PRODUCTION ELECTRONIQUE AVANCEE
Partie B : Pages Web Statiques

Objectifs

- Créer des pages Web Statiques en utilisant le langage HTML ;
- Programmer et intégrer des scripts dans une page Web ;

B. Le Langage JavaScript

I. Introduction

I.1. Les limites du langage HTML :

Le langage HTML présente plusieurs limites :

- Absence de structures de contrôles algorithmiques (conditionnelles et itératives) ;
- Un langage sans aucune logique de programmation procédurale
- Absence de possibilité d'interfaçage avec les bases de données.

Pour passer ces limites deux solutions sont proposées :

- Utilisation d'un langage du côté client (JavaScript, VbScript, …) permettant d'ajouter d'autres fonctionnalités omises par HTML et peuvent être exécutés par le navigateur ;
- Utilisation des langages du côté serveur (ASP, PHP, …) : offrent la possibilité de se connecter à des bases de données et de verrouiller le code source. Le test de ces langages se fait sur un serveur d'hébergement (Apache, Netscape server, …).

I.2. Le Langage JavaScript :

C'est un langage de script incorporé (introduit, inséré) aux balises HTML permettant d'améliorer la présentation et l'interactivité des pages web.

II. Le formalisme de base du JavaScript

Activité 1 : Créer un nouveau fichier texte et l'enregistrer sous le nom « TPJS1.html » et saisir le code suivant :

```
<html>
<head>
<title>Premier code en javascript</title>
</head>
<body>
<h1><center>Premier exemple en JavaScript</center></h1>
<script language="javascript">
alert("Bienvenue");
//alert permet d'afficher une fenêtre de message.
</script>
<h1><center>Fin du premier script</center></h1>
</body>
</html>
```

Balises HTML

Script

Balises HTML

```
// Commentaire
/*Commentaire sur plusieurs lignes*/
```

III. Les objets JavaScript

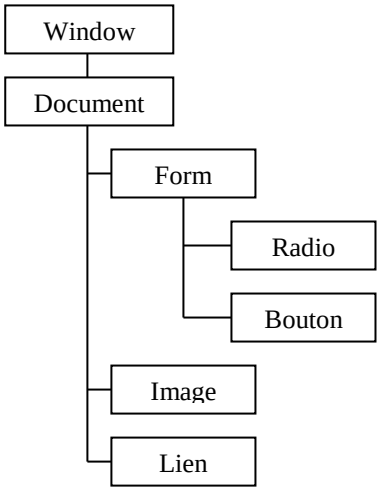
III.1. Les objets JavaScript et leurs hiérarchies

JavaScript divise une page web en objets et permet d'accéder à ces objets de tirer des informations et de les manipuler. On distingue deux catégories d'objets :

- Les objets d'interface : permettent de gérer les aspects visuels des différents contrôles graphiques : L'objet Window, document, button, radio, checkbox, etc.
- Les objets des propriétés et des fonctions prédéfinies : fournissent les différentes ressources requises pour la programmation : l'objet String, math, date, navigator, array et object.

a) La hiérarchie des objets d'interfaces

Exemple :



b) Les propriétés des objets

Pour accéder à une propriété, il faut donner le chemin complet de l'objet. En JavaScript on utilise la syntaxe : **nom_objet.nom_propriété**

Exemple : dans le cas des boutons Radio on dispose de la propriété sélection (checked= true) ou la non sélection (checked=false). on peut écrire ***if (window.document.form.radio[0].checked)***

Radio [0] : 0 désigne qu'il s'agit de la première case d'option retrouvée dans la page web.

c) Les méthodes des objets

Le langage JavaScript a prévu un ensemble de méthodes (fonctions) pour chaque objet. Par exemple pour l'objet Document on dispose de la méthode **write ()** « Ecrire dans le document ».

Activité : Créer un fichier nommé « TPJS2.html » et saisir le code suivant :

```
<html>
<head>
<title>methodes</title>
</head>
<body>
<script language="javascript">
document.write("methode write <br>");
x=2;
document.write(x);
document.write("<br>le contenu de la variable x est: "+x);
document.write("<br><b>le contenu de la bariable x est :</b>");
document.write("<font color='red'>"+x+"</font>");
</script>
</body>
</html>
```

- L'opérateur « + » joue le rôle de concaténation lorsqu'il est utilisé avec la méthode « write ».
- Il est possible de générer du code HTML lors de l'utilisation de la méthode « write ».
- Si le code HTML contient (") et pour ne pas confondre avec les guillemets de write il sera mieux de les remplacer par des apostrophes (').

