Programmation Web

JavaScript

Jean-Christophe Deneuville

jean-christophe.deneuville@xlim.fr

Université de Limoges
FACULTÉ DES SCIENCES ET TECHNIQUES

Xlim
institut de recherche

Historique

- HTML et XHTML sont des langages déclaratifs
  - difficile de réaliser des traitements (calculs, animations, etc.) à partir d'eux seuls
- Afin de « dynamiser » les pages Web, Netscape a inventé un langage de script, appelé initialement LiveScript et rebaptisé JavaScript

Présentation

- JavaScript est un langage reprenant quelques éléments de syntaxe de Java
- On l'intègre à des pages HTML et ses instructions sont exécutées par le navigateur.
- Il permet notamment de faire :
  - un contrôle du contenu des différents champs d'un formulaire,
  - des animations (comme des rollovers),
  - des calculs (sur des réels, des dates, …).
- Il est possible de faire des programmes JavaScript qui ne s'exécutent pas à l'intérieur d'un navigateur => scripts ASP qui sont exécutés par le serveur Web

Les surnoms de JS

- Les différentes appellations de JavaScript :
  - JavaScript : terme générique pour le langage (désignait initialement l'implémentation du langage par Netscape dans Netscape Navigator).
  - JScript : l'implémentation par Microsoft de JavaScript dans Internet Explorer.
  - ECMAScript : la normalisation de JavaScript accomplie par l'ECMA (European Computer Manufacturers Association).

Diverses façons d'intégrer un script dans une page HTML : Méthode 1

- Par écriture du code entre les balises <head> et </head> du document :

```
<html>
        <head>
            <title>Titre de la page</title>
            <script language="JavaScript">
                // Initialisation de variables globales
                /* Définitions  de fonctions */
            </script>
        </head>
        <body>
            <!-- … -->
        </body>
</html>
```

Méthode 2

- Par inclusion d'un fichier séparé contenant le code JavaScript (à la façon de #include en C) :

```
<html>
	<head>
		<title>Titre de la page</title>
		<script language="JavaScript" src="mesFonctions.js" >
		</script>
	</head>
	<body>
		<!-- … -->
	</body>
</html>
```

- C'est une solution préférable à la précédente quand on se préoccupe de la réutilisation des scripts.

Méthode 3

- Par écriture du code dans le corps du document :

```
<html>
	<head>
		<title>Titre de la page</title>
	</head>

	<body>
		<script language="JavaScript">
			document.write("<h1>Salut</h1>");
			document.write("<p>" + document.lastModified + "</p>");
		</script>
	</body>
</html>
```

Résultat obtenu avec Firefox 1.0

Salut

Monday, March 21, 2005 16:38:28

Méthode 4

- En utilisant un gestionnaire d'événement (Le code JavaScript sera exécuté quand un certain événement se produira) :

```
<html>
	<head>
	    <title>Titre de la page</title>
	</head>
	<body onLoad="document.getElementById('texte').style.color='blue';">
		<div id="texte">
		    Bla bla
		</div>
	</body>
</html>
```

- Remarque : La notation permettant d'aboutir jusqu'à l'attribut « color » de l'élément rappelle celle d'un langage objet comme Java.

Méthode 5

- Sous forme d'une URI :

```
<a href="javascript:window.alert('essai');">
Cliquez
</a>
```

- A chaque fois que l'internaute cliquera sur le lien, une boîte de message avec le message « essai » apparaîtra.

Eléments fondamentaux du langage
Les différents types de variables

- JavaScript est un langage où il n'y a pas besoin de déclarer le type des variables.
- Quand on attribue une valeur à une variable par une affectation, le type adéquat est déterminé automatiquement par l'interpréteur JavaScript.
- On parle de typage au niveau des valeurs.
- Le type d'une variable est de plus susceptible de varier au cours de l'exécution

Les différents types de variables

- On dispose des principaux types suivants :
  - nombres entiers et réels (tous les deux traités en interne comme des réels sur 64 bits à la norme IEEE 754),
  - booléen, valant « true » ou « false »,
  - chaîne de caractères (qui sont des objets, comme en Java),
  - objets (fournis par le navigateur ou définis par le programmeur).
- Par ailleurs, les noms des variables sont sensibles à la casse.

