrôle de texte nommé input et qui a comme propriété l'élément de valeur value. Ce qui donne document.form1.input.value.

- ❖ **Exemple2 (Ecrire une valeur dans une zone de texte)** : Taper le code JS suivant:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function afficher(form2) {
var testin =document. form2.input.value;
document.form2.output.value=testin
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="form2">
<INPUT TYPE="text" NAME="input" VALUE="">
Zone de texte d'entrée <BR>
<INPUT TYPE="button" NAME="bouton"
VALUE="Afficher"
onClick="afficher(form2)"><BR>
<INPUT TYPE="text" NAME="output" VALUE="">
Zone de texte de sortie
</FORM>
</BODY>
</HTML>
```



The screenshot shows a web form with two text input fields and a button. The top field is labeled "Zone de texte d'entrée" and the bottom field is labeled "Zone de texte de sortie". A button labeled "Afficher" is positioned between the two fields.



Constatation : Lorsqu'on clique le bouton "Afficher", Javascript appelle la fonction afficher() à laquelle on passe le formulaire dont le nom est cette fois form2 comme argument. Cette fonction afficher() définie dans les balises <HEAD> prend sous la variable testin, la valeur de la zone de texte d'entrée. A l'instruction suivante, on dit à Javascript que la valeur de la zone de texte output comprise dans le formulaire nommé form2 est celle de la variable testin. A nouveau, on a utilisé le chemin complet pour arriver à la propriété valeur de l'objet souhaité soit en Javascript **document.form2.output.value**.

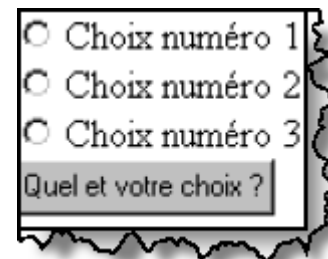
- ❖ **Exemple3 (Les boutons radio)** : Taper le code JS suivant:

Les boutons radio sont utilisés pour noter un choix, et seulement un seul, parmi un ensemble de propositions.

Propriété Description :

- ❖ **name** indique le nom du contrôle. Tous les boutons portent le même nom.
- ❖ **index** l'index ou le rang du bouton radio en commençant par 0.
- ❖ **checked** indique l'état en cours de l'élément radio
- ❖ **defaultchecked** indique l'état du bouton sélectionné par défaut.
- ❖ **value** indique la valeur de l'élément radio.

```
<HTML><HEAD><SCRIPT language="javascript">
function choixprop(form3) {
if (form3.choix[0].checked) alert("Vous avez choisi la proposition " +
form3.choix[0].value) ;
else if (form3.choix[1].checked) alert("Vous avez choisi la proposition
" + form3.choix[1].value) ;
else if (form3.choix[2].checked) alert("Vous avez choisi la proposition
" + form3.choix[2].value) ;
else alert("Vous n'avez rien choisi") ;
}</SCRIPT></HEAD>
<BODY>
<FORM NAME="form3">
<INPUT TYPE="radio" NAME="choix" VALUE="1">Choix numéro 1<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="2">Choix numéro 2<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="3">Choix numéro 3<BR>
<INPUT TYPE="button"NAME="but" VALUE="Quel et votre choix ?"
onClick="choixprop(form3)">
</FORM></BODY></HTML>
```



The screenshot shows a form with three radio buttons labeled "Choix numéro 1", "Choix numéro 2", and "Choix numéro 3". Below them is a button labeled "Quel et votre choix ?".



Constatation : Dans le formulaire nommé form3, on déclare trois boutons radio. Notez que l'on utilise le même nom pour les trois boutons. Vient ensuite un bouton qui déclenche la fonction **choixprop()**. Cette fonction teste quel bouton radio est coché. On accède aux boutons sous forme d'indice par rapport au nom des boutons radio soit choix[0], choix[1], choix[2]. On teste la propriété **checked** du bouton par **if(form3.choix[x].checked)**. Dans l'affirmative, une boîte d'alerte s'affiche. Ce message reprend la valeur attachée à chaque bouton par le chemin **form3.choix[x].value**.

❖ **Exemple4 (Les boutons case à cocher)** : Taper le code JS suivant:

Les boutons case à cocher sont utilisés pour noter un ou plusieurs choix (pour rappel avec les boutons radio un seul choix) parmi un ensemble de propositions. A part cela, sa syntaxe et son usage est tout à fait semblable aux boutons radio sauf en ce qui concerne l'attribut name.

Propriété Description

- ❖ **name** indique le nom du contrôle. Toutes les cases à cocher portent un nom différent.
- ❖ **checked** indique l'état en cours de l'élément case à cocher.
- ❖ **defaultchecked** indique l'état du bouton sélectionné par défaut.
- ❖ **value** indique la valeur de l'élément case à cocher.