III.2. Les différents emplacements du code JavaScript

Activité : Définir les différents emplacements du code JavaScript dans un document HTML.

a) Créer un fichier nommé « TPJS3.html » et saisir le code suivant :

```
<html><head><title>emp</title>
<script language="javascript">
function fermer()
{
 window.close();
}
</script>
</head>
<body>
<script language="javascript">
function message()
{
 document.write("pour fermer cette fenetre vous pouvez cliquer: ");
}
</script>
<script language="javascript">
message();
</script>
<a href ="" onclick="fermer()">ici</a>
</body></html>
```

b) Créer un fichier nommé « lib.js », déplacer les deux fonctions fermer() et msg() vers ce fichier et puis ajouter à la partie entête du fichier « TPJS3.html » le code suivant :
<SCRIPT LANGUAGE= "JavaScript" SRC="libe.js"></SCRIPT> et faire le test.

Code du fichier : TPJS3.html

```
<html><head>
<title>emp</title>
<script language="javascript" src="libe.js"></script></head>
<body>
<script language="javascript">
message();
</script>
<a href ="" onclick="fermer()">ici</a>
</body></html>
```

Code du fichier : libe.js

```
function fermer()
{
 window.close();
}
function messge()
{
 document.write("Vous pouvez fermer cette fenêtre en cliquant:");
}
```

Constatations :

- Les solutions permettant d'insérer du code JavaScript dans une page web sont :
  - Insérer des instructions JavaScript entre <script>…</script> dans <body> et </body> ;
  - Déclarer les fonctions puis les appeler dans <body> et </body> ;
  - Utiliser un des gestionnaires d'évènements rattachés aux balises html pour appeler des fonctions préalablement définies « onClik » et « onMouseOver ».
- La déclaration de fonctions JavaScript peut se faire :
  - Entre <head> et </head>
  - Entre <body> et </body>
  - Dans un fichier externe avec l'extension « .js » à inclure dans les fichiers HTML.

IV. Les variables

IV.1. La déclaration des variables

Les variables peuvent se déclarer de deux façons :

- Façon explicite : en utilisant la commande « VAR »
  Exemple : VAR Numero =1 VAR Prenom= "Yassine"
- Façon implicite : Décrire directement le nom de la variable suivi de la valeur attribuée :
  Exemple : Numero = 1

Activité1 : Créer un fichier nommé « tpjs4.html » et saisir le code suivant :

```
<Html>
<Head><title>Les variables</title>
<script language="javascript">
Var x=2;
      y=3;
function test()
 {
  x=22;
  var y=33;
 }
</script>
</head>
<body>
<script language="javascript">
document.write("la valeur de x est :"+x+" Alors que la valeur de y est: "+y);
test();
document.write("<br> la valeur de x est: "+x+" Alors que la valeur de y est: "+y);
</Script>
</Body></html>
```

IV.2. La visibilité des variables

- Les variables déclarées (de façon explicite ou implicite) au début du script, en dehors et avant toutes fonctions, sont des variables globales ;
- Dans une fonction :
  - Une variable déclarée par le mot clé Var est locale à cette fonction ;
  - Une variable déclarée sans le mot clé Var, sa portée sera globale.
- JavaScript utilise 4 types de données :
  - Des nombres (entier et réel) ;
  - Des chaînes de caractères ;
  - Des booléens : (True pour vrai et False pour faux) ;
  - Le mot null : Mot spécial qui indique l'absence d'une valeur.

Il ne faut pas déclarer le type de données d'une variable.

