Cours JavaScript

Introduction (1)

- Javascript permet de rendre dynamique un site internet développé en HTML.
- Javascript permet de développer de véritables applications fonctionnant exclusivement dans le cadre d'Internet.
- Le Javascript est un langage de script simplifié orienté objet dont la syntaxe est basée sur celle du Java.
- Javascript a été initialement élaboré par Netscape en association avec Sun Microsystem.
  - Plus tard, Microsoft développera son propre langage Javascript officiellement connu sous le nom de *JScript*.

Introduction (2)

- Contrairement à un applet Java qui est un programme compilé, les scripts écrits en Javascript sont **interprétés**
  - Le Java, représenté par un ou plusieurs fichiers autonomes dont l'extension sera *.*class* ou *.*jar*, est invoqué par une balise HTML spécifique
  - Le JavaScript est **écrit directement au sein du document HTML** sous forme d'un script encadré par des balises HTML spéciales.
- Javascript est standardisé par un comité spécialisé, l'ECMA (European Computer Manufactures Association).

HTML et JavaScript

- **la page HTML devra TOUJOURS contenir les deux balises spécifiques et indispensables**

```
<script language="JavaScript">
.....
</script>
```

- Le code JavaScript s'intègre de deux manière avec le code HTML
  - 1. **Insertion directe** dans le code HTML
    - Le code JavaScript s'insère le plus souvent dans la page HTML elle même.
    - C'est la méthode la plus simple et la plus fréquemment utilisée par les développeurs de sites Internet.
  - 2. **Insertion comme un module externe**

1. Insertion dans une page HTML

- Il existe **2 manières** d'insérer du code JavaScript dans une page HTML
  - 1.1 Insertion pour exécution directe
    - Le code s'exécute automatiquement lors du chargement de la page HTML dans le navigateur en même temps que le contenu de la page HTML s'affiche à l'écran.
    - Le code JavaScript est placé dans le corps même de la page HTML, entre les balises **<body>** ..... Et ..... **</body>**

```
<html>
<head>
<title>..... </title>
</head>
<body>
<script
language="JavaScript">
alert('bonjour');
</script>
</body>

</html>
```

1.2 Insertion et exécution événementielle

- Le code javascript est d'abord lu par le navigateur, stocké en mémoire, pour ne s'exécuter que sur demande, lors du déclenchement d ’un événement.
- Le code JavaScript est placé dans le corps même de la page HTML, entre les balises **<head>** ..... et ..... **</head>**
- Le code s'exécutera seulement lors d'un événement généré par intervention de l'utilisateur.
  - Il faut écrire le code correspondant à cet événement dans le corps du document HTML.

```
<html>
<head>
<title> ..........
</title>
<script
language="JavaScript">
function auRevoir ()
        { alert(' Au
revoir');}
</script>
</head>
<body
onUnload=auRevoir();>
Excécution différée
</body>
</html>
```

2. Insertion par appel de module externe

- On peut insérer du code JavaScript en faisant appel à un module externe se trouvant à n'importe quelle adresse (URI).

```
<script src="URL du module externe">
.....
</script>
```

  - Les deux balises [illegible] ntre les Tags **<body>** et **</body>** dans le cas d'une exécution directe ou entre les Tags **<head>** et **</head>** de la page HTML pour une exécution différée.
  - Stocké dans un fichier sur le serveur à son adresse d'appel sous forme de **TEXTE SIMPLE** portant l'extension **.txt** ou **.js**
  - Simplifie la maintenance des sites faisant appel à des modules JavaScript communs à plusieurs pages HTML.
  - Inconvénient : l'appel au code externe génère une requête supplémentaire vers le serveur, et encombre le réseau

Entrée et sortie de données avec JavaScript

- 3 types de boites de messages peuvent être affichés en utilisant Javascript : **Alerte, Confirmation et Invite**
  - **Méthode alert()**
    - sert à afficher à l'utilisateur des informations simples de type texte. Une fois que ce dernier a lu le message, il doit cliquer sur OK pour faire disparaître la boîte
  - **Méthode confirm()**
    - permet à l'utilisateur de choisir entre les boutons OK et Annuler.
  - **Méthode prompt()**
    - La méthode prompt() permet à l'utilisateur de taper son propre message en réponse à la question posée
- La méthode **document.write** permet d 'écrire du code HTML dans la page WEB