Exemple

```
var a;                  // Pas de type, la valeur « undefined »
                        // est attribuée à la variable a
var i = 1;              // Le mot-clé « var » est optionnel
var x = 0.17;
var erreur = false;
var chaine = "Hello" + ' world' ;    // + : opérateur de concaténation
var message = new String("Hello");   // Syntaxe objet
var s = 12 + "travaux";              // Conversion automatique
                                     // de l'entier en chaîne
var nombre = Math.round(0.55);       // Math est un objet proposé
                                     // par l'interpréteur JavaScript
var maintenant = new Date();         // Création d'une instance de la classe
                                     // Date, classe fournie par l'interpréteur
```

Quelques champs, méthodes, opérateurs et fonctions en rapport avec les chaînes de caractères

- + : concaténation.
- chaine.length : taille de la chaîne.
- chaine.toUpperCase() : renvoie la chaîne convertie en majuscule.
- chaine.toLowerCase() : renvoie la chaîne convertie en majuscule.
- chaine.substring(debut,fin) : renvoie la sous-chaîne constituée des caractères de la chaîne entre « debut » (inclus) et « fin » (exclu). Les indices des caractères formant une chaîne commencent à 0.
- chaine.charAt(pos) : fournit le caractère de la chaîne situé à l'indice « pos ».
- chaine.indexOf(sous_chaine) : recherche de la position d'une sous-chaîne dans une chaîne à partir du début de cette dernière.
- chaine.indexOf(sous_chaine,pos) : recherche de la position d'une sous-chaîne dans une chaîne à partir de l'indice « pos ».
- parseInt(chaine,base) : retourne la conversion de la chaîne en un entier (renvoie la valeur spéciale « NaN » (Not a Number) s'il n'y a pas un nombre au début de la chaîne).
- parseFloat(chaine) : retourne la conversion de la chaîne en un réel.

- Exemple :

```
<script language="JavaScript">
   msg = "salut";
   msg = msg.toUpperCase();
  document.write(msg.substring(1,msg.length));
</script>
```

- Résultat d'exécution du script ALUT
- Remarque : on dispose des séquences d'échappement « \" » et « \' » pour l'insertion de ces caractères dans des littéraux chaînes délimités par ceux-ci.

Les tableaux en JavaScript

- Création d'un tableau :
  - var tableau = new Array();
    Ceci crée un tableau vide. En JavaScript, tout tableau est un tableau dynamique (c'est-à-dire dont la taille peut évoluer au cours du temps).
  - var mon_tab = new Array("far","boo",3);
  - var mon_tab = ["far","boo",3] ;
    Le tableau est rempli avec 3 éléments de types différents.
  - var tab = new Array(30);
    Le tableau « tab » dispose de 30 éléments (de n'importe quel type). Ses indices vont de 0 à 29. Les valeurs des éléments sont fixées par défaut à la valeur spéciale « undefined ».

Les tableaux en JavaScript

- Modification des éléments d'un tableau :
  - `tab[0] = new String("far");`
  - `tab[1] = new String("boo");`
- Taille d'un tableau :
  - On dispose de l'attribut length pour obtenir la taille d'un tableau.
  - `var taille = tab.length; // la taille vaut 30`

- Tri d’un tableau :
  - On utilise la méthode `sort`.
- Exemple d’utilisation :

```
var mon_tab = new Array("far","boo",3);
document.write(mon_tab+"<br />");
mon_tab.sort();
document.write(mon_tab+"<br />");
```

- Résultat obtenu :

far,boo,3
3,boo,far

- On note que le nombre 3 a été traité comme une chaîne (dans l’ordre ASCII, les nombres sont avant les lettres).

- Découpage d'une chaîne avec placement des résultats dans un tableau :
- La méthode `split` permet de découper une chaîne en plusieurs sous-chaînes stockées dans un tableau. Le découpage se fait en indiquant un caractère servant de séparateur.
- Exemple d'utilisation :

```
var parties = new Array();
var chaine = "Le d&eacute ;but des cours";
parties = chaine.split(" ");
for (indice in parties)
    document.write(parties[indice] + "<br />");
```

- Résultat obtenu :

Le
début
des
cours

- Concaténation des éléments d'un tableau en une chaîne :
- C'est l'opération inverse de la précédente. On utilise à cette fin la méthode `join`.
- Exemple :

```
var tableau = new Array("nom","prenom","mot_de_passe");
var separateur = ":";

var chaine = tableau.join(separateur);

document.write(chaine,"<br />");
```

- Résultat obtenu :

nom:prenom:mot_de_passe

Les fonctions

- On définit une fonction avec le mot-clé « function ». On indique des paramètres formels s'il est besoin. L'instruction « return » permet quant à elle de renvoyer une valeur.