Activité2 : Créer un fichier nommé « tpjs4bis.html » et saisir le code suivant :

```
<html>
<body>
<script language="javascript">
b=prompt ("saisissez une valeur");
if(isNaN(b))
 {
  alert("conversion impossible");
}
  else
  {
   b=Number (b);
   b=b+1;
   document.write("la valeur de b est :"+b);
  }
</script>
</body>
</html>
```

Remarque :

- La Fonction « **isNaN** : is Not a Number » est une fonction booléenne permettant de vérifier si le contenu d'une variable donnée est numérique ou non. ;
- Alert et Prompt : sont deux méthodes de l'objet « window » permettant l'affichage et la saisie dans des boites de dialogue.
- Il existe des fonctions de conversion de type : String et Number
  - Var a = String (21.34) ; → a = "21.34"
  - Var a = Number ("10.5") ; → a = 10.5
- La fonction « eval » évalue une chaine de caractères sous forme de valeur numérique : Exemple : ch="5+10" ; x=eval (ch) ; → x=15 ;
- Il est préférable de précéder toute conversion avec la fonction « Number » par un test de validité avec la fonction « isNaN ».

V. Les opérateurs prédéfinis

V.1. Les opérateurs de calcul

Signe	Signification
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo (reste de la division)
=	Affectation

V.2. Les opérateurs de comparaison

Signe	Signification
==	Egal
<	Inférieur
<=	Inférieur ou égale
>	Supérieur
>=	Supérieur ou égal
!=	Différent

V.3. Les opérateurs associatifs

Sont des opérateurs qui réalisent un calcul dans lequel une variable intervient des deux cotés du signe (dans notre exemple x =11 et y = 5)

Signe	Nom	Signification	Exemple	Résultat
+=	Plus égal	x = x+y	x +=y	16
-=	Moins égal	x = x-y	x -=y	6
*=	Multiplié égal	x = x*y	x *=y	55
/=	Divisé égal	x = x/y	x /=y	2.2

V.4. Les opérateurs logiques

Signe	Nom	Exemple
&&	ET	(Condition1) && (Condition2)
||	OU	(Condition1) || (Condition2)
!	NON	! (Condition)

V.5. Les opérateurs d'incrémentation

Ces opérateurs vont augmenter ou diminuer la valeur d'une variable d'une unité.
Dans notre exemple x vaut initialement 3.

Signe	Signification	Exemple	Résultat
x++, ++x	Incrémentation (x++ est le même que x=x+1)	Y=x++ ; Z=++x ;	Y=3 et x=4 Y=4 et x=4
x--, --x	Décrémentation (x-- est le même que x=x-1)	Y=x-- ; Y=--x ;	Y=3 et x=2 Y=2 et x=2

V. Les Entrées/Sorties en JavaScript

- L'entrée (lecture) : est faisable avec la méthode « prompt » de l'objet « window » ou à l'aide d'objets graphiques du formulaire HTML ;
- La sortie : est possible en utilisant la méthode « write » de l'objet « document », la méthode « alert » de l'objet « window » ou à l'aide d'objets graphique du formulaire HTML.

Activité : Créer un fichier nommé « tp6js.html » et saisir le code suivant :

```
<html>
<body>
<h1>Permutation de deux valeurs</h1>
<script language="javascript">
a=Number(prompt("a=",""));
b=Number(prompt("b=",""));
a=a+b;
b=a-b;
a=a-b;
alert("après la permutation a="+a);
alert("après la permutation b="+b);
</script>
</body>
</html>
```

Syntaxe :

L'entrée : nom_variable=prompt("texte de la boite d'invite","valeur par défaut") ;
La sortie : document.write("message"+nom_variable) ;
Alert("message"+nom_variable) ;

Autre formulation

```
<html>
<head>
<title>entree-sortie</title>
</head>
<body>
<script language="javascript">
a=prompt("a=","");
if(isNaN(a))
{alert("ce n'est pas une valeur numérique");}
else
{a=Number(a);
}
b=prompt("b=","");
if(isNaN(b))
{alert("ce n'est pas une valeur numérique");}
else
{b=Number(b);
}
a=a+b;
b=a-b;
a=a-b;
alert("a="+a);
alert("b="+b);
</script>
</body>
</html>
```

VI. Les Structures de contrôle :

VI.1. Les structures conditionnelles

a) La structure IF

La Forme réduite	La Forme complète

IF (condition vraie)
{Une ou plusieurs instructions ;}
	IF (condition vraie)
	{Instructions 1 ;}
	ELSE
	{Instructions 2 ;}

Activité 1 : Déterminer le maximum de deux entiers (forme réduite)

```
<html>
<head>
<title>forme reduite</title>
</head>
<body>
<script language="javascript">
var a=Number(prompt("donner un premier entier:",""));
var b=Number(prompt("donner un deuxieme entier:",""));
max=a;
if (a<b)
{max=b;}
alert("le maximum est:"+max);
</script>
</body></html>
```