Entrée et sortie de données avec JavaScript

```
<html>
<head>
<title> une page simple </title>
</head>
<body>
Bonjour
<script language='javascript'>
alert('bonjour');
document.write (
  prompt('quel est votre nom ?','Indiquer votre nom ici')
  );
confirm('quel bouton allez-vous choisir ?');
</script>
</body>
</html>
```

La structure d'un script en JavaScript

- La syntaxe du langage Javascript s'appuie sur le modèle de Java et C
- **Règles générales**
  1. L'insertion des espaces peut s'effectué n'importe où dans le script
  - Chaque commande doit être terminée par un point-virgule (;).
  - Un nombre à virgule est séparé par un point (.) et non par une virgule
  - Le langage Javascript y est **sensible à la casse**
  - Il existe deux méthodes permettant d'intégrer des commentaires à vos scripts.
    - Placer un double slash (//) devant le texte
    - Encadrer le texte par un slash suivi d'une étoile (/*) et la même séquence inversée (*/)

Les variables (1)

- **Déclaration et affectation**
  - Le mot-clé ***var*** permet de déclarer une ou plusieurs variables.
  - Après la déclaration de la variable, il est possible de lui affecter une valeur par l'intermédiaire du signe d'égalité (=).
  - Si une valeur est affectée à une variable sans que cette dernière ne soit déclarée, alors Javascript la déclare automatiquement.

```
//Déclaration de i, de j et de k.
var i, j, k;

 //Affectation de i.
i = 1;
//Déclaration et affectation de prix.
var prix = 0;

//Déclaration et affectation de
caractere
var caractere = ["a", "b", "c"];
```

Les variables (2)

- **Déclaration et affectation**
  - La lecture d'une variable non déclarée provoque une erreur
  - Une variable correctement déclarée mais dont aucune valeur n'est affectée, est indéfinie (undefined).
- **La portée**
  - les variables peuvent être globales ou locales.
  - Une variable globale est déclarée en début de script et est accessible à n'importe quel endroit du programme.
  - Une variable locale est déclarée à l'intérieur d'une fonction et n'est utilisable que dans la fonction elle-même.

Les variables (3)

- **Contraintes concernant les noms de variables**
  - Les noms de variables ne peuvent contenir que des lettres, chiffres, ou le caractère "_" (underscore)
    - *Mon_Prenom* est un nom valide
  - Les caractères spéciaux et accentués sont interdits (é, à, ç, ï, etc..)
    - *Mon_Prénom* n'est pas un nom valide. Il y a un caractère accentué.
  - Les majuscules et les minuscules ont leur importance.
    - *MonPrenom* est différent de *Monprenom*.
  - Un nom de variable ne peut contenir d'espaces.
    - *Mon Prenom* n'est pas un nom de variable correct. Il y a un espace.
  - Les mots réservés JavaScript ne peuvent être utilisés comme noms de variable.

Les variables (4)

- Le type d'une variable dépend de la valeur stockée dans cette variable. **Pas de déclaration de type.**
  - Exemple
    var maVariable = ' Philippe ';
    maVariable =10;

- trois principaux types de valeurs
  - String
  - Number : 10^-308 >nombre < 10^308
    - Les nombres entiers
    - les nombres décimaux en virgule flottant
    - 3 valeurs spéciales :
      - Positive Infinity ou +Infinity (valeur infini positive)
      - Negative Infinity ou –Infinity (valeur infinie négative)
      - Nan (Not a Number) habituellement générée comme résultat d'une opération mathamatique incohérente
  - Boolean
    - deux valeurs littérales : true (vrai) et false (faux).

Valeurs spéciales

- JavaScript inclut aussi deux types de données spéciaux :
  - **Null :** possède une seule valeur, **null**, qui signifie l'absence e données dans une variable
  - **Undefined :** possède une seule valeur, **undifined**. Une variable dont le contenu n'est pas clair car elle n'a jamais stocké de valeur, pas même **null** est dite non définie (undifined).