```
<HTML>
<HEAD>
<script language="javascript">
function reponse(form4) {
if ( (form4.check1.checked) == true && (form4.check2.checked) ==
true && (form4.check3.checked) == false
&& (form4.check4.checked) == true)
{ alert("C'est la bonne réponse! ") }
else
{alert("Désolé, continuez à chercher.")}
}
</SCRIPT>
</HEAD>
<BODY>
Entrez votre choix :<br>
<FORM NAME="form4">
<INPUT TYPE="CHECKBOX" NAME="check1" VALUE="1">Choix
numéro 1<BR>
<INPUT TYPE="CHECKBOX" NAME="check2" VALUE="2">Choix
numéro 2<BR>
<INPUT TYPE="CHECKBOX" NAME="check3" VALUE="3">Choix
numéro 3<BR>
<INPUT TYPE="CHECKBOX" NAME="check4" VALUE="4">Choix
numéro 4<BR>
<INPUT TYPE="button"NAME="but" VALUE="Corriger"
onClick="reponse(form4)">
</FORM>
</BODY>
</HTML>
```



Constatation : Dans le formulaire nommé form4, on déclare quatre cases à cocher. Notez que l'on utilise un nom différent pour les quatre boutons. Vient ensuite un bouton qui déclenche la fonction reponse(). Cette fonction teste quelles cases à cocher sont sélectionnées. Pour avoir la bonne réponse, il faut que les cases 1, 2 et 4 soient cochées. On accède aux cases en utilisant chaque fois leur nom. On teste la propriété checked du bouton par (form4.nom_de_la_case.checked). Dans l'affirmative (&& pour et logique), une boîte d'alerte s'affiche pour la bonne réponse. Dans la négative, une autre boîte d'alerte vous invite à recommencer.

❖ **Exemple5 (Liste de sélection)** : Taper le code JS suivant:

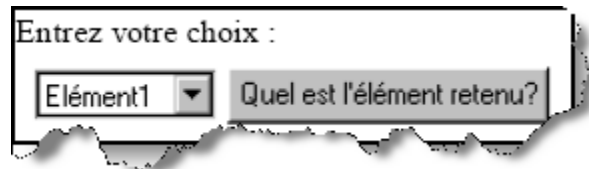
Le contrôle liste de sélection vous permet de proposer diverses options sous la forme d'une liste déroulante dans laquelle l'utilisateur peut cliquer pour faire son choix. Ce choix reste alors affiché.

La boîte de la liste est créée par la balise <SELECT> et les éléments de la liste par un ou plusieurs tags <OPTION>. La balise </SELECT> termine la liste.

Propriété Description

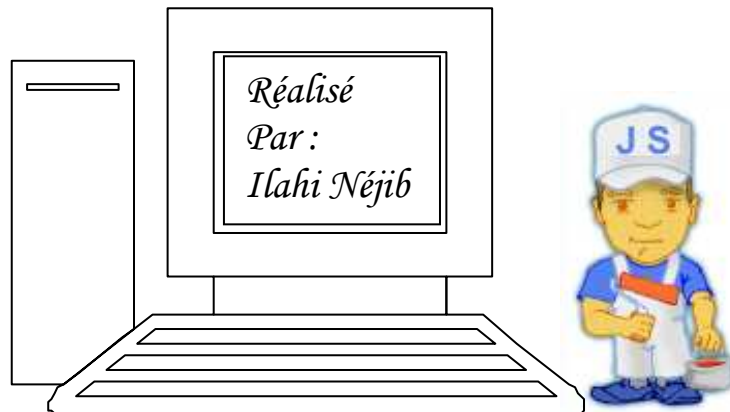
- ❖ **name** indique le nom de la liste déroulante.
- ❖ **length** indique le nombre d'éléments de la liste. S'il est indiqué dans le tag <SELECT>, tous les éléments de la liste seront affichés. Si vous ne l'indiquez pas un seul apparaîtra dans la boîte de la liste déroulante.
- ❖ **selectedIndex** : indique le rang à partir de 0 de l'élément de la liste qui a été sélectionné par l'utilisateur.
- ❖ **defaultselected** : indique l'élément de la liste sélectionné par défaut. C'est lui qui apparaît alors dans la petite boîte.