Activité 2 : Déterminer le maximum de deux entiers (forme généralisée)

```
<html>
<head>
<title>forme reduite</title>
</head>
<body>
<script language="javascript">
var a=Number(prompt("donner un premier entier:",""));
var b=Number(prompt("donner un deuxieme entier:",""));
if (a<b)
{max=b;}
else
{max=a;}
alert("le maximum est:"+max);
</script>
</body>
</html>
```

Remarques:

- Dans le cas d'une seule instruction, les accolades sont facultatives;
- Il est possible d'imbriquer des structures conditionnelles (forme généralisée) ;
- Autre syntaxe (expression) ? instruction 1 : instruction 2.

b) La structure switch

La syntaxe est la suivante :

```
Switch (expression)
{
Case v1: bloc 1;
Break;
Case v2: bloc 2;
Break ; …
Default: bloc n;
Break ;}
```

Activité : Ecrire un code javaScript permettant de lire le numéro du mois et la valeur d'une année puis de déterminer le nombre de jours de ce mois.

```
<html>
<head>
<title>nombre de jours</title>
</head>
<body>
<script language="javascript">
var m=Number(window.prompt("donner le numéro du mois",""));
var a=Number(window.prompt("donner la valeur de l'année",""));
switch(m)
{
case 1:case 3:case 5:case 7:case 8:case 10:case 12:j=31;break;
case 4:case 6:case 9:case 11:j=30;break;
case 2:if(a%4==0)j=29; else j=28;break;}
alert("le nombre de jours du mois n°"+m+" est: "+j);
</script>
</body>
</html>
```

Remarques:
- L'instructions break permet de quitter la structure switch après l'exécution du bloc convenable ;
- Lorsque le résultat de l'expression est différent de toutes les valeurs envisagées ; c'est le bloc n relatif à default qui sera exécuté ;
- La clause default (bloc n) est facultative.

VI.2. Les structures itératives

a) La structure For

Permet de répéter l'exécution d'un bloc d'instructions un certain nombre de fois connu d'avance :

```
For (initialisation ; condition ; progression)
{
 Instructions ;
}
```

- Initialisation : les instructions d'initialisations nécessaires ;
- Condition : la condition de continuité (bouclage) ;
- Progression : définit le pas du compteur (pas forcément + ou – 1)

Activité : Ecrire un code JavaScript permettant d'afficher tous les nombres parfaits compris entre 2 et 1000 ; sachant qu'un nombre N est dit parfait s'il est égal à la somme de ses diviseurs sauf lui même. Exemple pour N=6 ➔ 1+2+3=6

```
<html>
<head>
<title>boucle pour</title>
</head>
<body>
<script language="javascript">
for (n=2;n<=1000;n++)
{
 s=0;
 for(i=1;i<=n/2;i++)
  if(n%i==0) s+=i;
 if (s==n) document.write(n+"<br>");
}
</script>
</body>
</html>
```

b) La structure do…while

Permet de répéter l'exécution des instructions tant que la condition est vérifiée. L'équivalent en langage pascal est repeat…until (à la seule différence que la condition utilisée après until est une condition d'arrêt alors que la condition utilisée après while est une condition de continuité)

```
Do
{
Instructions;
} while (condition(s));
```

Activité : Ecrire un code JavaScript permettant d'afficher tous les nombres premiers compris entre deux entiers (a>1 et a<b et b<1001). Un nombre est dit premier s'il n'est divisible que par 1 et par lui-même.

```
<html>
<head>
<title>do while</title>
</head>
<body>
<script language="javascript">
do
{
a=Number(window.prompt("donner un entier: ",""));
b=Number(window.prompt("donner un autre entier:",""));
}while(a<=1||a>=b||b>=1001);
for (n=a;n<=b;n++)
{i=1;
 do
  {i=i+1;
  }while (n%i!=0 && i<=n/2);
if (i>n/2) document.writeln(n);}
</script>
</body>
</html>
```

c) La structure while

Lorsque le nombre de répétitions n'est pas connu d'avance on utilise la structure itérative while.

```
While (condition(s))
{
 Instructions ;
}
```