Les opérations sur les chaînes

- La concaténation
  - Var chaine = « bonjour » + « FI3/FCD1 »;
- Déterminer la longueur d'une chaîne
  - Var ch1 = « bonjour »;
  - Var longueur = ch1.length;
- Identifier le nième caractère d'une chaîne
  - Var ch1 =« Rebonjour ! »;
  - Var carac = ch1.charAt(2);
- Extraction d'une partie de la chaîne
  - Var dateDuJour = « 04/04/03 »
  - Var mois = datteDuJour.substring(3, 5);
    - 3: est l'indice du premier caractère de la sou-chaîne à extraire
    - 5 : indice du dernier caractère à prendre en considération ; ce caractère ne fera pas partie de la sous-chaîne à extraire

Les fonctions prédéfinies (1)

- **eval**
  - Cette fonction exécute un code Javascript à partir d'une chaîne de caractères.

```
 ...
<SCRIPT LANGUAGE="JavaScript">
function evaluation()  {
document.formulaire.calcul.value=eval(document.formulaire.saisie.value);  }
</SCRIPT>
...
<FORM NAME="formulaire">
Saisissez une expression mathématique : <INPUT TYPE="text" NAME=saisie MAXLENGTH=40 SIZE=40>
<INPUT TYPE="button" VALUE="evaluation." onClick="evaluation()">
<INPUT TYPE="text" NAME=calcul MAXLENGTH=40 SIZE=40>
</FORM>...
```

Les fonctions prédéfinies (2)

- **isFinite**
  - Détermine si le parametre est un nombre fini. Renvoie *false* si ce n'est pas un nombre ou l'infini positif ou infini négatif.

```
isFinite(240) //retourne true
isFinite("Un nombre") //retourne false
```

- **isN**
  - détermine si le parametre n'est pas un nombre (NaN : Not a Number).

```
isNaN("un nombre") //retourne true
isNaN(20) //retourne false
```

Les fonctions prédéfinies (3)

- **parseFloat**
  - analyse une chaîne de caractères et retourne un nombre décimal.
  - Si l'argument évalué n'est pas un nombre, renvoie *NaN* (Not a Number).

- **parseInt**
  - analyse une chaîne de caractères et retourne un nombre entier de la base spécifiée.
  - La base peut pren
    (binaire).

```
var numero="125";
var nombre=parseFloat(numero); //retourne le nombre 125
```

```
var prix=30.75;
var arrondi = parseInt(prix, 10); //retourne 30
```

Les fonctions prédéfinies (4)

- **Number**
  - convertit l'objet spécifié en valeur numérique

```
var jour = new Date("December 17, 1995 03:24:00");//converit la date en millisecondes
alert (Number(jour));
```

- **String**
  - convertit l'objet spécifié en chaîne de caractères

```
jour = new Date(430054663215);//Convertit le nombre en date Mois jour, Annéee etc.
alert (String(jour));
```

  - retourne la valeur hexadécimale à partir d'une chaîne de caractère codée en ISO-Latin-1.

```
escape("!&") //retourne %21%26%
```

Les Objets (1)

- Les objets de JavaScript, sont soit des entités pré définies du langage, soit créés par le programmeur.
  - Par exemple, le navigateur est un objet qui s'appelle "**navigator**".
  - La fenêtre du navigateur se nomme "**window**"
  - La page HTML est un autre objet, que l'on appelle "**document**".
  - Un formulaire à l'intérieur d'un "**document**", est aussi un objet.
  - Un lien hypertexte dans une page HTML, est encore un autre objet. Il s'appelle "**link**". etc…
- Les objets javascript peuvent réagir à des "Evénements".
- Tous les navigateurs ne supportent pas les mêmes objets
- Accès aux propriétés d'un objet
  - voiture.couleur.value
  - voiture.couleur.value = verte

Les objets (2)

- **L'opérateur New**
  - L'opérateur *new* est utilisé pour créer une nouvelle instance ou un nouveau type d'objet défini par l'utilisateur ou de l'un des types d'objets prédéfinis, *Array*, *Boolean*, *Date*, *Function*, *Image*, *Number*, *Object*, ou *String*.
    - **nouvel_objet = new type_objet(parametres)**

```
texte = new String("Une chaîne de caractère");
```