```
<HTML>
<HEAD>
<script language="javascript">
function liste(form5) {
alert("L'élément " + (form5.list.selectedIndex
+ 1)); }
</SCRIPT>
</HEAD>
<BODY>
Entrez votre choix : <FORM NAME="form5">
<SELECT NAME="list">
<OPTION VALUE="1">Elément 1</option>
<OPTION VALUE="2">Elément 2</option>
<OPTION VALUE="3">Elément 3</option>
</SELECT>
<INPUT TYPE="button"NAME="b"
VALUE="Quel est l'élément retenu?"
onClick="liste(form5)"> </FORM>
</BODY>
</HTML>
```



Constatation : Dans le formulaire nommé form5, on déclare une liste de sélection par la balise <SELECT></SELECT>. Entre ses deux balises, on déclare les différents éléments de la liste par autant de tags <OPTION>. Vient ensuite un bouton qui déclenche la fonction liste(). Cette fonction teste quelle option a été sélectionnée. Le chemin complet de l'élément sélectionné est form5.nomdelaliste.selectedIndex. Comme l'index commence à 0, il ne faut pas oublier d'ajouter 1 pour retrouver le juste rang de l'élément.

Exercice d'application page106(carnet d'adresse Email en JS)



TP TIC (J S)

Matière : Technologie de l'information et de la communication
Classe : 4 SI
Enseignant : Ilahi Néjib

Durée : 2 heures
Date : Janvier 08
A.S :2007/2008



TP N°9 (Les Objets de JS):

Objectifs :

- Savoir Manipuler les différents Objets de JS.
- Savoir utiliser les différentes méthodes des ces objets.

I- L'objet string :

String est un mot anglais qui signifie "*chaîne*", L'objet *String* est un objet qui contient un certain nombre de propriétés et de méthodes permettant la manipulation de **chaînes de caractères**.

I-1- Les propriétés de l'objet string :

L'objet *string* a une seule propriété : la propriété *length* qui permet de retourner la longueur d'une chaîne de caractères. Cette propriété est très utile car lorsque l'on traite une chaîne de caractères on aime généralement savoir à quel moment s'arrêter.

La syntaxe de la propriété *length* est la suivante:

```
x = nom_de_la_chaine.length;
```

```
x = ('chaîne de caracteres').length;
```

❖ Exemple1 : (afficher la longueur d'une chaîne)

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function long(form1) {
var test = document.form1.input.value;
alert("La longueur de la chaîne : " + test.length);
}
</SCRIPT>
</HEAD><BODY>
<FORM NAME="form1">
<INPUT TYPE="text" NAME="input" VALUE=""><BR>
<INPUT TYPE="button" NAME="bouton" VALUE="length" onClick="long(form1)">
</FORM>
</BODY>
</HTML>
```

I-2- Les méthodes de l'objet string :

1. **Chaine.charAt(position)** : Retourne le caractère situé à la position donnée en paramètre

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function controle(form1) {
var test = document.form1.input.value;
for(i=0;i<test.lenght;i++)
alert("Le caractère : "+i+" est : "+ test.charAt(i));
}
</SCRIPT>
</HEAD><BODY>
<FORM NAME="form1">
<INPUT TYPE="text" NAME="input" VALUE=""><BR>
<INPUT TYPE="button" NAME="bouton" VALUE="length" onClick="controle(form1)">
</FORM>
</BODY>
</HTML>
```

2. **indexOf(sous-chaîne,position) & lastIndexOf(sous-chaîne,position)** : Retourne respectivement la 1^{ère} et la dernière position d'une sous-chaîne (lettre ou groupe de lettres) dans une chaîne de caractère, en effectuant la recherche de gauche à droite (indexOf) et de droite à gauche(lastIndexOf).

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function controle() {
var test = "Javascript";
var v="a";

alert("La première position de gauche à droite : " + v+"est : " + test.indexOf(v,0) );
alert("La première position de droite à gauche : " + v+"est : " + test.lastIndexOf(v,9));
}
</SCRIPT>
</HEAD><BODY>
<FORM NAME="form1">
<INPUT TYPE="button" NAME="bouton" VALUE="indexOf&lastIndexOf" onClick="controle()">
</FORM>
</BODY>
</HTML>
```

