Javascript

**1.**   **Introduction**

Le langage JavaScript a été créé par la société Netscape pour la version 2.0 de son navigateur Web. Comme PHP il s’agit d’un langage qui permet de créer des programmes embarqués dans des pages Web mais à l’inverse de PHP les programmes sont ici en général exécutés par le navigateur.

On utilise beaucoup JavaScript pour créer des pages dynamiques comportant des menus, des fenêtres filles (les fameux pop-ups tant décriés), des permutations d’images lorsque le pointeur de la souris passe au dessus (on appelle cela des « roll-over ») et pour vérifier des formulaires avant leur émission.

Il existe de nombreuses versions de JavaScript (appelé également JScript par Microsfot) qui ne sont malheureusement pas compatibles entre elles. Il conviendra donc pour une utilisation professionnelle de tester les pages avec plusieurs navigateurs comme Netscape 4, Mozilla et Internet Explorer par exemple.

Le langage a néanmoins été standardisé en 1997 par l'European Computer Manufacturers Association (ECMA) sous le nom d’ECMAScript (ECMA-262). Cette version correspond à la version 1.4 de JavaScript et à la version 4.0 de Jscript.

**2.**   **Inclusion dans des pages HTML**

Pour intégrer du JavaScript dans du code HTML il faut utiliser la balise SCRIPT :

<SCRIPT language=JavaScript>

< !--

    code JavaScript...

-->

</SCRIPT>

Le code compris entre <SCRIPT> et </SCRIPT> n'est pas affiché, il est interprété par le navigateur. Comme il existe encore des navigateurs qui ignorent les scripts, il est recommandé de mettre le contenu du script sous forme de commentaire HTML :

<SCRIPT language=JavaScript>

<!--

  code JavaScript

//-->

</SCRIPT>

Ainsi le script sera simplement ignoré au lieu de placer n’importe quoi sur la page.

***Position du code dans un fichier HTML***

* dans l'en-tête : pour définir des fonctions dès le chargement ;
* entre l'en-tête et le corps ;
* dans le corps : pour gérer les événements.

***Insertion de code dans une URL d'une balise HTML***

Le code se met sous la forme :

javascript:code

Exemple : pour exécuter du code JavaScript il est possible d'utiliser :

<A href="JavaScript:code">texte</A>

Quand on clique sur le texte le code s'exécute (le plus souvent c'est un appel de fonction).

NB : dans href, si on écrit "JavaScript:void(0)" cela empêche le navigateur de faire un chargement quand on clique sur le lien.

***En liant depuis la page HTML un fichier externe, dans lequel sont placées les*** ***instructions JavaScript :***

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

***Code JavaScript pour les événements dans les balises***

On peut insérer du code JavaScript pour indiquer les actions à effectuer en réaction à un événement provoqué par l'utilisateur (souvent ce code inséré se contente d'appeler une fonction déclarée plus haut dans le document au sein d'un élément SCRIPT). Pour réagir à un événement provoqué par le client :

<BALISE onEvenement="code">

|  |  |  |
| --- | --- | --- |
| Attributs d'événements existants | | |
| **nom** | **Événement** | **s'applique à** |
| onAbort | interruption de chargement | image |
| onBlur | perte du focus | champs de saisie des formulaires ou à une fenêtre |
| onFocus | attribution du focus |
| onClick | clic sur un objet | button, checkbox, radio, reset, submit, lien retourne *false* pour annuler l'action |
| onChange | changement de champ de saisie | fileupload, password, select, text, textarea |
| onDblClick | double clic | lien, image, bouton(JS1.2) |
| onError | erreur JS ou de chargement d'une image | image, fenêtre |
| onKeyDown | touche enfoncée | document, image, lien, text retourne *false* pour annuler |
| onKeyPress | l'utilisateur a appuyé sur une touche |
| onKeyUp | touche relâchée |
| onLoad | Chargement | image, window |
| onMouseDown | un bouton de la souris est enfoncé | document, lien, image, button retourne *false* pour annuler |
| onMouseUp | le bouton de la souris est relâché |
| onMouseMove | déplacement de la souris |  |
| onMouseOut | la souris vient de quitter une zone | lien, image, layer retourne *true* pour empêcher l'affichage de l'URL |
| onMouseOver | la souris entre sur une zone |
| onReset | annulation des données saisies | formulaire ; retourne *false*pour annuler, ex : <FORM action="fich.php3" onSubmit="return Verif()"> |
| onSubmit | soumission du formulaire |
| onSelect | sélection d'un champ de texte | champs de saisie |
| onUnload | déchargement d'un document | image, fenêtre |