Activité : Ecrire un code JavaScript permettant d'afficher le PGCD de deux entiers donnés a et b tels que (a>1 et b>1) en utilisant la méthode de différences.

```
<html>
<head>
<title>boucle while</title>
</head>
<body>
<script language="javascript">
var a,b;
do
{
a=Number(prompt("donner un entier",""));
b=Number(prompt("donner un autre entier",""));
}while (a<=1||b<=1);
while (a!=b)
if (a>b)
a-=b;
else
b-=a;
alert("le pgcd est "+a);
</script></body>
</html>
```

VII. Les Fonctions en JavaScript :

En JavaScript, il existe deux types de fonctions :

- Les fonctions prédéfinies, on les appelle « **méthodes** ». Elles sont associées à un objet particulier comme par exemple la méthode **Alert ()** de l'objet **window**.
- Les fonctions déclarées par le programmeur selon les besoins de l'application.

Pour déclarer une fonction, on utilise la syntaxe suivante :

```
function nom_de_la_fonction(arguments)
{
  Instructions ;
  [return nom_variable]
}
```

Activité : modifier l'exemple précédent en utilisant une fonction qui retourne le PGCD de deux entiers.

```
<html>
<head>
<title>Les fonctions</title>
</head>
<body>
<script language="JavaScript">
function pgcd(a,b)
{
 while(a!=b)
 {
  if(a>b)
  a-=b;
  else
  b-=a;
  }
return a;
}
var n,m;
do
{
n=Number(prompt("donner un entier",""));
m=Number(prompt("donner un autre entier",""));
}while(n<=1||m<=1);
var k=pgcd(n,m);
document.write("le pgcd est: "+k);
</script>
</body>
</html>
```

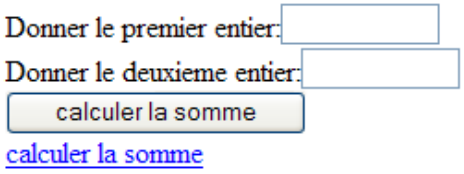
Remarques :

- La mention des arguments est facultative mais les parenthèses doivent rester ;
- Une variable déclarée dans une fonction par le mot var est locale à cette dernière, sans le mot var sa portée est globale ;
- Il est possible de définir une fonction sans l'utilisation de la clause « return » : on retrouve ainsi l'équivalent d'une procédure ;
- Il est possible de déclarer une fonction dans la partie entête de la page web <head> et </head>.

VIII. La gestion des évènements en Javascript :

Javascript offre la possibilité d'utiliser plusieurs évènements, contrairement au HTML qui se contente de l'unique clic. La programmation des évènements par l'association à chaque évènement une action. La syntaxe est la suivante :
<nom_balise OnEvenement = "fonction()">
Exemple: ici

Activité 1: Créer un fichier nommé « evenement1.html » composé de deux zones saisi et Ecrire un code Javascript permettant de calculer la somme de deux entiers saisis dans les zones de texte en cliquant sur un lien hypertexte ou un bouton.




```
<html>
<head>
<title>onclick</title>
<script language="JavaScript">
function somme()
{
var a=window.document.f1.e1.value;
var b=window.document.f1.e2.value;
a=Number(a);
b=Number(b);
s=a+b;
alert("la somme est: "+s);
}
</script>
</head>
<body>
<form name="f1">
Donner le premier entier:<input type="text" name="e1" size="10"><br>
Donner le deuxieme entier:<input type="text" name="e2" size="10"><br>
<input type="button" value="calculer la somme" name="b1" Onclick="somme()"><br>
<a href="#" Onclick="somme()">calculer la somme</a>
</form>
</body>
</html>
```

- **OnClick** : peut être utilisé avec les boutons et les liens hypertextes. Il est possible de programmer l'évènement OnClick avec les objets de type case à cocher, l'option radio et la zone de liste.