Les objets (3)

- **L'opérateur Typeof**
  - L'opérateur *typeof* renvoie une chaîne de caractères indiquant quel est le type de l'opérande.

```
var i = 1;
typeof i; //retourne number
var titre="Les raisins de la colère";
typeof titre; //retourne string
var jour = new Date();
typeof jour; //retourne object
var choix = true;       typeof choix; //retourne boolean
var cas = null;         typeof cas; //retourne object
typeof parseFloat; //retourne function
typeof Math; //retourne object (IE 5.*, NS 6.*, NS 4.78, Opera 6.*, Opera 5.*
typeof Math; //retourne function NS 3.*, Opera 3.*
```

L'objet *String (1)*

- Propriété :
  - *length* : retourne la longueur de la chaîne de caractères;
- Méthodes :
  - *anchor( )* : formate la chaîne avec la balise <A> nommée;
  - *b( )* : formate la chaîne avec la balise <B>;
  - *big( )* : formate la chaîne avec la balise <BIG>;
  - *charAt( )* : renvoie le caractère se trouvant à une certaine position;
  - *charCodeAt( )* : renvoie le code du caractère se trouvant à une certaine position;
  - *concat( )* : permet de concaténer 2 chaînes de caractères;
  - *fromCharCode( )* : renvoie le caractère associé au code;
  - *indexOf( )* : permet de trouver l'indice d'occurrence d'un caractère dans une chaîne;

L'objet *String (2)*

- *italics( )* : formate la chaîne avec la balise <I>;
- *lastIndexOf( )* : permet de trouver le dernier indice d'occurrence d'un caractère;
- *link( )* : formate la chaîne avec la balise <A> pour permettre de faire un lien;
- *slice( )* : retourne une portion de la chaîne;
- *substr( )* : retourne une portion de la chaîne;
- *substring( )* : retourne une portion de la chaîne;
- *toLowerCase( )* : permet de passer toute la chaîne en minuscule;
- *toUpperCase( )* : permet de passer toute la chaîne en majuscules;

L'objet *Array*

- Propriété :
  - *length* : retourne le nombre d'éléments du tableau;
- Méthodes :
  - *concat( )* : permet de concaténer 2 tableaux;
  - *join( )* : converti un tableau en chaîne de caractères;
  - *reverse( )* : inverse le classement des éléménts du tableau;
  - *slice( )* : retourne une section du tableau;
  - *sort( )* : permet le classement des éléments du tableau;

L'objet *Math (1)*

- Propriétés :
  - *E* : renvoie la valeur de la constante d'Euler (~2.718);
  - *LN2* : renvoie le logarithme népérien de 2 (~0.693);
  - *LN10* : renvoie le logarithme népérien de 10 (~2.302);
  - *LOG2E* : renvoie le logarithme en base 2 de e (~1.442);
  - *LOG10E* : renvoie le logarithme en base 10 de e (~0.434);
  - *PI* : renvoie la valeur du nombre pi (~3.14159);
  - *SQRT1_2* : renvoie 1 sur racine carrée de 2 (~0.707);
  - *SQRT2* : renvoie la racine carrée de 2 (~1.414);

L'objet *Math (2)*

- Méthodes :
  - *abs( ), exp( ), log(), sin( ), cos( ), tan( ), asin( ), acos( ), atan( ), max( ), min( ), sqrt( )* sont les opérations mathématiques habituelles;
  - *atan2( )* : retourne la valeur radian de l'angle entre l'axe des abscisses et un point;
  - *ceil( )* : retourne le plus petit entier supérieur à un nombre;
  - *floor( )* : retourne le plus grand entier inférieur à un nombre;
  - *pow( )* : retourne le résultat d'un nombre mis à une certaine puissance;
  - *random( )* : retourne un nombre aléatoire entre 0 et 1;
  - *round( )* : arrondi un nombre à l'entier le plus proche.