**3.**   **Syntaxe du langage**

***3.1 Opérateurs et expressions***

Les types de données standards en JavaScript sont :

      number (entiers et réels représentés par des valeurs à virgule flottante avec éventuellement une partie exposant : 3.4e12)

      string (chaînes de caractères)

      boolean (booléens – valeurs true ou false)

      Array (tableaux)

      object (objets cf. la programmation par objet)

La déclaration des variables n’est pas obligatoire (bien que fortement conseillée).

Elle est réalisée de la façon suivante :

            var i = 10 ;

            var j = 3.4e12;

            var s = "chaine";

**3.1.1. Opérateurs arithmétiques**

Quelques exemples valent souvent mieux qu’un long discours...

      a + b    Addition            Somme de a et b.

      a - b    Soustraction      Reste de la différence de b et a.

      a \* b    Multiplication     Produit de a par b.

      a / b    Division            Dividende de a par b.

      a % b    Modulo             Reste de la division entière de a par b.

**3.1.2. Opérateurs sur les chaînes**

Il n’existe qu’un seul opérateur sur les chaînes, c’est l’opérateur de concaténation “**+**”.

a = "Hello ";

b = a + "World!"; // b contient "Hello World!"

Il est à noter que vous pouvez spécifier une chaîne en l’encadrant entre des simples ou des doubles quotes (guillemets simples ou doubles).

**3.1.3. Opérateurs binaires**

Il s’agit des opérateurs binaires dits “bit à bit”, et non pas des opérateurs logiques booléens.

      a & b                Et         Les bits qui sont à 1 dans a ET dans b sont mis à 1.

      a | b                Ou        Les bits qui sont à 1 dans a OU dans b sont mis à 1.

      ~ a                   Non      Les bits sont inversés.

**3.1.4. Opérateurs Logiques**

      ! a                   Non      Résultat vrai si a est faux, et réciproquement

      a && b             Et         Résultat vrai si a ET b sont vrais

      a || b              Ou        Résultat vrai si a OU b est vrai (ou les deux)

**3.1.5. Opérateurs de Comparaison**

Encore du classique. Sans surprise pour les habitués de la syntaxe C.

      a == b   Egal                  Résultat vrai si a est égal à b

      a != b   Différent          Résultat vrai si a est différent de b

      a < b    Inférieur           Résultat vrai si a est strictement inférieur à b

      a > b    Supérieur          Résultat vrai si a est strictement supérieur à b

      a <= b   Inf ou égal        Résultat vrai si a est inférieur ou égal à b

      a >= b   Sup ou égal       Résultat vrai si a est supérieur ou égal à b

***3.2. structures de contrôle***

Les structures de contrôle **if**, **If…else**, ainsi que les instructions de boucle **do…while**, **for**, **break**, **continue** et **switch** sont disponibles en JavaScript et sont identiques au C (ici aussi il ne faut pas oublier les break lors des switch !).