Activité 2: Créer un fichier nommé « evenement2.html » composé de deux zones de saisie, et offrant les fonctionnalités suivantes :

- Lorsque le curseur se trouve dans la première zone le message « Saisissez votre nom et prénom » est affiché dans la barre d'état ;
- Lorsque le curseur quitte la première zone : le contenu de la barre d'état se transforme en *** et le texte saisi est mis en majuscule ;
- Au fur et à mesure que le texte est saisi dans la première zone, la deuxième zone indique le nombre de caractères tapés.

```
<html>
<head>
<title>evenements</title>
<script language ="JavaScript">
function majuscule()
{
np=window.document.f1.nom.value;
npmaj=np.toUpperCase();
window.document.f1.nom.value=npmaj;
}
function affiche(msg)
{
 window.status=msg;
}
function longueur()
{
l=window.document.f1.nom.value;
window.document.f1.nbc.value=l.length;
}
</script>
</head>
<body>
<form name="f1">
NOM ET PRENOM : <input type="text" name="nom" OnFocus="affiche('saisissez votre nom et prénom')" OnChange="majuscule()" OnBlur="affiche('***')" OnKeyUp ="longueur()">
Nombre de cacarctères tapés : <input type="text" name="nbc" size="3" value="0">
</form>
</body>
</html>
```

Les évènements :

- **OnFocus** : survient lorsqu'un champ de saisi est prêt à recevoir ce que l'utilisateur à l'intention de taper au clavier (c à d quand la zone a le focus) ;
- **OnBlur : a lieu lorsqu'un champ de formulaire perd le focus (quand l'utilisateur termine la saisie et clique en dehors du champ pour passer à un autre) ;**
- **OnChange** : ressemble à l'évènement OnBlur avec une différence (la zone de texte à perdu le focus et son contenu est modifié par l'utilisateur) ;
- **OnKeyUp** : est déclanché au cours de la saisi d'une zone de texte lorsque l'utilisateur relâche une touche.

Les méthodes :

- **Length :** permet de retourner la longueur d'une chaîne ou 0 si elle est vide ;
- **toUpperCase() :** transforme une chaîne de caractères en majuscule ;
- **Status :** Valeur du texte affiché dans la barre d'état de la fenêtre.

Activité 3 : Créer un fichier nommé « evenement2.html » composé d'une image (e1.jpg) et un objet texte « Exemple3 » avec les fonctionnalités suivantes :

- Au chargement de la page un message de bienvenue est affiché, de même un message d'au revoir est affiché lorsqu'on la quitte ;
- Deux messages sont affichés lorsqu'on survole et en quitte le lien hypertexte (exemple3) ;
- Lorsque l'image e1.jpg est survolée, elle remplacée par une autre image e2.jpg.

```
 <html>
<head>
<title>evenement3</title>
<script language="javascript">
function bienvenue()
{
alert("soyez la bienvenue");
}
function aurevoir()
{
alert("merci au revoir");
}
function activer()
{
document.images["img1"].src="e2.jpg";
}
function desactiver()
{
document.images["img1"].src="e1.jpg";
}
</script>
</head>
<body OnLoad="bienvenue()" OnUnload="aurevoir()">
<a href="#" OnMouseOver="activer()" OnMouseOut="desactiver()"><img src="e1.jpg"
name="img1" width="100" height="100"></a><br>
<a href="#" OnMouseOver="alert('bonne chance')" OnMouseOut="alert('bon travail')">Exemple
3</a>
</body>
</html>
```

- ❖ OnLoad : survient lorsque la page a fini de se charger ;
- ❖ OnUnload : survient lorsque l'utilisateur quitte la page ;

OnLoad et OnUnload sont utilisés sous forme d'attributs de la balise <BODY> Ou <FRAMESET>

- ❖ OnMouseOver : se produit lorsque le pointeur de la souris passe au dessus d'un lien ou d'une image sans cliquer ;
- ❖ OnMouseOut : se produit lorsque le pointeur quitte la zone sensible (lien ou image). Généralement OnMouseOut est associé à OnMouseOver.

IX. Les formulaires en Javascript :

IX.1. Le contrôle zone de texte

Activité : Créer un nouveau document html comportant un script qui permet de lire un entier dans une zone de saisie puis d'afficher son carré dans une autre zone.

```
<html><head><title>carre</title>
<script language="JavaScript">
function calcul()
{
var n=document.f1.v1.value;
n=Number(n);
m=n*n;
document.f1.v2.value=m;
}
</script>
```

```
</head>
<body>
<form name="f1">
Donner un entier:<input type="text" name="v1" size="10" value="">
<input type="button" name="b1" value="calculez le carré" OnClick="calcul()">
Le carré est:<input type="text" name="v2" size="10" value="">
</form></body></html>
```

Remarques :