3. **Chaine.substring(position1,long)** : La méthode retourne la sous chaîne (lettre ou groupe de lettres) comprise entre les positions 1 et 2 données en paramètre.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function controle() {
var test = "Javascript";
alert("Entre 1 et 4 dans : " + test+" la sous chaine : "+ test.substring(0,4));
}</SCRIPT>
</HEAD><BODY>
<FORM NAME="form1">
<INPUT TYPE="button" NAME="bouton" VALUE="Substring" onClick="controle()">
</FORM>
</BODY>
</HTML>
```


4. **Chaine.toLowerCase() & chaine.toUpperCase()** : Convertit respectivement tous les caractères d'une chaîne en minuscule et majuscules.

II- L'objet MATH :

L'objet Math est, comme vous l'imaginez, un objet qui a de nombreuses méthodes et propriétés permettant de manipuler des nombres et qui contient des fonctions mathématiques courantes. Quelque soit la méthode ou la propriété utilisée, il est indispensable de le préfixer avec *Math*.

II-1- Les propriétés de l'objet MATH :

Le tableau suivant décrit les méthodes de l'objet *Math*.

Méthode	description	Exemple
abs()	Cette méthode renvoie la valeur absolue d'un nombre, il renvoie donc le nombre s'il est positif, son opposé (positif) s'il est négatif	<ul style="list-style-type: none"> • <code>x = Math.abs(3.26);</code> <code>//donne x = 3.26</code> • <code>x = Math.abs(-3.26);</code> <code>//donne x = 3.26</code>
ceil()	Renvoie le plus petit entier supérieur ou égal à la valeur donnée en paramètre	<ul style="list-style-type: none"> • <code>x = Math.ceil(6.01);</code> <code>//donne x = 7</code> • <code>x = Math.ceil(3.99);</code> <code>//donne x = 4</code>
floor()	La méthode <i>floor()</i> retourne le plus grand entier inférieur ou égal à la valeur donnée en paramètre.	<ul style="list-style-type: none"> • <code>x = Math.floor(6.01);</code> <code>//donne x = 6</code> • <code>x = Math.floor(3.99);</code> <code>//donne x = 3</code>
round()	Arrondit à l'entier le plus proche la valeur donnée en paramètre. Si la partie décimale de la valeur entrée en paramètre vaut 0.5, la méthode <i>Math()</i> arrondi à l'entier supérieur.	<ul style="list-style-type: none"> • <code>x = Math.round(6.01);</code> <code>//donne x = 6</code> • <code>x = Math.round(3.80);</code> <code>//donne x = 4</code> • <code>x = Math.round(3.50);</code> <code>//donne x = 4</code>
max(Nombre1, Nombre2)	<i>max()</i> renvoie le plus grand des deux nombres donnés en paramètre	<ul style="list-style-type: none"> • <code>var x = Math.max(6,7.25);</code> <code>//donne x = 7.25</code> • <code>var x = Math.max(-8.21,-3.65);</code> <code>//donne x = -3.65</code> • <code>var x = Math.max(5,5);</code> <code>//donne x = 5</code>

min (Nombre1, Nombre2)	Retourne le plus petit des deux nombres donnés en paramètre	<ul style="list-style-type: none"> • x = Math.min(6,7.25); //donne x = 6 • x = Math.min(-8.21,-3.65); //donne x = -8.21 • x = Math.min(5,5); //donne x = 5
pow (Valeur1, Valeur2)	Retourne le nombre <i>Valeur1</i> à la puissance <i>Valeur2</i>	<ul style="list-style-type: none"> • x = Math.pow(3,3); //donne x = 27 • x = Math.pow(9,0.5); //(racine carrée) //donne x = 3
random ()	La méthode <i>random()</i> renvoie un nombre pseudo-aléatoire compris entre 0 et 1. La valeur est générée à partir des données de l'horloge de l'ordinateur.	<ul style="list-style-type: none"> • x = Math.random(); //donne x = 0.6489534931546957
sqrt (Valeur)	Revoit la racine carrée du nombre passé en paramètre	<ul style="list-style-type: none"> • x = Math.sqrt(9); //donne x = 3

❖ **Exemple** :(Retourner le maximum de deux nombres)

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function controle() {
a = prompt("donner un entier ");
b = prompt("donner un entier ");
x = Number(a);
y = Number(b);
alert("le max entre " + x+ " et : "+y+ " est : "+ Math.max(x,y));
}</SCRIPT>
</HEAD><BODY>
<FORM NAME="form1">
<INPUT TYPE="button" NAME="bouton" VALUE="Maximum" onClick="controle()">
</FORM>
</BODY>
</HTML>
```

❖ **Constatation** :

- **Prompt** est une méthode qui permet de retourner une chaîne de caractère.
- **Number** est une méthode qui permet de convertir une chaîne en un nombre.