Exemple de for :

for(i=0 ; i<5; i++)

{

    document.writeln(i);

}

***3.4. fonctions***

En JavaScript on peut définir des fonctions un peu comme en C mais on ne précisera pas le type des arguments ou de la valeur retournée. Les passages de paramètres se font par valeur (les modifications réalisées dans la fonction ne sont pas effective en dehors de celle-ci sauf pour les objets et les tableaux – cf. plus loin)

function nom\_fonction(arg1,...,argN){

instruction1;  
...  
instructionM;  
return val;  // optionnel

}

Appel :

variable = nom\_fonction(arg1, ..., argN);

***3.5 Javascript et la programmation orientée objet***

Le langage JavaScript est un langage orienté objet. Ce concept sera présenté en détail plus tard dans votre cursus. Néanmoins nous le présenterons ici très simplement.

Les objets peuvent être vus simplement comme une collection de variables (appelées ici attributs ou propriétés) et de fonctions (appelées ici méthodes) liés entre eux par un nom qui s’utilise comme une variable et qu’on appelle une référence.

Pour bien comprendre le concept, prenons un exemple : la fenêtre du navigateur qui exécute le script. Cet objet informatique est défini par une variable référence nommée **window**.

Pour accéder aux propriétés de cet objet on écrira (ici pour connaître la hauteur de la fenêtre) :

            hauteur = window.innerHeight;

De même pour utiliser une méthode de l’objet on écrira (ici pour redimensionner la fenêtre ) :

            window.resizeTo(500, 500);

On peut aussi affecter des valeurs aux propriétés (ainsi pour redimensionner la fenêtre on peut aussi écrire) :

window.innerHeight = 500;

window.innerWidth = 500;

Enfin, on peut créer de nouveaux objets en utilisant le mot clé **new**. Par exemple pour créer une image on écrira :

var mon\_image = new Image();

Image() est une méthode un peu particulière qu’on appelle un constructeur. Les constructeurs peuvent avoir des paramètres. Enfin, il peut y avoir plusieurs paramètres pour un même type d’objet.

Pour éviter de réécrire le nom d’un objet on peut utiliser le mot clé **with**. Exemple :

            with (window) {

innerHeight = 500;

innerWidth = 500;

            }

***3.6 Tableaux***

Les tableaux en JavaScript sont en fait des objets un peu particuliers.

On peut les créer de deux façons différentes :

En utilisant les constructeurs Array :

var mon\_tableau = new Array();

var mon\_tableau = new Array(taille);

var mon\_tableau = new Array(el1, el2, ..., elN);

En utilisant une construction spécifique (depuis JavaScript 1.2) :

var mon\_tableau = [el1, el2, ..., elN]

Un tableau peut comme en PHP contenir des éléments de types différents :

var tab = new Array(3, 'chaine', 4.5);  
var tab = [3, 'chaine', 4.5];

**3.6.1 Propriétés**

mon\_tableau.length -> taille du tableau

i = tab.length;

**3.6.2 Accès aux éléments**

mon\_tableau[0]  
mon\_tableau[1]  
...  
mon\_tableau[N-1]

NB : pour le parcours du tableau on peut utiliser une boucle for un peu spéciale :

for (i in mon\_tableau) {

            document.writeln(mon\_tableau[i]);

}

**3.6.3 Méthodes**

|  |  |
| --- | --- |
| join(separateur) | concatène les cases du tableau avec le separateur (JavaScript 1.1, ECMA) |
| pop() | dépile et retourne le dernier élément du tableau (Navigator 4) |
| push(val,...) | ajoute des éléments au tableau (Navigator 4) |
| reverse() | renverse le tableau (JavaScript 1.1, ECMA) |
| shift() | supprime et retourne le 1er élément (Navigator 4) |
| slice(deb, fin) | retourne une portion du tableau (JavaScript 1.2) |
| sort() | tri des éléments (JavaScript 1.1, ECMA) |
| splice(deb, nbel, val, ...) | retire des éléments du tableau (Navigator 4) |
| toString() | conversion du tableau en chaîne (JavaScript 1.1, ECMA) |
| unshift(val,...) | insère des éléments (Navigator 4) |

***3.7 Interaction avec le navigateur Web***

L’interaction avec le navigateur passe par l’objet window et ses sous-objets. Il existe un objet window pour chaque zone du navigateur (fenêtre ou frame).

