Une Introduction à la programmation événementielle

Application à la programmation web

et sa pratique en JavaScript

Interprétation de code JavaScript dans un navigateur via du code HTML.

Prolégomène

Protocole Internet : Transmission Control Protocol/Internet Protocol

Un des protocoles du Web : HyperText Transfer Protocol

Adresse du Web (Uniform Resource Locator) :

`nom_du_protocole://nom_de_domaine/chemin_d'accès`

Langage principal des pages Web : HyperText Markup Language.

Caractéristiques de HTML

Langage de balises.

Langage interprété.

Langage de description de document hypertexte.

Format HTML

Les balises représentent les instructions.

Les balises vont souvent par paires.

Le format habituel d'un document HTML

```
<HTML>
 <HEAD>
  <TITLE> Le nom du document </TITLE>
 </HEAD>
 <BODY>
  Le corps du document : du code HTML
 </BODY>
</HTML>
```

Caractéristiques de JavaScript

Langage interprété.

Langage dépendant d'un hôte pour les entrées/sorties.

Langage faiblement typé.

Langage teinté d'objet.

Etat des forces en présence

JavaScript de Netscape.

JScript de Microsoft.

Un Standard ECMAScript.

JavaScript : Langage de l'Internet

Script résidant sur le serveur web :
- en fichier indépendant `.js` ou
- inseré dans une page HTML.

Téléchargé par le navigateur (le client).

Exécuté (interprété) par le navigateur.

Entrée/Sortie via HTML (`document.write(expression)`).

Déboggage via une console :
`javascript:` dans l'adresse du navigateur.

Les services de JavaScript

- agir sur la mise en forme du document lors de son chargement ;
- agir sur les fenêtres affichées par le navigateur ;
- détecter les actions de l'utilisateur (gestion des événements) ;
- réaliser des automatismes simples ;
- réaliser des tâches en retour ;
- enrichir les formulaires ;
- coopérer avec d'autres produits externes au navigateur.

Les types

- `boolean`
- `string`
- `Number`
- `Array`
- `Object`

La fonction `typeof(variable)` renvoie le type de la variable.

null est la valeur des variables n'en ayant pas reçue.

undefined correspond à une erreur.

La conversion d'une chaîne de caractère en nombre décimal : parseFloat()

La conversion d'une chaîne de caractère en nombre entier : parseInt()

La fonction isNaN() teste si la valeur passée en argument n'est pas un nombre.

La conversion d'un nombre en une chaîne de caractère : toString().

Javascript ne dispose pas de type particulier pour les caractères. (C'est la chaîne de caractères avec un seul caractère qui en fait office.)

Les variables et leur déclaration

Déclaration d'une variable :

```
var nom_de_variable = une_valeur;
```

Le script `document.write(une_autre_variable)` dans cette page a généré une erreur (car la variable `une_autre_variable` n'est pas définie) qui apparaît dans la console mais qui n'empêche pas l'affichage de la page.

Les variables et l'affectation

En chargeant une nouvelle page HTML, le contexte de l'interprète est réinitialisé.

Langage faiblement typé : les variables peuvent prendre plusieurs types différents au cours d'une exécution :

```
var une_variable = true;
```

Le type de la variable `une_variable` est `boolean` et sa valeur est `true`.

```
une_variable = 1;
```

Le type de la variable `une_variable` est maintenant `number` et sa valeur est 1.

Les fonctions

```
function nom_de_la_fonction (parametre,...,parametre) {
   Instructions JavaScript
}
```

Les variables (et objets) déclarées dans une fonction lui sont locales.

Le résultat d'une fonction est renvoyé par l'instruction return.

Script pour calculé une racine d'une équation du second degré :

```
function racine_simpliste(a,b,c) {
   var delta = b*b-4*a*c;
   return (-b-Math.sqrt(delta))/(2*a);
}
```

Une racine de l'équation `-5x^2 + 3x + 2 = 0` est `racine_simpliste(coef_a,coef_b,coef_c)` (=1).