III- L'objet DATE :

L'objet *Date* permet de travailler avec toutes les variables qui concernent les dates et la gestion du temps. Il s'agit d'un objet inclus de façon native dans Javascript, et que l'on peut toujours utiliser. La syntaxe pour créer un objet-date peut être une des suivantes:

Nom_de_l_objet = new Date() ;

Exemples : (1) `var now = new Date();` (2) `var myDate = new Date("month dd, yyyy hh:mm:ss");`
 (3) `var myDate = new Date("month dd, yyyy");` (4) `var myDate = new Date(yy, mm, dd, hh, mm, ss);`
 (5) `var myDate = new Date(yy, mm, dd);` (6) `var myDate = new Date(milliseconds);`

La date est stockée dans une variable sous la forme d'une chaîne qui contient le jour, le mois, l'année, l'heure, les minutes, et les secondes. Il est donc difficile d'accéder à un seul élément d'un objet *date* avec les [fonctions de manipulation de chaînes de caractères](#), étant donné que chacun des éléments peut avoir une taille variable. Heureusement, les méthodes de l'objet *Date* fournissent un moyen simple d'accéder à un seul élément, ou bien de le modifier.

Leur syntaxe est la suivante: **Objet_Date.Methode()** Les méthodes dont le nom commence par le radical *get* (mot anglais qui signifie *recupérer*) permettent de renvoyer une partie de l'objet *Date* :

Méthode	Description	Type de valeurs retournée
<code>getDate()</code>	Permet de récupérer la valeur du jour du mois	L'objet retourné est un entier (entre 1 et 31) qui correspond au jour du mois: L'objet retourné est un entier qui correspond au jour de la semaine:
<code>getDay()</code>	Permet de récupérer la valeur du jour de la semaine pour la date spécifiée	<ul style="list-style-type: none"> • 0: dimanche • 1: lundi ...
<code>getFullYear()</code>	Permet de récupérer la valeur de l'année sur 4 chiffres pour la date passée en paramètre	L'objet retourné est un entier qui correspond à l'année (XXXX) : 2007
<code>getHours()</code>	Permet de récupérer la valeur de l'heure	L'objet retourné est un entier (entre 0 et 23) qui correspond à l'objet Date.
<code>getMilliseconds()</code>	Permet de récupérer le nombre de millisecondes	L'objet retourné est un entier (entre 0 et 999) qui correspond aux millisecondes de l'objet passé en paramètre.
<code>getMinutes()</code>	Permet de récupérer la valeur des minutes	L'objet retourné est un entier (entre 0 et 59) qui correspond aux minutes de l'objet Date.
<code>getMonth()</code>	Permet de récupérer le numéro du mois	L'objet retourné est un entier (entre 0 et 11) qui correspond au mois : <ul style="list-style-type: none"> • 0: janvier • 1: février ...
<code>getSeconds()</code>	Permet de récupérer le nombre de secondes	L'objet retourné est un entier (entre 0 et 59) qui correspond aux secondes de l'objet passé en paramètre.
<code>getTime()</code>	Permet de récupérer le nombre de millisecondes depuis le 1 ^{er} janvier 1970	L'objet retourné est un entier. Cette méthode est très utile pour convertir des dates, soustraire ou ajouter deux dates, etc.
<code>getTimezoneOffset()</code>	Retourne la différence entre l'heure locale et l'heure GMT (Greenwich Mean Time)	L'objet retourné est un entier, il représente le nombre de minutes de décalage
<code>getYear()</code>	Permet de récupérer la valeur de l'année sur 2 chiffres pour l'objet Date.	L'objet retourné est un entier qui correspond à l'année (XX) :

❖ **Exemple1** :(Un script qui donne simplement l'heure.)

```
<<HTML>
<HEAD>
<SCRIPT LANGUAGE="Javascript">
function getDt(){
dt=new Date();
cal=""+ dt.getDate()+"/"+(dt.getMonth()+1) + "/20" +dt.getYear();
hrs=dt.getHours();
min=dt.getMinutes();
sec=dt.getSeconds();
tm=" "+((hrs<10)?"0":"")+hrs+":";
tm+=((min<10)?"0":"")+min+":";
tm+=((sec<10)?"0":"")+sec+" ";
document.write(cal+tm);
}
</SCRIPT>
</HEAD>
<BODY >
<SCRIPT LANGUAGE="Javascript">
getDt();
</SCRIPT>
</BODY>
</HTML>
```