**3.7.1 Propriétés de window**

|  |  |
| --- | --- |
| closed | pour savoir si la fenêtre a été fermée ou non |
| defaultStatus | texte à afficher par défaut dans la barre d'état en bas |
| document | accès à toutes les informations de la zone |
| frames[] | tableau d'objets pour les frames de la zone |
| window.frames.length, window.frames[num] |
| history | pour naviguer |
| length | nombre de cadres dans la fenêtre |
| location | informations sur l'URL de la zone |
| name | nom de la zone |
| navigator | informations sur le navigateur utilisé |
| opener | fenêtre à partir de laquelle a été créée la zone |
| parent | document contenant <FRAME> associé au frame |
| screen | propriétés de l’écran qui affiche le navigateur |
| status | texte à afficher dans la barre d'état |

**Document**

Note : on peut utiliser directement document au lieu de window.document.

Exemple :

            document.writeln("Hello World !") ;

|  |  |
| --- | --- |
| Propriétés de window.document | |
| bgColor | couleur de fond |
| fgColor | couleur du texte par défaut |
| alinkColor | couleur des liens activés |
| linkColor | couleur des liens non visités |
| vlinkColor | couleur des liens visités |
| anchors[num] | cible d'un lien hypertexte |
| window.document.anchors.length = nombre d'ancres  window.document.anchors[i].name = nom de l'ancre |
| cookie | cookies du document |
| forms[num] | formulaires du document |
| images[num] | images du document |
| links[] | tableau pour les liens du document |
| window.document.links.length attributs : hash, host, hostname, href, pathname, port, protocol, search, target (c.f. propriétés de location) |
| lastModified | date de modification du document |
| title | titre du document |
| URL | URL du document |

Méthodes de document

* write(paramètre1, ..., paramètreN) : insertion dynamique dans le document.
* writeln(paramètre1, ..., paramètreN) : idem avec retour à la ligne
* open() : vide le document et en crée un nouveau
* close() : affichage de tout ce qui a été écrit avec write() ou writeln()

**History**

Propriétés de history

* current : URL du document courant
* length : nombre d'éléments dans l'historique
* next : URL suivante dans l'historique
* previous : URL précédente dans l'historique

Méthodes de history

* back() pour simuler l’appui sur le bouton back
* go(+/-n) effectuer un retour ou une avancée dans l'historique de +/- n pages
* forward() pour simuler l’appui sur le bouton forward

Exemples :

<INPUT type="button" value="back" onClick="window.history.back()">

<INPUT type="button" value="back2" onClick="window.history.go(-2)">

<INPUT type="button" value="forward" onClick="window.history.forward()">

**Location**

Propriétés de location

* href : totalité de l'URL
* hash : partie ancre de l'URL (après #)
* host : nom du serveur et port
* hostname : adresse IP ou nom
* pathname : chaîne après le nom du script et avant '?'
* port : numéro de port
* protocol : nom du protocole (http)
* search partie requête (après '?')
* Méthodes de location
* reload() : pour recharger le document
* replace(URL) : pour charger le document situé à l'adresse URL

 <INPUT type="button" value="recharger" onClick="window.location.reload()">

<INPUT type="button" value="remplacer" onClick="window.location.replace('../Navigator.html')">

**Screen**

|  |  |  |
| --- | --- | --- |
| propriétés de l'objet Screen | | |
| height | hauteur de l'écran en pixels |  |
| width | largeur de l'écran en pixels |  |
| availHeight | hauteur moins les barres de menu ou de tâches |  |
| availWidth | largeur disponible |  |
| colorDepth | nombre de bits pour coder un pixel dans le navigateur |  |

**Navigator**

|  |  |
| --- | --- |
| propriétés de l'objet Navigator | |
| appName | nom d'application du navigateur (Netscape, Internet Explorer) |
| appVersion | plate-forme et version du navigateur (4.5 [fr] (Macintosh; I; PPC)) parseFloat(navigator.appVersion) ->4.5 |
| language | langage du navigateur (fr, en) sous Navigator 4 |
| platform | système d'exploitation (MacPPC) |

Pour accèder au nom d'application du navigateur il suffit donc d'écrire : navigator.appName.