La conditionnelle

```
if (Condition) {alors} else {sinon}
```

La boucle while

```
while (TantQueConditionVraie) {faire}
```

La boucle for

for (Initialisation; TantQueConditionVraie; Incrémentation){faire}


```
var i,j;
for (i = 1; i < 6; i++) {
   for (j = 1; j <= 6-i ; j++) {
      document.write(" ");
   }
   for (j = 1; j <= i ; j++) {
      document.write("<TT>*</TT>");
   }
   document.write("<BR>");
}
i = 3;
while(i != 0) {
   document.write("    <TT>*</TT><BR>");
   i = i - 1;
}
```

JavaScript : un langage teinté d'objet

Objet JavaScript : données (attributs) et fonctions (méthodes) regroupées sous une même désignation formant l'ensemble des propriétés de l'objet.

Création d'un objet ne contenant aucune propriété :

```
nouvel_objet = new Object
```

Evoquer une propriété, c'est la créer !

```
un_objet.attribut_1 = valeur_1;
un_objet.attribut_2 = new Object;
un_objet.attribut_2.sous_attribut = valeur_2;
un_objet.methode = un_nom_de_fonction;
un_objet.methode(parametre,...,parametre);
```


Constructeur et prototype

Création d'un objet selon un modèle (pseudo-classe) :
- constructeur +
- prototypes (propriétés identiquement initialisées à la création).

```
/* Le constructeur */
function nom_du_modele(parametre_1,...,parametre_n) {
   this.propriete_1 = parametre_1;
   ...
   this.propriete_n = parametre_n;
}
/* Les prototypes */
nom_du_modele.prototype.propriete_n+1 = propriete_commune_1;
...
nom_du_modele.prototype.propriete_n+p = propriete_commune_p;
```


Avec les déclarations suivantes :

```
function mon_proto(val) {
   this.val = val;
}
mon_proto.prototype.objet = new Object;
mon_proto.prototype.objet.val = "init mon_proto.prototype.objet.val";
var elt1 = new mon_proto("init elt1.val");
var elt2 = new mon_proto("init elt2.val");
```

les objets `elt1` et `elt2` sont initialisés à :

```
elt1.val : init elt1.val
elt2.val : init elt2.val
elt1.objet.val : init mon_proto.prototype.objet.val
elt2.objet.val : init mon_proto.prototype.objet.val
```


Mais avec la modification suivante :

```
elt1.val = "modif elt1.val";
elt1.objet.val = "modif elt1.objet.val";
```

les objets ont pour attributs :

```
elt1.val : modif elt1.val
elt2.val : init elt2.val
elt1.objet.val : modif elt1.objet.val
elt2.objet.val : modif elt1.objet.val
```

Le prototype lui-même a été modifié :

```
var elt3 = new mon_proto("init elt3.val");
elt3.val :  modif elt3.val
elt3.objet.val : modif elt1.objet.val
```

Les tableaux

Déclaration d'un tableau soit par sa taille, soit explicitement :

```
new Array(taille_du_tableau)
new Array(elt1,...,eltn)
```

Accès aux éléments (le premier élément est d'indice 0) :

```
nom_du_tableau[indice]
```

Accès à la taille du tableau par l'attribut length


```
var les_mois = new Array(12);
les_mois[0]="Décembre";
var les_jours_du_WE = new Array("Samedi","Dimanche");
var nb_jours = les_jours_du_WE.length /* nb_jours == 2 */
```

Un objet est un tableau à indexation littérale. (Un mot et non seulement un entier peut servir d'index).

La boucle for in

Les tableaux sont des objets.

for (variable in objet){faire}

```
var copie = new Object;
for (prop in objet) {
   if (typeof(objet[prop])== "object") {
      copie[prop] = copieur(objet[prop]);
   }
   else {
      copie[prop] = objet[prop];
   }
}
```


Où mettre du JavaScript dans du HTML ?

Directement le code :

```
<SCRIPT> Code JavaScript </SCRIPT>
```

ou bien l'appel à un fichier de scripts.

```
<SCRIPT SRC="Fichier_JavaScript.js"> </SCRIPT>
```

Dans l'entete du fichier HTML <HEAD>...</HEAD> pour les déclarations et dans le corps <BODY>...</BODY> pour ce qui doit être affiché et calculé.