L'objet *Date (1)*

- Propriété : aucune;
- Méthodes :
  - *getFullYear( ), getYear( ), getMonth( ), getDay( ), getDate(), getHours( ), getMinutes( ), getSeconds( ), getMilliseconds( ):* retournent respectivement l'année complète, l'année (2chiffres), le mois, le jour de la semaine, le jour du mois, l'heure, les minutes, les secondes et les millisecondes stockés dans l'objet *Date*;
  - *getUTCFullYear( ), getUTCYear( ),* … retournent respectivement l'année complète, l'année (2chiffres), … stockés dans l'objet *Date* en temps universel;
  - *setFullYear( ), setYear( ),* … remplacent respectivement l'année complète, l'année (2chiffres), … dans l'objet *Date*;

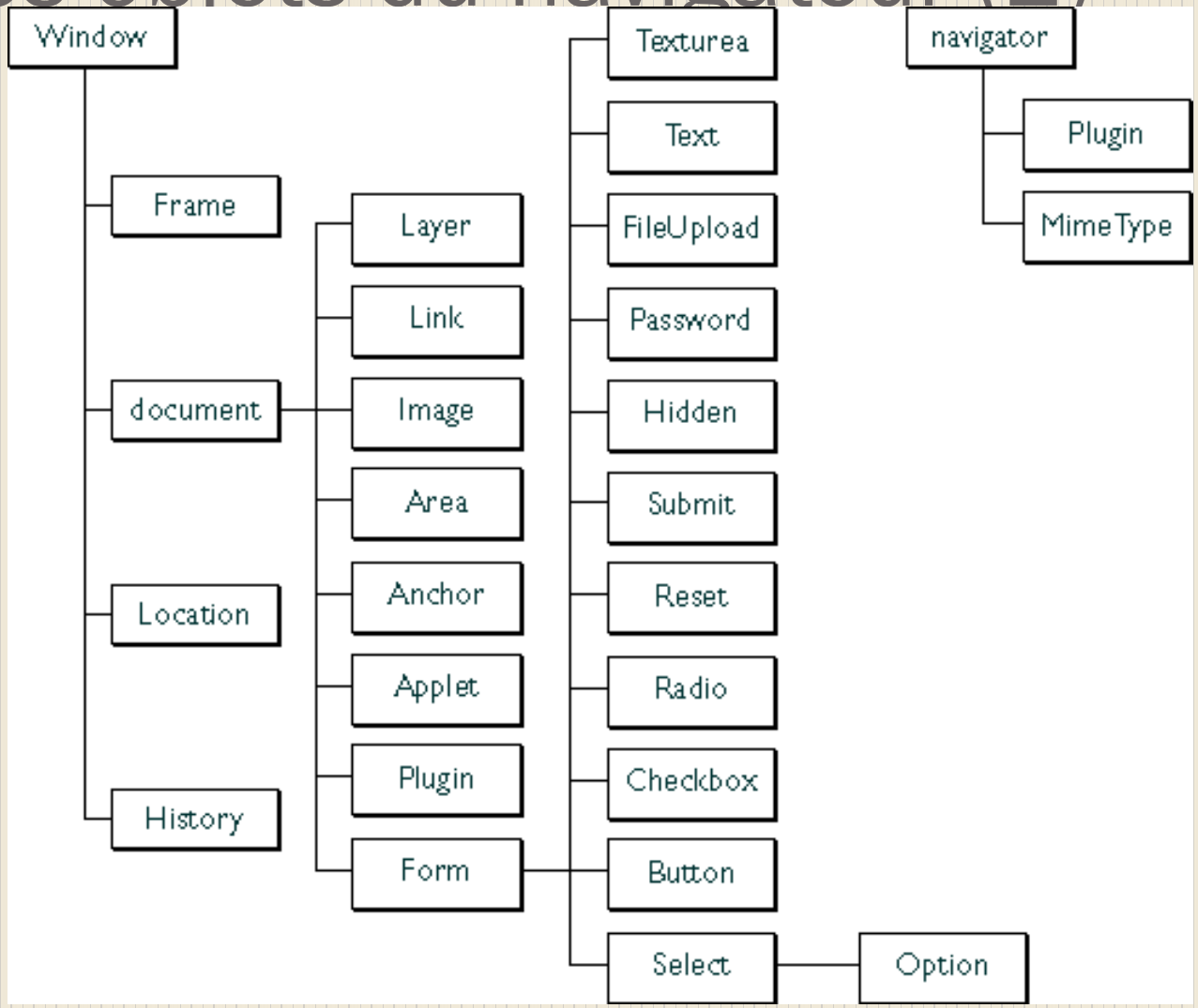
L'objet *Date (2)*

- *setUTCFullYear( ), setUTCYear( ),* … remplacent l'année complète, l'année (2chiffres), … dans l'objet *Date* en temps universel;
- *getTime( )* : retourne le temps stocké dans l'objet *Date*;
- *getTimezoneOffset( )* : retourne la différence entre l'heure du client et le temps universel;
- *toGMTString( ), toLocaleString( ), toUTCString( )* : convertissent la date en chaîne de caractère selon la convention GMT, selon la convention locale ou en temps universel;

Les objets du navigateur (1)

- L'objet le plus haut dans la hiérarchie est ***window*** qui correspond à la fenêtre même du navigateur.
- L'objet ***document*** fait référence au contenu de la fenêtre.
- ***document*** regroupe au sein de propriétés l'ensemble des éléments HTML présents sur la page. Pour atteindre ces différents éléments, nous utiliserons :
  - **soit des méthodes propres à l'objet** *document*, comme la méthode *getElementById( )*, qui permet de trouver l'élément en fonction de son identifiant (ID);
  - **soit des collections d'objets** qui regroupent sous forme de tableaux Javascript tous les éléments de type déterminé.

Les objets du navigateur (2)



L'objet *window (1)*

- Propriétés : (accessibles avec IE et N)
  - *closed* : indique que la fenêtre a été fermée;