**3.7.2 Méthodes de window**

|  |  |
| --- | --- |
| open(arguments) | ouvrir une nouvelle fenêtre |
| close() | fermer une fenêtre |
| alert(texte) | ouvre une fenêtre d'alerte contenant le texte |
| prompt(txt, val) | ouvre une fenêtre avec un message txt et un champ de texte contenant au départ la valeur val, retourne le texte tapé par l’utilisateur à la place de val |
| confirm(texte) | fenêtre de confirmation qui retourne un booléen |
| back() | bouton back du navigateur |
| find() | recherche dans le document |
| forward() | bouton forward du navigateur |
| home() | bouton accueil du navigateur |
| stop() | bouton stop du navigateur |
| moveBy(x,y) | déplace la fenêtre (zone, couche) de x pixels horizontaux et y verticaux |
| moveTo(x,y) | déplace le coin gauche en (x,y) |
| resizeBy(l,h) | pour retrécir ou agrandir la fenêtre |
| resizeTo(l,h) | largeur et hauteur de la fenêtre à agrandir ou réduire |
| scrollBy(x,y) | déplacement horizontal et vertical du document dans la fenêtre (ascenseurs) |
| scrollTo(x,y) | déplacement du coin x,y du document en haut de la fenêtre |

La méthode **open**(URL, nom, options) prend comme paramètres :

         URL : URL du document à afficher dans la fenêtre (optionnel)

         nom : nom de la fenêtre (target pour les autres documents HTML)

         option :

o        height : hauteur initiale

o        width : largeur initiale

o        left : distance bord gauche fenêtre - bord gauche écran

o        location = yes / no : barre de localisation

o        menubar = yes / no : barre de menus

o        resizable = yes / no : autorisation de redimensionnement

o        scrollbars = yes / no : barres de défilement

o        status = yes / no : barre de statut

o        toolbars = yes / no : barre d'outils

Pour plus d’informations vous pouvez consulter la page suivante :

<http://devedge.netscape.com/library/manuals/2000/javascript/1.3/reference/>

**3.7.3 Interaction avec les formulaires**

**Accès aux textes contenus dans les champs**

Les éléments de formulaire concernés sont :

* boîtes de saisie de texte (INPUT type="text")
* saisie de textes sur plusieurs lignes (TEXTAREA)
* boîtes de saisie masquées (INPUT type="password")
* boîtes de soumission de fichier (INPUT type="file")
* données masquées (INPUT type="hidden")

**Lecture**

window.document.nom\_formulaire.nom\_champ.value

**Écriture**

window.document.nom\_formulaire.nom\_champ.value = texte;

NB : le champ contiendra le texte affecté à l'élément, ce texte remplacera celui qui était affiché.

**Lire le texte par défaut**

Lorsqu'on définit les éléments de formulaires textuels il est possible de donner un texte par défaut qui s'affiche lorsque le formulaire est chargé (par exemple un nombre de personnes à 4 ou une adresse à n'oubliez pas de donner l'étage).

Pour obtenir dans un script le texte qui est affiché au chargement, il suffit d'écrire :   
window.document.nom\_formulaire.nom\_champ.defaultValue

**Accès aux choix cochés par le client**

Les éléments de formulaire concernés sont :

* cases à cocher (INPUT type="checkbox")
* boutons radio (INPUT type="radio")

**Recherche des choix cochés**

window.document.nom\_formulaire.nom\_champ[num\_case].checked (retourne une valeur true/false)

window.document.nom\_formulaire.nom\_champ[num\_case].value (retourne le nom donné à la boîte)