Les entrées/sorties via les formulaires HTML

Un formulaire HTML est un ensemble d'entrées/sorties exploitables par JavaScript.

Les informations d'un formulaire sont exploitables par un serveur Web via un script Common Gateway Interface.

```
<FORM NAME="nom_du_formulaire">
Des balises HTML de mise en page et des balises d'entrées/sorties :
<INPUT NAME="nom_de_l_élément" caractéristiques de l'élément >
</FORM>
```

Chaque formulaire est accessible par son nom :

```
document.nom_du_formulaire
```


Chaque élément d'un formulaire est accessible par son nom :

```
document.nom_du_formulaire.nom_de_l_élément
```

En cas de conflit sur les noms :

Chaque $i^{ème}$ formulaire est aussi accessible via le tableau `forms`, attribut de l'objet `document` : `document.forms[i-1]`.

Chaque $j^{ème}$ élément est aussi accessible via le tableau `elements`, attribut de l'objet `document.forms[i-1]` :`document.forms[i-1].elements[j-1]`.

Les attributs communs aux éléments

L'attribut `NAME` qui nomme l'élément ;

L'attribut `TYPE` qui qualifie l'entrée/sortie :
`TEXT`, `BUTTON`, `RADIO`, `CHECKBOX`, `HIDDEN`, etc...

L'attribut `VALUE` qui référence la valeur de l'attribut.

Quelques éléments de formulaire

La zone de texte :

```
<INPUT TYPE="TEXT" NAME="login" VALUE="" SIZE=20 MAXLENGHT=18>
```

La bouton :

```
<INPUT TYPE="BUTTON" NAME="mon_bouton" VALUE="le_texte_sur_le_bouton">
```

Les cases à cocher (n parmi) :

```
<INPUT TYPE="CHECKBOX" NAME="boisson" VALUE="eau"> Eau <BR>
<INPUT TYPE="CHECKBOX" NAME="boisson" VALUE="jus_d_orange"> Jus
d'Oranges
```

Les boutons d'option (un parmi) :

```
<INPUT TYPE="RADIO" NAME="sexe" VALUE="homme"> Homme <BR>
<INPUT TYPE="RADIO" NAME="sexe" VALUE="femme" CHECKED> Femme
```

Le mot de passe :

```
<INPUT TYPE="PASSWORD" NAME="mot_de_passe">
```


La soumission du formulaire :

```
<INPUT TYPE="SUBMIT" VALUE="Envoyer">
```

La soumission du formulaire :

```
<INPUT TYPE="RESET" VALUE="Effacer">
```

L'élément caché du formulaire :

```
<INPUT TYPE="HIDDEN" NAME="resultat">
```


D'autres éléments de formulaire

La grande zone de texte :

```
<TEXTAREA NAME="Commentaires" ROWS=5 COLS=60></TEXTAREA>
```

Le menu avec les balises <OPTION> :

```
<SELECT NAME="age" SIZE=2>
<OPTION VALUE="jeune" SELECTED> Moins de 40 ans
<OPTION VALUE="vieu"> Plus de 40 ans
</SELECT>
```


L'option sélectionnée est obtenue via :

```
i = nom_du_formulaire.nom_du_menu.selectedIndex
```

par :

```
nom_du_formulaire.nom_du_menu.options[i].value
```

La programmation événementielle en JavaScript

Programmation événementielle : capture (ou interception) de l'interaction entre l'hôte et l'utilisateur (les événements) pour y réagir par des tâches.