- Pour affecter le contenu d'une zone de saisie à une variable :
Nom_variable=window.document.nom_formulaire.nom_zone_de_saisie.value ;
- Pour modifier la valeur d'une zone de saisie :
Window.document.nom_formulaire.nom_zone_de_saisie.value=expression ;

IX.2. Les boutons radio

Activité : Créer un fichier html qui comporte un script Javascript permettant de donner en entrée le signe du discriminant d'une équation de second degré pour afficher le nombre de solutions possibles.

```
<html><head>
<title>radio</title>
<script language="JavaScript">
function choisir()
{
if(document.f1.choix[0].checked==true)
alert("il ya"+document.f1.choix[0].value+" solution");
if(document.f1.choix[1].checked==true)
alert("il ya"+document.f1.choix[1].value+" solution");
if(document.f1.choix[2].checked==true)
alert("il ya"+document.f1.choix[2].value+" solutions");
}
</script></head>
<body>
<center><h3>Signe du discriminant d'une  equation  de second degres</h3></center>
<form name="f1">
<input type="radio" name="choix" value="0">Strictement negatif<br>
<input type="radio" name="choix" value="1">Nul<br>
<input type="radio" name="choix" value="2">Strictement positif<br>
<input type="button" name="b1" value="nombre de solutions" OnClick="choisir()">
</form></body>
</html>
```

Remarques :

- Pour vérifier l'état d'une case d'option (cochée ou pas), on utilise la propriété **« checked »**qu'on peut affecter à une variable booléenne :
Nom_variable=document.nom_formulaire.nom_case[indice].checked
- On peut récupérer la valeur d'un bouton radio à l'aide de la propriété « value »
Nom_variable=document.nom_formulaire.nom_case[indice].value
- On peut récupérer le nombre d'option dans un groupe de cases d'options à l'aide de la propriété « length » : Nom_variable=document.nom_formulaire.nom_case.length
- Les boutons radio sont utilisés pour sélectionner un seul choix, parmi un ensemble de propositions ;
- L'indice des boutons radio commence à partir de la valeur 0.

IX.3. Les boutons case à cocher (checkbox)

Activité : Créer un fichier html qui comporte un script Javascript permettant à un utilisateur de sélectionner les nombres premiers parmi la liste des nombres suivantes : 11 ; 24 ; 37 ; 49.

```
<html><head>
<title>Case a cocher</title>
<script language="JavaScript">
function reponse()
{
if((document.f1.ch1.checked==true)&&(document.f1.ch2.checked==false)
&&(document.f1.ch3.checked==true)&&(document.f1.ch4.checked==true))
alert("c'est la bonne réponse");
else
alert("vérifiez votre choix");
}
</script>
</head>
<body>
<center><h3>Quels sont les nombres premiers parmi ces cinq entiers?</h3></center>
<form name="f1">
<input type="checkbox" name="ch1" value="1">11<br>
<input type="checkbox" name="ch2" value="2">24<br>
<input type="checkbox" name="ch3" value="3">37<br>
<input type="checkbox" name="ch4" value="4">49<br>
<input type="button" name="b1" value="Reponse" OnClick="reponse()">
</form>
</body>
</html>
```

Remarques :

- Les cases à cocher sont utilisées pour sélectionner un ou plusieurs choix ;
- Pour vérifier si une case est cochée ou non on utilise la syntaxe suivante :
  Nom_variable=document.nom_formulaire.nom_case.checked
- Pour récupérer la valeur d'une case à cocher on utilise la syntaxe suivante :
  Nom_variable=document.nom_formulaire.nom_case.value