  - *defaultStatus* : indique le message par défaut dans la barre de status;
  - *document* : retourne l'objet *document* de la fenêtre;
  - *frames* : retourne la collection de cadres dans la fenêtre;
  - *history* : retourne l'historique de la session de navigation;
  - *location* : retourne l'adresse actuellement visitée;
  - *name :* indique le nom de la fenêtre;

L'objet *window (2)*

- *navigator* : retourne le navigateur utilisé;
- *opener* : retourne l'objet *window* qui a créé la fenêtre en cours;
- *parent* : retourne l'objet *window* immédiatemment supérieur dans la hiérarchie;
- *self* : retourne l'objet window correspondant à la fenêtre en cours;
- *status* : indique le message affiché dans la barre de status;
- *top* : retourne l'objet *window* le plus haut dans la hiérarchie.

L'objet *window (3)*

- Méthodes :
  - *blur( )* : enlève le focus de la fenêtre;
  - *close( )* : ferme la fenêtre;
  - *focus( )* : place le focus sur la fenêtre;
  - *moveBy( )* : déplace d'une distance;
  - *moveTo( )* : déplace la fenêtre vers un point spécifié;
  - *open( )* : ouvre une nouvelle fenêtre;
  - *print( )* : imprime le contenu de la fenêtre;
  - *resizeBy( )* : redimensionne d'un certain rapport;
  - *resizeTo( )* : redimensionne la fenêtre;
  - *setTimeout( )* : évalue une chaîne de caractère après un certain laps de temps.

L'objet *document (1)*

- Propriétés :
  - *applets* : retourne la collection d'applets java présente dans le document;
  - *cookie* : permet de stocker un cookie;
  - *domain* : indique le nom de domaine du serveur ayant apporté le document;
  - *forms* : retourne la collection de formulaires présents dans le document;
  - *images* : retourne la collection d'images présentes dans le document;
  - *links* : retourne la collection de liens présents dans le document;

L'objet *document (2)*

- *referrer* : indique l'adresse de la page précédente;
- *title* : indique le titre du document.

