Université Sidi Mohamed Ben Abdllah

Faculté des sciences Dhar El Mahraz
Licence professionnelle SIGL

WEB DYNAMIQUE

JAVASCRIPT

RÉALISÉ PAR: PR. MAHRAZ MED ADNANE

ANNÉE UNIVERSITAIRE:2017/2018

INTRODUCTION

- JavaScript permet de dynamiser un site Web.
- Code JavaScript intégré aux pages HTML.
- Code interprété par le navigateur client ≠ code PHP (interprété du coté serveur).
- JavaScript est un langage événementiel (association d'actions aux événements déclenchés par l'utilisateur (passage de souris, clic, saisie clavier, etc...).

INTÉRÊTS DE JAVASCRIPT ?

- Supporté par les principaux navigateurs, c.-à-d., il ne nécessite pas de plug-in particulier.
- Accès aux objets contenus dans un document HTML
- Possibilité de mettre en place des animations sans l'inconvénient des longs temps de chargement nécessités par les données multimédia.
- Langage relativement sécurisé : il est impossible de lire ou d'écrire sur le disque client (impossibilité de récupérer un virus par ce biais).

INTÉGRATION DE JAVASCRIPT DANS HTML

Il existe **2 manières** pour insérer un code JavaScript dans une page HTML:

- JavaScript dans HTML

```
<html>
    <head>
        <title>Page HTML</title>
    </head>
    <body>
        <script>
            alert('bonjour');
        </script>
    </body>
</html>
```

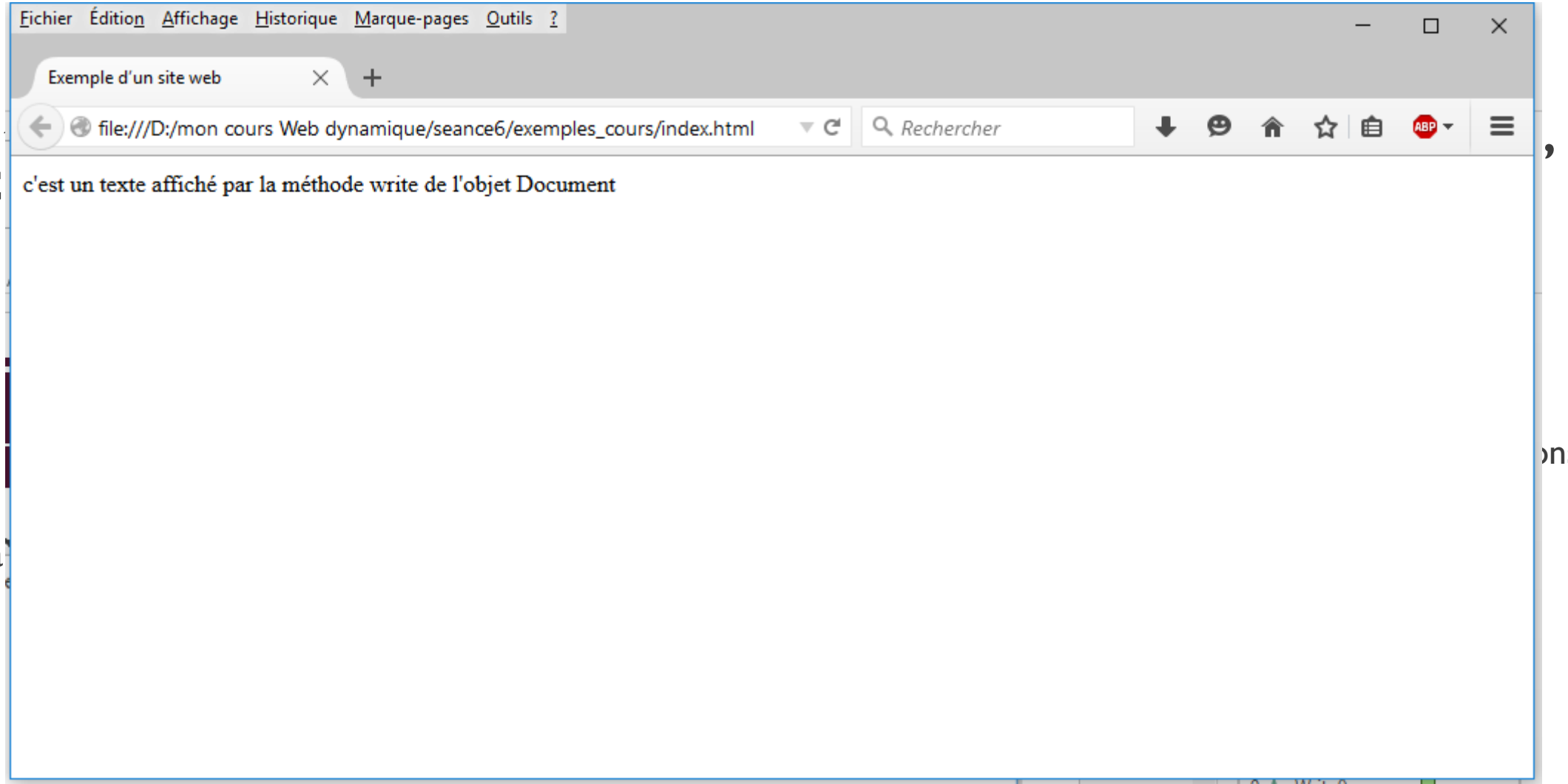
- JavaScript à l'extérieur du HTML

```
<html>
    <head>
        <title>Page HTML</title>
        <script src="monScript.js">
        </script>
    </head>
    <body>

    </body>
</html>
```