❖ **Exemple2** :(Un script avec une trotteuse)

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="Javascript">
function getDt(){
dt=new Date();
hrs=dt.getHours();
min=dt.getMinutes();
sec=dt.getSeconds();
tm=" "+((hrs<10)?"0":"")+hrs+":";
tm+=((min<10)?"0":"")+min+":";
tm+=((sec<10)?"0":"")+sec+" ";
document.horloge.display.value=tm;
setTimeout("getDt()",1000);
}
</SCRIPT>
</HEAD>
<BODY onLoad="getDt()">
<FORM name="horloge">
<INPUT TYPE="text" NAME="display" SIZE=15 VALUE ="">
</FORM>
</BODY>
</HTML>
```

Constatation : Vous souhaitez peut-être que votre affichage de l'heure change toutes les secondes. Rappeler vous la minuterie setTImeOut [Un peu de tout... Avancé]. Il suffit d'ajouter au script un setTImeOut qui affiche l'heure toutes les secondes. La fonction qui affiche l'heure étant getDt(), l'instruction à ajouter est donc setTImeOut("getDt()",1000); Et le tour est joué.



TP TIC (J S)

Matière : Technologie de l'information et de la communication
Classe : 4 SI
Enseignant : Ilahi Néjib

Durée : 2 heures
Date : Janvier 08
A.S :2007/2008



TP N°10 (L' Objet Window):

Objectifs :

- Savoir l'Objet Windows.
- Savoir utiliser les différentes méthodes de l'objet Windows.

I- L'objet window :

L'objet window est l'objet par excellence dans Javascript, car il est le parent de chaque objet qui compose la page web.

Les méthodes de l'objet window

L'objet *window* possède des méthodes relatives à l'ouverture et à la fermeture des fenêtres.

Les méthodes alert(), confirm() et prompt()

Les méthodes *alert()*, *confirm()* et *prompt()* sont des méthodes qui font apparaître une boîte de dialogue, elles sont expliquées en détail dans le chapitre [Boîte de dialogue](#).

Les méthodes open(), et close()

Les méthodes *open()* et *close()* sont bien évidemment destinées à permettre l'ouverture et la fermeture de fenêtres. Toutefois la syntaxe de la méthode *open()* est longue car elle admet un nombre important de paramètres:

La méthode *open()* permet d'ouvrir une fenêtre, voici sa syntaxe:

```
window.open("URL", "nom_de_la_fenetre", "options_de_la_fenetre");
```

Si vous utilisez cette instruction dans un lien hypertexte, veillez à remplacer les guillemets doubles par des guillemets simples :



```
<A href=" javascript:window.open('URL',  
                                'nom_de_la_fenetre',  
                                'options_de_la_fenetre')">Lien vers URL</A>
```

URL désigne l'url de la page qui sera affichée dans la nouvelle fenêtre, c'est-à-dire l'emplacement physique de celle-ci.

Les options de la fenêtre sont les suivantes:

option	description
directories = <i>yes/no</i>	Affiche ou non les boutons de navigation
location = <i>yes/no</i>	Affiche ou non la barre d'adresse
menubar = <i>yes/no</i>	Affiche ou non la barre de menu (fichier, edition, ...)
resizable = <i>yes/no</i>	Définit si la taille de la fenêtre est modifiable ou non
scrollbars = <i>yes/no</i>	Affiche ou non les ascenceurs (barres de défilement)
status = <i>yes/no</i>	Affiche ou non la barre d'état
toolbar = <i>yes/no</i>	Affiche ou non la barre d'outils
width = largeur (en pixels)	Définit la largeur
height = hauteur (en pixels)	Définit la hauteur

Ainsi, il est possible d'utiliser cette méthode avec n'importe quel gestionnaire d'événement, directement dans le code à exécuter ou bien dans une fonction.



- les options doivent être saisies les unes après les autres, séparées par des virgules, sans espace
- l'ensemble des options doit être encadré par les guillemets

Chacune des fenêtres doit cependant être fermée, il faut donc se servir de la méthode *close()* qui permet de fermer une fenêtre.

La méthode *close()* requiert le nom de la fenêtre comme argument, il suffit donc de créer un bouton (image, hypertexte, ou bouton de formulaire) qui permettra de fermer cette fenêtre.

❖ **Exemple1 : (Créer les deux pages suivantes)**

Fichier **test.htm** :