- Exemples de définitions de fonctions :

```
function somme(x,y)
{
    return x + y;
}

function premierCaractere(s)
{
    return s.charAt(0);
}
```

- Exemples d'utilisation :

```
var resultat = somme(2,3);
      document.write(resultat + "<br />");
      var msg = "coucou";
      document.write(premierCaractere(msg.toUpperCase()) + "<br />");
      afficherNombre(somme(1,5));
```

- Résultat obtenu :

5

C

6

- Si l'on ajoute la ligne suivante :

```
      var c = premierCaractere(true);
```

Erreur : s.charAt is not a function

```
function afficherNombre(n)
{
var chaine = "<em>"; // var permet de définir une variable locale
// qui pourra masquer une variable globale
// de même nom.
chaine += n;      // Equivalent à : chaine = chaine + n;
chaine += "</em>" ;

document.write(chaine) ;
}
```

Les structures de contrôle

- On retrouve les principales structures de contrôle de Java (provenant du C).
- Exemples :

```
if (x < 10)
   x++;
else
   document.write(x + "<br />");
```

- Remarque :
  - On dispose des opérateurs de comparaison classiques suivants : ==, !=, <, >, <=, >=.
  - Les opérateurs === et !== testent en plus l'égalité ou l'inégalité des types des deux opérandes (Ils n'effectuent pas de conversion automatique de type).
- Ainsi :

```
var x = 12;
if (x == "12")
    document.write("Test N°1 : vrai.<br>");
else
    document.write("Test N°1 : faux.<br>");
if (x=== "12")
    document.write("Test N°2 : vrai.<br>");
else
    document.write("Test N°2 : faux.<br>");
```

- a pour résultat :

Test N°1 : vrai.
Test N°2 : faux.

- Nota Bene : Contrairement à Java, on peut utiliser les opérateurs de comparaison conventionnels (==, !=, <=, etc.) avec des chaînes de caractères.
- On peut employer par ailleurs les opérateurs logiques suivants : !, || et &&.

```
switch(chaine)
{
    case "debut" :
      document.write("D&eacute;but.<br>");
      break;
    case "fin" :
      document.write("Fin.<br>");
      break;
    default :
      document.write("En cours...<br>");
}
```

```
for (indice = 1 ; indice < 10 ;
  indice++)
    valeur = valeur * indice;


while (indice < 10)
   indice++ ;


do
   indice++;
while (!(indice == 10));
```

L'instruction `break` termine l'exécution de la boucle la plus imbriquée.

Création et manipulation d'objets

- Supposons que nous désirions créer des objets d'une classe Personne composée de 3 champs : « nom » et « prenom » de type chaîne, « age » de type entier.
- On définit pour cela un constructeur qui est une fonction prenant le nom de la classe :

```
function Personne(nom,prenom,age)
   {
      this.nom=nom;
      this.prenom=prenom;
      this.age=age;
   }
```

- « this » représente l'objet courant (comme en Java). On voit que le contrôle des types se fera dans les méthodes (Si l'on traite dans une méthode de la classe Personne le champ « age » comme une chaîne il y aura probablement une erreur).

- Ecrivons maintenant une méthode pour cette classe :

```
function afficher()
{
	aff_nom = "Nom : " + this.nom + "<br />";
	aff_prenom = "Prenom : " + this.prenom + "<br />";
	aff_age = "&Acirc;ge : " + this.age + "<br />";
	document.write(aff_nom + aff_prenom + aff_age);
}
```

- Il faut à présent relier la fonction ainsi définie à la classe Personne. Pour cela, on rajoute l'instruction suivante dans le constructeur :

```
this.afficher=afficher;
```