**Cocher ou décocher un choix**

window.document.nom\_formulaire.nom\_champ[num\_case].checked = true/false

**Obtenir le choix coché par défaut au chargement**

window.document.nom\_formulaire.nom\_champ[num\_case].defaultChecked

**Accès aux choix dans les listes de choix**

L'élément de formulaire concerné est SELECT

**Lire les choix**

window.document.nom\_formulaire.nom\_champ.options[num\_case].selected ( retourne une valeur true/false)

window.document.nom\_formulaire.nom\_champ.options[num\_case].value (retourne le nom donné au choix)

window.document.nom\_formulaire.nom\_champ.length (retourne le nombre de choix possibles dans la liste)

window.document.nom\_formulaire.nom\_champ.options[num\_case].text (retourne le texte HTML associé)

NB : pour les listes à choix unique il est possible d'obtenir directement quel est le choix sélectionné avec :  
window.document.nom\_formulaire.nom\_champ.selectedIndex

**Modifier les choix**

window.document.nom\_formulaire.nom\_champ.options[num\_case].selected = true/false

Dans l'exemple ci-dessous nous supposons qu'on peut choisir plusieurs plats (attribut multiple dans SELECT)

**Accès aux textes sur les boutons**

Les éléments de formulaire concernés sont :

* boutons de soumission (INPUT type="submit")
* boutons de soumission avec image(INPUT type="image")
* boutons d'annulation (INPUT type="reset")
* boutons simples (INPUT type="button")

window.document.nom\_formulaire.nom\_boutton.value

**Evènements dans un formulaire**

Lorsque le client clique dans le formulaire, modifie des champs, coche des boutons radio..., il est possible d'exécuter du code JavaScript. Ce code sert généralement à :

* vérifier la validité des données entrées (pas de lettres dans un numéro de fax)
* donner des informations (affichage d'un message dans la barre d'état lorsqu'on clique dans un champ de formulaire)

Les événements se placent dans l'élément de formulaire comme des attributs, ils sont de la forme **nom\_evenement = "code JavaScript"**, le code JavaScript étant la plupart du temps un appel à une fonction JavaScript définie plus haut dans la page.

ex :  
<INPUT type="button" value="Informations" name="binfo" onClick="Info()">

Les évènements suivants sont utiles :

         onClick (clic sur un élément). Balises concernées :

o        cases à cocher (INPUT type="checkbox")

o        boutons radio (INPUT type="radio")

o        boutons de soumission (INPUT type="submit") (on devra utiliser une instruction JavaScript return pour indiquer si la soumission doit être réalisée)

o        boutons de soumission avec image (INPUT type="image")

o        boutons d'annulation (INPUT type="reset")

o        boutons simples (INPUT type="button")

         onChange (modification dans l'élément). Balises concernées :

o        boîtes de saisie de texte (INPUT type="text")

o        saisie de textes sur plusieurs lignes (TEXTAREA)

o        boîtes de saisie masquées (INPUT type="password")

o        boîtes de soumission de fichier (INPUT type="file")

o        listes (SELECT)

         onFocus, onBlur (attribution, perte du focus). Balises concernées :

o        boîtes de saisie de texte (INPUT type="text")

o        saisie de textes sur plusieurs lignes (TEXTAREA)

o        boîtes de saisie masquées (INPUT type="password")

o        boîtes de soumission de fichier (INPUT type="file")

o        listes (SELECT)

         onSelect (sélection dans un champ). Balises concernées :

o        boîtes de saisie de texte (INPUT type="text")

o        saisie de textes sur plusieurs lignes (TEXTAREA)

o        boîtes de saisie masquées (INPUT type="password")

o        boîtes de soumission de fichier (INPUT type="file")

* onReset, onSubmit. Balise concernée : FORM

Lorsque le formulaire est soumis en appuyant sur le bouton de soumission il est possible d'appeler un événement onSubmit (pour vérifier par exemple la validité des données avant d'envoyer le formulaire ou pour demander la confirmation de la soumission). Le script devra renvoyer soit true soit false (avec une instruction JavaScript return) pour indiquer si on doit soumettre ou non le formulaire.