- Méthodes :
  - *close( )* : ferme le document en écriture;
  - *open( )* : ouvre le document en écriture;
  - *write( )* : écrit dans le document;
  - *writeln( )* : écrit dans le document et effectue un retour à la ligne

L'objet *navigator*

- Propriétés
  - *appName* : application (Netscape, Internet Explorer)
  - *appVersion* : numero de version.
  - *platform* : système d'exploitation (Win32)
  - *plugins*
  - *language*
  - *mimeTypes*
  - *JavaEnabled()*

Les événements (1)

- Javascript est dépendant des événements
  - se produisent lors d'actions diverses sur les objets d'un document HTML.
    - onLoad;
    - onClick
    - onMouseover
    - onMouseout
    - …
- Il est possible de baser l'exécution de fonctions sur des événements

Les événements (2)

- **Evénement onLoad**
  - Se produit lorsque une page web est chargée dans la fenêtre du navigateur
  - Toute la page (y compris les images qu'elle contient si leur chargement est prévu) doit avoir été chargée pour qu'il ait lieu
  - Cet événement peut être associé à une image seulement ; auquel cas, il se produit une fois son chargement terminé

```
<HTML><BODY onLoad="alert('page chargée');">
Exemple de l'événement onLoad
</BODY></HTML>
```

Les événements (3)

- **Evénement onClick**
  - Se produit lorsque l'utilisateur clique sur un élément spécifique dans une page, comme un lien hypertexte, une image, un bouton, du texte, etc.
  - Ces éléments sont capables de répondre séparément à cet événement
  - Il peut également être déclenché lorsque l'utilisateur clique n'importe où sur la page s'il a été associé non pas à un élément spécifique, mais à l'élément body tout entier

```
<HTML><BODY>
<INPUT TYPE="Button" Value="cliquer ici"
onClick="alert('Clic')">
</BODY></HTML>
```

Les événements (4)

- **Événement onMouseover**
  - Analogue à onClick sauf qu'il suffit que l'utilisateur place le pointeur de sa souris sur l'un des éléments précités (lien hypertexte, image, bouton, texte, etc.) pour qu'il ait lieu
- **Événement onMouseout**
  - A l'inverse de onMouseover, cet événement se produit lorsque le pointeur de la souris quitte la zone de sélection d'un élément.

```
<HTML><BODY>
<IMG SRC="image.gif" onMouseOver="src='image2.gif';"
onMouseOut="src='image.gif';">
</BODY></HTML>
```

Nommage des objets-éléments

- Pour pouvoir manipuler un objet en javaScript, il doit posséder un nom
- Pour pouvoir distinguer les différents objets-éléments d'une page web, il suffit de leur donner un nom à travers de l'attribut NAME
  - <Table Name=« tableau1 »>…
  - <Table Name=« tableau2 »>…
  - <Form Name = « formulaire1 »>…
  - <Form Name =« formulaire2 »>…
  - <Textarea Name =« texte1»>…
- Dans le cas où l'objet serait unique alors pas besoin de nom pour désigner cet objet
  - Exemple : le cas de BODY(une seul BODY par document), DOCUMENT (un seul DOCUMENT par fenêtre)

Manipulation des objets

- Pour adresser un objet, il ne suffit pas de donner son nom : il faut aussi préciser son « chemin d'accès » dans l'arborescence de la structure

```
<HTML><BODY
onLoad="window.document.formulaire.zone.value='Bonjour';">
<FORM name="formulaire"><INPUT NAME="zone" TYPE="text">
</FORM></BODY></HTML>
```

courante

- Dans le cas de cadres (frames), il est pertinent de donner le nom de la fenêtre
- Il est possible aussi d'omettre window.document : l'adressage réussi puisqu'il n'y a qu'un seul objet « document» dans la fenêtre

Les Cookies (1)

- **Un "Cookie" est une chaîne de caractères qu'une page HTML (contenant du code JavaScript) peut écrire à un emplacement UNIQUE et bien défini sur le disque dur du client.**
  - Cette chaîne de caractères ne peut être lue que par le seul serveur qui l'a générée.
- **Que faire avec un cookie**
  - Transmettre des valeurs (contenu de variables) d'une page HTML à une autre.
    - Par exemple, créer un site marchand et constituer un "caddie" pour le client. Caddie qui restera sur son poste et vous permettra d'évaluer la facture finale au bout de la commande. Sans faire appel à quelque serveur que ce soit.
  - Personnaliser les pages présentées à l'utilisateur en reprenant par exemple son nom en haut de chaque page.