Avant le chargement de la page :

```
<BODY ONLOAD="Code_JavaScript">
```

Via un lien :

```
<A HREF= "javascript:Code_JavaScript">
lien pour lancer le code JavaScript
</A>
```


Lors de la modification d'un attribut d'un formulaire :

```
<FORM NAME="de_l_un_a_l_autre">
<INPUT NAME="attribut_0" TYPE="TEXT"
 ONCHANGE="document.de_l_un_a_l_autre.attribut_1.value=
  document.de_l_un_a_l_autre.attribut_0.value;">
<INPUT NAME="attribut_1" TYPE="TEXT"
 ONCHANGE="document.forms[0].elements[0].value=
  document.forms[0].elements[1].value;">
</FORM>
```


Interception d'événements souris

Avant l'utilisation d'un lien :

```
<A HREF= "URL" ONCLICK="Code_JavaScript"> lien vers l'URL </A>
```

En se positionnant sur un lien :

```
<A HREF= "URL" ONMOUSEOVER="Code_JavaScript"> lien vers l'URL </A>
```

En quittant un positionnement sur un lien :

```
<A HREF= "URL" ONMOUSEOUT="Code_JavaScript"> lien vers l'URL </A>
```


Fenêtres préformatées

Le simple message d'alerte :

```
alert("Message de la fenêtre d'alerte");
```

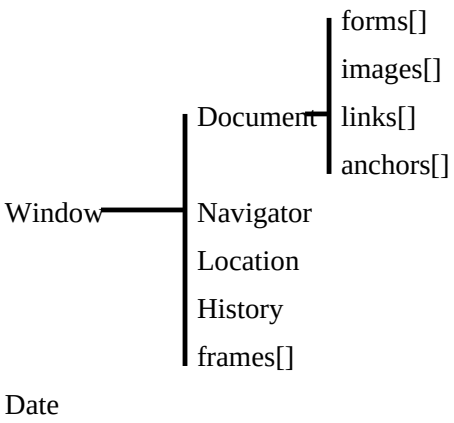
La boîte de confirmation :

```
un_boolean = confirm("Message de la fenêtre de confirmation");
```

La boîte de dialogue :

```
une_string = prompt("Message de la fenêtre de dialogue",
                    "Ecrivez ici");
```

Hiérarchie des objets



L'objet Window

Appel à une méthode de l'objet `window` pour ouvrir une fenêtre.

```
variable = window.open("URL","Nom","opt_1=val_1,...,opt_n=val_n")
```

Les caractéristiques les plus classiques : `WIDTH` et `HEIGHT` exprimées en points.

Dans une fenêtre, celle-ci est accessible via le mot-clé window.

La fenêtre appelante est accessible par la propriété opener.

L'appel à la méthode `close()` ferme la fenêtre.

L'objet Document

L'objet Document comporte l'ensemble des propriétés relative à la page HTML.

Les champs forms[], images[], links[], anchors[] contiennent respectivement l'ensemble des formulaires, l'ensemble des images, l'ensemble des liens, l'ensemble des ancres.

Quelques méthodes de l'objet Document :
open() : l'ouverture ;
clear() : l'effacement ;
write() : l'écriture ;
close() : la fermeture (l'affichage)

L'objet Navigator

L'objet navigator est une propriété de l'objet window.

Les propriétés de l'objet navigator permettent de reconnaître le navigateur utilisé, en particulier :
`appName` : le nom du navigateur,
`appVersion` : le numéro de version.

L'objet Location

L'objet Location comporte l'ensemble des informations sur le document affiché dans la fenêtre.

La propriété href est la chaîne de caractère de l'URL.

Pour recharger le document :

```
<A HREF="javascript:location.reload(true)">
```

Bouton de retour à la page précédente via location :

```
<A HREF="javascript:location.href='nom_page_precedente.html'">
```


L'objet history

L'objet History comporte l'ensemble des informations sur l'historique du document.

Revenir au document précédent :

```
<A HREF="javascript:window.history.go(-1)">
```


L'objet Date

L'objet flashDate permet d'obtenir et de fixer la date et l'heure dans de nombreux standard.

```
aujourdhui = new Date();
```

Les propriétés principales de l'objet `Date` :
`getSeconds()` : les secondes entre 0 et 59 ;
`getMinutes()` : les minutes entre 0 et 59 ;
`getHours()` : les heures entre 0 et 23 ;
`getDate()` : le jour du mois compris entre 1 et 31 ;
`getMonth()` : le mois compris entre 0 et 11 ;
`getFullYear()` : l'année.