- On peut aussi – et c'est sans doute plus lisible - intégrer la définition de la méthode « afficher » directement dans le constructeur.
- Exemple complet avec création d'objets :

```
<script language="JavaScript">
          function Personne(nom,prenom,age)
          {
              this.nom=nom;
              this.prenom=prenom;
              this.age=age;
```

```
        function afficher()
        {
        aff_nom = "Nom : " + this.nom + "<br />";
        aff_prenom = "Prenom : " + this.prenom + "<br />";
        aff_age = "&Acirc;ge : " + this.age + "<br />";
        document.write(aff_nom + aff_prenom + aff_age);
        }
        this.afficher=afficher;
       }
        // Creation et manipulation d'objets de la classe Personne
        var individu = new Personne("Durand","Eric",23);
        individu.afficher();
        individu.age=87;
        individu.afficher();
</script>
```

Résultat :

Nom : Durand
Prenom : Eric
Âge : 23
Nom : Durand
Prenom : Eric
Âge : 87

- On note que tous les membres de la classe sont en accès public.
- On peut accéder à tous les membres d'un objet avec une syntaxe particulière de la boucle for :

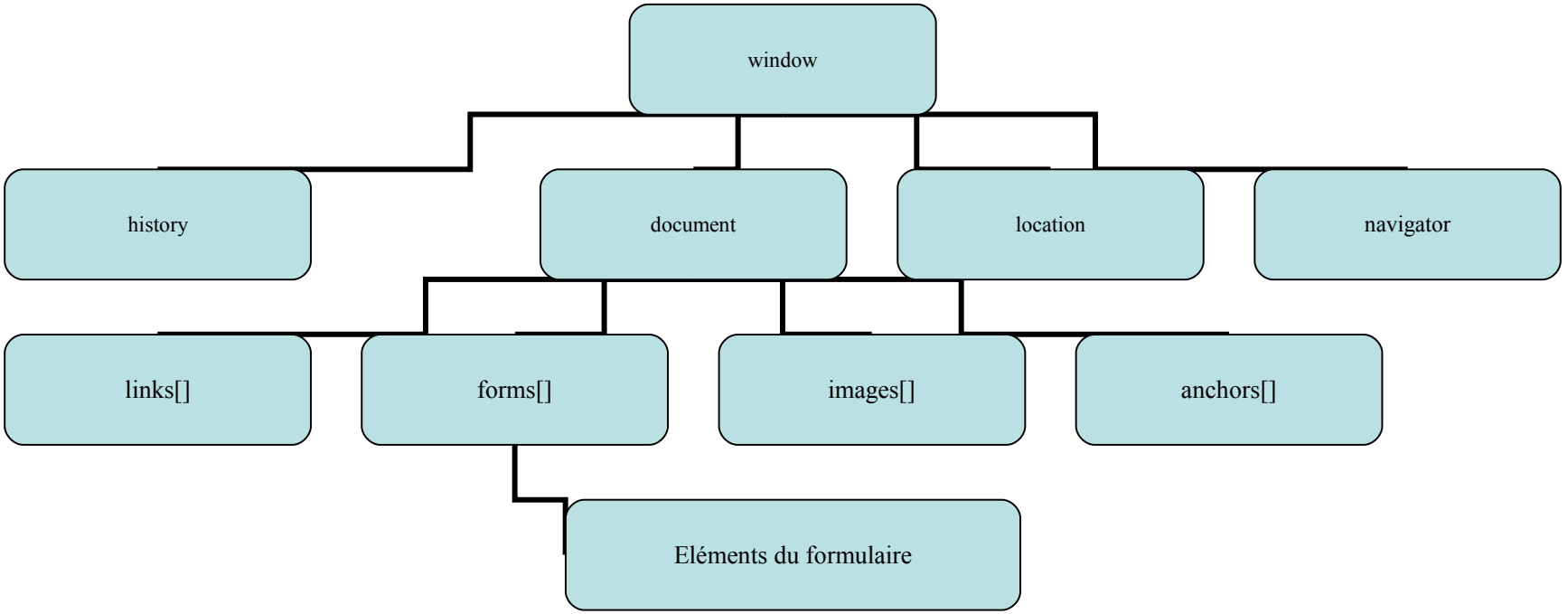
```
var individu = new Personne("Dupont","Olivier",18);
for (v in individu)
   document.write(v + "<br>");
```