Les Cookies (2)

- Limitations lors de l'utilisation des cookies.
  - On ne peut pas écrire autant de cookies que l'on veut sur le poste de l'utilisateur (client d'une page). Il y a des limites :
    - Limites en nombre : Un seul serveur (ou domaine) ne peut pas être autorisé à écrire plus de 20 cookies.
    - Limites en taille : un cookie ne peut excéder 4 Ko.
    - Limites du poste client : Un poste client ne peut stocker plus de 300 cookies en tout.
  - Où sont stockés les cookies
    - En général, ils sont pour Netscape, dans le répertoire de l'utilisateur (si il y a des profils différents) sous le nom de "cookie.txt".
    - Microsoft Internet Explorer stocke les cookies dans des répertoires tels que "C:\WINDOWS\Cookies" ou encore "C:\WINDOWS\TEMP\Cookies".

Les Cookies (3)

- **Structure d'un cookie**
  - **Nom=Contenu; expires=expdate; path=Chemin; domain=NomDeDomaine; secure**
    - Nom=Contenu;
      - Sont deux variables suivies d'un ";" . Elles représentent l'en-tête du cookie.
      - La variable *Nom* contient le nom à donner au cookie.
      - La variable *Contenu* contient le contenu du cookie
      - Exemple ma_cookie=« oui:visite»

Les Cookies (4)

- **Expires= expdate;**
  - Le mot réservé **expires** suivi du signe "=" (égal). Derrière ce signe, vous mettrez une date d'expiration représentant la date à laquelle le cookie sera supprimé du disque dur du client.
  - La date d'expiration doit être au format :
    Wdy, DD-Mon-YYYY HH:MM:SS GMT
  - Utiliser les fonctions de l'objet **Date**
  - Règle générale : 'indiquer un délai en nombre de jours (ou d'années) avant disparition du Cookie.

Les Cookies (5)

- **path=Chemin;**
  - path représente le chemin de la page qui a créé le cookie.
- **domain=NomDeDomaine;**
  - domain représente le nom du domaine de cette même page
- **secure**
  - secure prend les valeurs "true" ou "false" selon que le cookie doit utiliser des protocoles HTTP simples (non sécurisés) ou HTTPS (sécurisés).
- Les arguments path, domain et secure sont facultatifs.
  - lorsque ces arguments sont omis, les valeurs par défaut sont prises.
  - Pour secure, la valeur est "False" par défaut.

Les Cookies (6)

- **Ecrire un cookie**
  - Un cookie est une propriété de l'objet document (la page HTML chargée dans le navigateur) alors l'intruction d'écriture de cookie est:
    - **document.cookie = Nom + "=" + Contenu + "; expires=" + expdate.toGMTString() ;**

```
var Nom = "MonCookie" ; // nom du cookie
var Contenu = "Hé... Vous avez un cookie sur votre disque !" ; // contenu du cookie
var expdate = new Date () ; // crée un objet date indispensable
puis rajoutons lui 10 jours d'existence :
expdate.setTime (expdate.getTime() + ( 10 * 24 * 60 * 60 * 1000)) ;
document.cookie = Nom + "=" + Contenu + "; expires=" + expdate.toGMTString() ;
```

Les Cookies (7)

- **Lecture d'un cookie**
  - Accéder à la propriété cookie de l'objet document.
  - **Document.cookie**

```
var LesCookies ; // pour voir les cookies
LesCookies = document.cookie ; // on met les cookies dans la variable LesCookies
```

- **Modification d'un cookie**
  - Modifier le contenu de la variable ***Contenu*** puis réécrire le cookie sur le disque dur du client

```
Contenu = "Le cookie a été modifié..." ; // nouveau contenu
document.cookie = Nom + "=" + Contenu + "; expires=" +
expdate.toGMTString() ; // écriture sur le disque
```

Les Cookies (8)

- **Suppression d'un cookie**
  - Positionner la date de péremption du cookie à une valeur inférieure à celle du moment où on l'écrit sur le disque.

```
// on enlève une seconde (ça suffit mais c'est nécessaire)
expdate.setTime (expdate.getTime() - (1000)) ;

// écriture sur le disque
document.cookie = Nom + "=" + Contenu + "; expires=" + expdate.toGMTString() ;
```