L'événement reset est appelé lorsqu'on a cliqué sur le bouton annuler.

**Quelques méthodes sont aussi disponibles**

|  |  |
| --- | --- |
| **nom** | **rôle** |
| blur() | enlève le focus d'un champ du formulaire |
| click() | simule un clic de souris sur le champ du formulaire |
| focus() | attribue le focus à un champ du formulaire |
| select() | sélectionne le contenu d'un champ du formulaire |

Exemple : window.document.nom\_form.nom\_champ.focus()

Il existe également les méthodes submit() et reset() sur le formulaire.

**Un exemple complet**

Soit le formulaire suivant :

<form name=commande action="traitement.php" method="post" onSubmit=”return Verif();”>

    Nom : <input type="text" name="**nom**"><br>

    Email : <input type="text" name="**email**"><br>

    Boisson choisi : <br>

    <input type=radio name="**boisson**" value="medoc">Médoc<br>

    <input type=radio name="**boisson**" value="madiran">Madiran<br>

    <input type=radio name="**boisson**" value="chablis">Chablis<br>

    <input type=radio name="**boisson**" value="riesling">Riesling<br>

    Quantité :

    <input type=text name="**quantite**" onFocus="window.status='Entrez le nombre de caisses'">

    <br>

    <input type="submit" onClick="return Confirmer()">

</form>

Voici les fonctions JavaScript qu’on peut placer par exemple dans l’entête du document :

<SCRIPT language="JavaScript">

<!--

// fonction qui vérifie si « chaine » est une chaîne composée de chiffres

// elle affiche le message d’erreur « message\_d\_erreur » dans le cas contraire

function VerifNb(chaine, message\_d\_erreur)

{

  var chiffre = true;

  // si la chaîne est vide elle n’est pas correcte

  // les chaînes comme les tableaux ont la propriété length

  if (chaine.length == 0) {

    window.alert(message\_d\_erreur);

    return false;

  }

  // si on trouve un caractère non compris entre 0 et 9 c’est incorrect

  for(i=0 ; i < chaine.length; i++)

      // la méthode charAt permet de récupérer le I-ème caractère

      if(chaine.charAt(i)<'0'||chaine.charAt(i)>'9') {

            chiffre=false;

      }

  if(!chiffre)

  {

    window.alert(message\_d\_erreur);

    return false;

  }

  else {

     return true;

  }

}

// fonction qui vérifie le formulaire

function Verif()

{

  with(window.document.commande) // pour éviter de répéter le nom du formulaire

  {

    // on vérifie que le nom et l’adresse e-mail ont été saisis

    if(nom.value.length == 0)

    {

       alert("veuillez entrer votre nom");

       return false;

    }

    if(email.value.length == 0)

    {

       alert("veuillez entrer une adresse email");

       return false;

    }

    // on regarde si un boisson a été choisi

    var choix = false;

    for(i=0; i<boisson.length; i++) {

         if(boisson[i].checked) choix = true;

    }

    if(!choix) {

       alert("veuillez choisir un boisson");

       return false;

    }

    // on vérifie la quantité avec la fonction ci-dessus

    if(!VerifNb(quantite.value, "veuillez écrire le nombre de caisses en chiffres"))

    {

        // si ce n’est pas bon on sélectionne le contenu du champ quantité

        // pour que l’utilisateur n’est plus qu’a corriger

        quantite.focus();

        quantite.select();

        return false;

    }

  }

  // si on renvoie true le formulaire sera soumis

  return true;

}

// fonction qui demande à l’utilisateur s’il veut vraiment soumettre le form

function Confirmer()

{

    return window.confirm("Voulez-vous vraiment soumettre ?");

}

//-->

</SCRIPT>

**4. ressources web**

      <http://devedge.netscape.com/library/manuals/2000/javascript/1.3/reference/>

      <http://magali.contensin.free.fr/html/InternetLicPro.html> (cours sur le web dont est tiré celui-ci)

      <http://www.editeurjavascript.com/cours/index.php>