Résultat :

nom
prenom
age
afficher

Le DOM

- Chaque navigateur construit en interne une hiérarchie d'objets (le DOM : Document Object Model) en rapport avec la page HTML affichée, l'état du navigateur et la fenêtre d'affichage de celui-ci.
- La façon dont est structuré le DOM et la notation des divers objets le constituant ont été standardisées par le W3C. Ceci permet d'éviter des incompatibilités entre navigateurs qui pourraient choisir des représentations spécifiques pour leur DOM.


window
history
document
location
navigator
links[]
forms[]
images[]
anchors[]
Eléments du formulaire

- Avec JavaScript, il est possible d'accéder aux objets du DOM, de consulter et parfois de modifier les propriétés de ces objets, enfin d'appeler des méthodes proposées par ces objets.
- Exemple :

```
<form  name="commande"
    action ="/cgi-bin/traiter">
    Nom du produit :
    <input type="text" name="produit">
    <br />
    <input type="submit" value="commander">
</form>
```

Dans un script, l'accès à la valeur contenue dans le champ de texte « produit » peut se faire avec l'une des syntaxes suivantes :

```
document.commande.produit.value
```

```
document.forms[0].elements[0].value
```

On peut aussi ajouter le nom de l'objet « `window` » au début des suites de noms précédentes.

L'objet `document` contient un tableau d'objets Form nommé « forms ».

`document.forms[0]` correspond au premier objet Form du document, soit le premier formulaire du document HTML.

- Un objet `Form` contient un tableau d'objets, appelé « elements », objets qui correspondent aux composants du formulaire.
- Les éléments du formulaire sont rangés dans l'ordre dans lequel ils sont déclarés dans le code HTML.

Quelques propriétés et méthodes des objets `Window`

- L'objet window correspond à la fenêtre courante du navigateur. On peut de plus créer des fenêtres pop-up qui seront eux-aussi des objets Window.
- Propriétés :
  - document : document associé à la fenêtre
  - history : historique de la fenêtre
  - location : objet en rapport avec l'URL de la fenêtre
  - status : message de la ligne d'état
  - frames[] : tableau des frames d'une fenêtre
  - length : nombre de frames de la fenêtre

Quelques propriétés et méthodes des objets `Window`

Méthodes :

- `alert(message)` : affiche un message dans une boîte de message
- `prompt(message)` : renvoie une chaîne de caractères saisie dans une boîte de dialogue
- `setTimeout(code_JavaScript,delai)` : diffère l'exécution d'un code JavaScript d'un délai exprimé en millisecondes
- `open(URL,nom,caracteristiques)` : crée une nouvelle fenêtre
- Exemple :

```
lmsi =
  window.open( "http://www.msi.unilim.fr","Laboratoire
  MSI", "width=100,height=100,toolbar=0,status=0");
```

- `close()` : fermeture d'une fenêtre
- Exemple :

```
lmsi.close();
```

Quelques propriétés et méthodes des objets `Document`.

Propriétés :

- anchors[] : tableau des ancres du document HTML
- images[] : tableau des images
- cookie : le(s) cookie(s) du document
- lastModified : date de dernière modification du document
- title : titre du document
- URL : URL du document

Quelques propriétés et méthodes des objets `Document`.

Méthodes :

- getElementById(identifiant) : renvoie l'objet correspondant à l'identifiant (attribut « id » d'une balise)
- write(valeur, …) : ajoute des données HTML au document pour l'affichage

Quelques propriétés d'éléments de formulaires.

- Les listes déroulantes :
  - options[] : tableau des différentes options de la liste
  - selectedIndex : numéro de l'option sélectionnée

  Pour obtenir dans un script la valeur choisie pour une liste, on peut donc écrire :
  - indice = document.formulaire.liste.selectedIndex;
  - valeur = document.formulaire.options[indice].value;

Quelques propriétés d'éléments de formulaires.

- Les boutons radio :

  Supposons que l'attribut « name » d'un ensemble de boutons radio soit égal à la valeur « radio ».

Alors :

- `document.formulaire.radio` est un tableau de boutons radio (d'objets Radio).
- `document.formulaire.radio[0].checked` est une valeur booléenne indiquant si c'est le premier bouton radio qui est sélectionné.
- `document.formulaire.radio[0].value` correspond à la valeur indiquée dans l'attribut « value » du premier bouton radio.

Remarque : `document.formulaire.radio.value` est une propriété qui n'existe pas (puisque `document.formulaire.radio` est un tableau). On ne peut pas savoir aussi simplement quel bouton radio est sélectionné.