IX.4. Liste de sélection

Activité :

```
<html>
<head>
<title>liste</title>
<script language="JavaScript">
function ajout()
{
np=document.f1.znp.value;
adr=document.f1.zadr.value;
var taille=document.f1.carnet.options.length;
var nouveau=true;
if(taille!=0)
{
for (i=0;i<taille;i++)
{
elt=document.f1.carnet.options[i].text;
```

```
 if(elt==np)
  {
  nouveau=false;
  alert("contact deja existant");
  break;
  }
  }
  }
  if(nouveau)
  {
  nelt=new Option(np,adr);
document.f1.carnet.options[taille]=nelt;
}}
function supp()
{
sel=document.f1.carnet.options.selectedIndex;
if(sel==-1)
alert("veuillez sélectionner un élément SVP");
else
{
document.f1.carnet.options[sel]=null;}
}
function adresse()
{
sel=document.f1.carnet.options.selectedIndex;
if(sel==-1)
alert("veuillez sélectionner un élément SVP");
else
{
np=document.f1.carnet.options[sel].text;
adr=document.f1.carnet.options[sel].value;
alert("l'adresse mail de:"+np+" est: "+adr);
}
}
</script>
</head>
<body>
<p align="center"><font color="red" size="6">CARNET D'ADRESSES</font></p>
<form name="f1">
Nom et Prénom: <input type="text" name="znp" size="30"><br><br>
Adresse  E-mail:<input type="text" name="zadr" size="30"><br><br>
<table>
<tr>
<td><input type="button" name="b1" value="Ajouter" OnClick="ajout()"></td>
<td><input type="button" name="b2" value="Supprimer" OnClick="supp()"></td>
<td><input type="button" name="b3" value="Retrouver Adresse E-mail"
OnClick="adresse()"></td>
</tr>
</table>
<select name="carnet" size="5"></select>
</form></body>
</html>
```

Remarques :

- La gestion des listes déroulantes se fait à travers l'objet **« options »** **document.nom_formulaire.nom_liste.options**
- Les propriétés de l'objet options sont :
  - **Length** : retourne le nombre d'éléments d'une liste déroulante ;
  - **selectedIndex** : retourne l'indice de l'élément sélectionné dans une liste déroulante à sélection unique. (L'indice d'une liste déroulante commence par 0)
- Pour une liste déroulante à sélection multiple, **selectedIndex** retourne l'indice du premier élément sélectionné pour cela on utilise les propriétés des éléments de l'objet **« options »** :
  - **selected** : renvoie **true** si l'option est sélectionnée, **false** sinon ;
  - **text** : renvoie le texte de l'élément sélectionné ;
  - **value** : renvoie la valeur de l'élément sélectionné.
- L'ajout d'une nouvelle option :
  nom_variable=new Otion(texte, valeur) ;
  document.nom_formulaire.nom_liste.options[taille]=nom_variable ;
  La taille correspond à la position de l'élément à ajouter dans la liste
- La suppression d'une option :
  Document.nom_formulaire.nom_liste.options[i]=null ;
  i correspont à la position de l'élément à supprimer.

X. Application :

Créer un fichier HTML qui permet de remplir un formulaire par le nom, le prénom, l'adresse Email, l'age et les deux boutons de confirmation et d'annulation. Le contrôle de saisie est obligatoire :

- Le nom et le prénom doivent être non vides ;
- L'adresse Email doit être non vide et comportant le caractère @ ;
- L'age doit être un nombre >0.

La validation doit appeler le fichier « sauvegarde.html » (existant dans le même répertoire du formulaire) qui affiche le message « merci pour votre visite ».

```
<html>
<head>
<title>saisie formulaire</title>
<script language="javascript">
function verification()
{
if(document.f1.nom.value=="")
{
alert("veuillez entrer votre nom SVP");
return false;
}
if(document.f1.pre.value=="")
{
alert("veuillez entrer votre prénom SVP");
return false;
}
if(document.f1.mail.value=="")
{
alert("veuillez entrer votre adresse Email SVP");
return false;
}
if(document.f1.mail.value.indexOf("@")==-1)
{
```

```
alert("ce n'est pas une adresse électronique");
document.f1.mail.value="";
return false;
}
n=document.f1.age.value;
if(isNaN(n)||n=="")
{
alert("cette information n'est pas un nombre");
return false
}
}
</script>
</head>
<body>
<center><h1>Remplissage du formulaire</h1></center>
<form name="f1" action="sauvegarde.html" method="post" OnSubmit="return verification()">
Nom:<input type="text" name="nom" size="40"><br>
Prénom:<input type="text" name="pre" size="40"><br>
Email:<input type="text" name="mail" size="40"><br>
Age:<input type="text" name="age" size="3"><br>
<center><input type ="submit" value="valider"><input type="reset" value="annuler"></center>
</form>
</body>
</html>
```

Remarque : La méthode **indexOf()** : permet de rechercher (de gauche à droite) la première position d'une sous chaîne dans une chaîne de caractères donnée si elle existe ou -1 dans le cas contraire.
P=ch.indexOf(sch,pos) ;
La position pos permet de déterminer la position du caractère à partir duquel la recherche est effectuée.