```
<HTML>
<BODY>
<H1>Ceci est un test<HI>
<FORM>
<INPUT TYPE="button" value= " Continuer " onClick="window.close()">
</FORM>
</BODY>
</HTML>
```

Dans la page de départ (**depart.html**) :

```
<HTML>
<BODY>
<H1>Page de départ<H1>
<FORM name="f1">
<INPUT TYPE = "button" value="Ouvrir une nouvelle fenêtre"
onClick="open('test.htm', 'new', 'width=300,height=150,toolbar=no,location=no,
directories=no,status=no,menubar=no,scrollbars=no,resizable=no')">
</FORM></body></html>
```

❖ Exemple2 : (Ouverture par un bouton)

```
<HTML>
<SCRIPT LANGUAGE="javascript">
function new_window() {
xyz="open('test.htm', 'new', 'width=300,height=150,toolbar=no,location=no,
directories=no,status=no,menubar=no,scrollbars=no,resizable=no')">
// sans espaces ni passage à la ligne
}
</SCRIPT>
<body>
<FORM>
<INPUT TYPE ="button" value="Ouvrir une nouvelle fenêtre"onClick="new_window()">
</FORM></body></html>
```

❖ Exemple3 : (Fermeture automatique d'une fenêtre après x secondes)

```
<HTML>
<BODY onLoad='compt=setTimeout("self.close()",4000);'>
<H1>Ceci est un test</H1>
<FORM>
<INPUT TYPE="button" value=" Continuer " onClick="clearTimeout(compt);self.close();">
</FORM>
</BODY>
</HTML>
```

Dans la page de départ :

```
<HTML>
<BODY>
<H1>Page de départ<H1>
<body>
<FORM>
<INPUT TYPE ="button" value="Ouvrir une nouvelle fenêtre"
onClick="open('testc.htm', 'new', 'width=300,height=150,toolbar=no,location=no,
directories=no,status=no,menubar=no,scrollbars=no,resizable=no')">
(sans espaces ni passage à la ligne)
</FORM></body></html>
```

❖ Constatation:

- Avec ce script, sans intervention de l'utilisateur, la nouvelle fenêtre se ferme de façon automatique après 4 secondes. En cliquant sur le bouton, l'utilisateur interrompt prématurément le compteur et ferme la fenêtre. Avec ce système, on est certain que la nouvelle fenêtre sera fermée.



❖ **Exemple5 : (Ouverture en cliquant sur un lien)**

Dans la page de départ, on a :

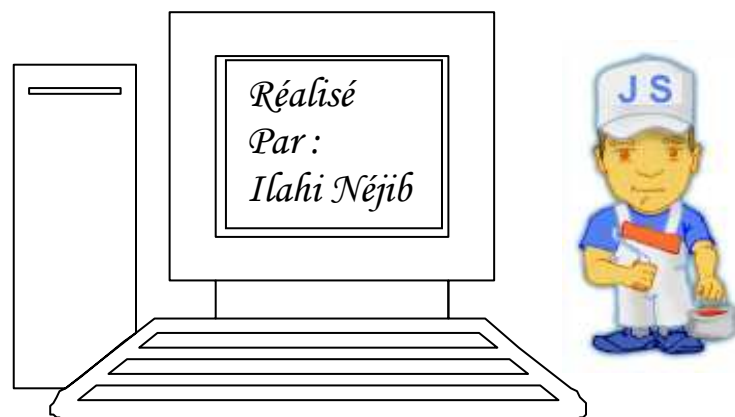
```
<HTML>
<BODY>
<A HREF="#" onClick="open('test.htm', '',
'width=300,height=150,toolbar=no,location=no,directories=no,status=no,menubar=
no,scrollbar=no,resizable=no')">lien de test</A>
</body></html>
```

❖ **Exemple6 : (On passe par l'ouverture d'une nouvelle fenêtre par l'appel d'une fonction.)**

Dans la page de départ :