Les gestionnaires d’événements

- Il est possible de prendre en compte avec JavaScript des événements associés à certaines balises. Ceci se fait en ajoutant des attributs de la forme :

```
onEventName="codeJavascript"
```

Exemple

```
<html>
<head>
	<title>Titre de la page</title>
	<script language="JavaScript">
		function afficher(chaine)
		{
			document.formulaire.msg.value = chaine;
		}
	</script>
</head>
<body>
	<form name="formulaire">
		<input type="text" name="msg">
		<br />
		<input type="button" value="Ping" onClick="afficher('Ping');">
		<input type="button" value="Pong" onClick="afficher('Pong');">
	</form>
</body>
</html>
```

- Nous avons introduit un formulaire avec un champ de texte et deux boutons. A chacun des boutons nous avons associé un gestionnaire d'événement avec l'attribut onClick. onClick permet d'indiquer une fonction JavaScript à employer quand l'utilisateur clique sur l'élément (ici, un bouton).
- Dans l'exemple, c'est la même fonction (le même gestionnaire d'événement) qui est appelé quand l'internaute clique sur l'un ou l'autre des boutons. C'est la chaîne de caractères passée en paramètre au gestionnaire d'événement qui change selon les boutons.
- Cliquer sur le bouton « Ping » affecte la valeur « Ping » au champ de texte, et réciproquement pour le bouton « Pong ».
- Remarquez que nous avons détourné l'utilisation « normale » d'un formulaire pour en faire un support pour une application dynamique.

Exemple de rollover

```
<html>
      <head>
         <title>Titre de la page</title>
      </head>
      <body>
         <a href="http://www.linux-france.org/"
   onMouseOver="document.getElementById('image').src='penguin2.jpg';"
  onMouseOut="document.getElementById('image').src='penguin1.jpg';">
         <img id="image" alt="Linux" src="penguin1.jpg">
         </a>
      </body>
</html>
```

- L’exemple ci-dessus crée un lien dont l’image change selon que le pointeur de la souris la survole ou non. Cet effet est appelé « rollover ».
- Quand le pointeur de la souris survole l’image servant de lien, l’événement `MouseOver` a lieu et son gestionnaire se déclenche (Ici, c’est une simple instruction JavaScript qui modifie l’image en manipulant le DOM).
- Quand le pointeur de la souris sort de l’image servant de lien, c’est le gestionnaire spécifié par la valeur de l’attribut `onMouseOut` qui se déclenche.

DHTML

- DHTML signifie Dynamic HTML. Il ne s'agit pas d'un langage ou d'une norme. Ce terme renvoie à l'utilisation de JavaScript pour manipuler le DOM et des feuilles de style afin de produire des effets « sophistiqués ».

Exemple d’animation de texte

```
<html>
<head>
<title>Animation</title>
<script language="JavaScript">
var pos= 0;
function min(x,y)
{
                    return (x < y) ? x : y;
}
var pos_max=min(window.screen.width,window.screen.height);
function animer()
{
                    if (pos <= pos_max)
                    {
                          document.getElementById("texte").style.position="absolute";
                          document.getElementById("texte").style.left=pos;
                          document.getElementById("texte").style.top=pos;
                          pos++;
                    }
                    window.setTimeout("animer();",1);
}
</script>
</head>
<body onLoad="animer();"> <h1 id="texte">MOUVEMENT</h1> </body> </html>
```

Exemple de modification de la couleur et du contenu d'un paragraphe en fonction du clic sur un bouton

```
<html>

<head>

      <title>Modification dynamique d'un paragraphe</title>

      <script language="JavaScript">

          var couleur = "blue";

          function afficher(chaine)

          {
             document.getElementById("paragraphe").firstChild.nodeValue =
   chaine;

             document.getElementById("paragraphe").style.color = couleur;

             couleur = (couleur == "blue") ? "red" : "blue";

          }

      </script>

</head>
```

```
    <body>
        <p id="paragraphe" style="color: green">C'est parti...</p>
        <form name="formulaire">
        <input type="button" value="Salut"
            onClick="afficher('Salut les amis !');">
        <input type="button" value="Au revoir"
            onClick="afficher('Au revoir les amis !');">
        </form>
    </body>
</html>
```

Au revoir les amis !