```
<SCRIPT LANGUAGE="Javascript">
```

```
function opnw(){
msg=window.open("", "", "width=300,height=50,toolbar=no,location=no,directories=no,
status=no,menubar=no,scrollbars=no,resizable=no");
msg.document.write('<HTML> <BODY>' +
'<CENTER><H1>Ceci est un test<H1></CENTER>' +
'</BODY></HTML>')
// sans espaces ni passage à la ligne
}
</SCRIPT>
```



TP TIC (J S)

Matière : Technologie de l'information et de la communication
Classe : 4 SI
Enseignant : Ilahi Néjib

Durée : 2 heures
Date : Janvier 08
A.S :2007/2008



TP N°11 (L'Objet ARRAY):

Objectifs :

- Savoir l'Objet ARRAY de JS.
- Savoir utiliser les différentes méthodes de l'objet ARRAY et ses propriétés.

I- L'objet ARRAY :

L'objet Array est un objet du noyau Javascript permettant de créer et de manipuler des **tableaux**.

Voici la syntaxe à utiliser pour créer une variable tableau :

```
var myArray = new Array(element1[, element2, ...]);  
var myArray = new Array();  
var myArray = new Array(sizeInteger);  
var myArray = [element0, element1, ..., elementN];
```

Si aucun élément n'est précisé en paramètre, le tableau est vide à la création. Dans le cas contraire, il sera initialisé avec les valeurs des éléments passés en paramètres.

II- Les propriétés et les méthodes standards de l'objet ARRAY :

Le tableau suivant décrit les méthodes de l'objet *Array*.

Méthode	description
Tableau.join()	Cette méthode renvoie une chaîne de caractères contenant tous les éléments du tableau.
Tableau.pop()	Cette méthode supprime le dernier élément du tableau et retourne sa valeur.
Tableau.push(valeur1[, valeur2, ...])	Cette méthode ajoute un ou plusieurs éléments au tableau à la fin du tableau.
Tableau.reverse()	Cette méthode inverse l'ordre des éléments du tableau.
Tableau.shift()	Cette méthode supprime le premier élément du tableau et retourne sa valeur.
Tableau.lenght	Cette propriété contient le nombre d'éléments du tableau..
Tableau.sort()	Cette méthode permet de trier les éléments d'un tableau.
Tableau.unshift(valeur1[, valeur2, ...])	Cette méthode permet d'ajouter un ou plusieurs éléments au début du tableau.

❖ Exemple1 : (Tableau multidimensional)

Fichier test.htm :

```
<HTML>
<BODY>
<H2>Exploitation d'un Tableau unidimensionnel</H2>
<script language="javascript">
var mois=new Array ( "janvier" ,"fevrier" ,"mars" ,"avril" ,"mai"
,"juin","juillet","aout","septembre","octobre","novembre","decembre" ) ;
document.write("<table bgcolor='white' align='center'>");
for (i=0;i<12;i++)
{
document.write("<tr><td bgcolor='#8f8fbf'><font face='verdana'
color='white'><center>");
document.write(mois[i] );
document.write("</center></font></td></tr>");
}
document.write("</table>");
</script></body></html>
```



Résultat

❖ Exercice 9 page 116 : (Tableau multidimensional)

```
<HTML><head>
<script language="javascript">
nom = new Array(3);
nom[0] = new Array(3);
nom[1] = new Array(3);
nom[2] = new Array(3);
nom[0][0]="1200"; nom[0][1]="1250";
nom[0][2]="1300";
nom[1][0]="800"; nom[1][1]="850"; nom[1][2]="900";
nom[2][0]="500"; nom[2][1]="520"; nom[2][2]="540";
function affi() {
i = document.form.liste.selectedIndex;
j= document.form.taille.selectedIndex
document.form.txt.value=nom[i][j];
}
</script></head>
<BODY>
<H2>Exploitation d'un Tableau multidimensional</H2>
<FORM name="form" >
<SELECT NAME="liste">
<OPTION>Chemises</option>
<OPTION>Polos </option>
<OPTION>T-shirts </option>
</SELECT>
<SELECT NAME="taille">
<OPTION>T. Small </option>
<OPTION>T. Médium </option>
<OPTION>T. Large </option>
</SELECT>
<INPUT TYPE="button" VALUE="Donner le prix"
onClick="affi()">
<INPUT TYPE="TEXT" NAME="txt">
</FORM></body></html>
```

Tarif	T. Small	T. Médium	T. Large
Chemises	1200	1250	1300
Polos	800	850	900
T-shirts	500	520	540

Choix de l'article : Chemises ▼

Choix de la taille : T. Small ▼

Donner le prix

Résultat