ENTRÉE ET SORTIE DE DONNÉES AVEC JAVASCRIPT

- 3
  C
  -
  -
  -
- La



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>
```

DÉCLARATION DE VARIABLES

- Utilisation de l'instruction var *variable=valeur;*
  - Pas de typage (détection automatique par l'interpréteur)
  - Nom de variable **sensible à la casse**.
  - Portée :
    - déclaration en dehors de fonction ⇒ globale
    - déclaration dans une fonction ⇒ locale

DÉCLARATION DE VARIABLES

```
<html>
<head>
     <title>Exemple</title>
</head>
<body>
     <script>
          var nom;
          var texte;
          nom=prompt("quel est votre nom");
          texte="je connais votre nom, c'est "+"<g><i>"+nom+"</g></i>";
          document.write(texte);
     </script>
</body>
</html>
```

STRUCTURES DE CONTRÔLE

Test conditionnel : if … else …

```
<html>
<head>
     <title>Exemple</title>
</head>
<body>
     <script>
                    var age=prompt("quel est votre age");
                    if (age>=18){
                      alert("vous etes Majeur...");
                    }else{
                      alert("vous etes Mineur...");
                    }
     </script>
</body>
</html>
```

STRUCTURES DE CONTRÔLE

- Boucle itérative :

```
<html>
<head>
     <title>Exemple</title>
</head>
<body>
     <script>
                    var nb=prompt("Donnez un nombre");
                    var somme=0;
                    for(var i=1;i<=nb;i++){
                       somme+=i;
                    }
                    alert("La somme des nombres entre 0 est "+ nb +" est "+somme);
     </script>
</body>
</html>
```

STRUCTURES DE CONTRÔLE

- Boucle conditionnelle

while(*condition*) { ... *instructions* ... }

```
<html>
      <head>
                  <title>Exemple</title>
      </head>
      <body>
                  <script>
                     var nb=prompt("Donnez un nombre");
                     var somme=0, i=0;
                     while(i<=nb){
                           somme+=i; i++;
                     }
                     alert("la somme des nombres entre 0 est "+nb+" est "+somme);
                  </script>
      </body>
</html>
```

FONCTIONS

- syntaxe :

```
function nom_fonction ([param1, …]){
        //Corps de la fonction
}
```

- Corps de la fonction
  - Déclaration des variables locales, propres à la fonction,
  - Instructions réalisés par la fonction,
  - Instruction **return** pour renvoyer une valeur ou un objet (Facultative)

FONCTIONS ET PORTÉE DES VARIABLES

- La portée d'une variable déclarée dépend de l'endroit où elle est déclarée :
  - VARIABLE GLOBALE: déclarée en dehors de la fonction.
  - VARIABLE LOCALE: déclarée à l'intérieur d'une fonction aura une portée limitée à cette seule fonction.

```
<script>
     var nom="Mohamed"; //variable globale
     function afficher(){
          alert("Votre nom est: "+nom);
     }
     alert("Votre nom est: "+nom);
</script>
```

```
<script>
    var nom="Mohamed"; //variable globale
    function saisir(){
        var nom; //variable locale
        nom=prompt("Quel est votre nom:");
        Window.alert(window.nom);
        return nom;
    }
    alert("Votre nom est: "+saisir());
    alert("Votre nom est: "+nom);
</script>
```

FONCTIONS ANONYMES

- syntaxe :

```
function ([param1, …]){
    //Corps de la fonction
}
```

```
Var x= function ([param1, …]){
    //Corps de la fonction
}
```

- On peut attribuer cette fonction à une variable.
- Pour que cette fonction s'appelle automatiquement:

```
(function(){//Corps de la fonction})();
```

- Isoler les variables déclarées au sein de la fonction au monde extérieur.

TABLEAU DE DONNÉES (ARRAY)

- Déclaration par l'utilisation de var.
- Le premier élément du tableau est indexé à 0.
- Il est possible de déclarer un tableau sans dimension fixée: Sa taille s'adapte en fonction du contenu.

```
   // création implicite d'un tableau
var mon_tableau = ["Ali", 'Mohamed', "Sarah", 10, 6];

   // création d'un tableau de 10 éléments
var mon_tableau = Array(10);

   // création d'un tableau avec l'opérateur « new »
var mon_tableau = new Array(10);
var mon_tableau = new Array();
```

UTILISATION DE TABLEAUX

- Accès aux éléments d'un tableau: Utilisation des crochets : [ ]

```
var tableau=new Array();
tableau[0]=10;
tableau[1]=5;
```

- La propriété Length

```
tableau.length
```

- Parcourir un tableau

```
// Parcourir un tableau sans connaître le nombre
d'éléments
var tableau= new Array(1, "a", 9) ;
tableau[200] = 12 ;
for (var i in tableau)
            alert("tableau[" + i + "] = "+tableau[i]);
```

TABLEAUX ASSOCIATIFS

L'indice est une chaîne de caractères

```
var tab=new Array();
tab["nom"]  ="Ben ali";
tab["prenom"]   ="Mohamed";
tab["age"]  =25;
tab["adresse"]  ="Fes";
...
```

```
...
alert("Votre nom est: "+tab["nom"]);
...
```

La propriété Length de l'objet Array() pour ce genre de tableau ne fonctionne pas.

TABLEAUX MULTI-DIMENSIONNELS

Array permet de stocker des objets, donc des tableaux.

```
...
var row0=new Array();
var row1=new Array();
var row2=new Array();
var morpion=new Array();
morpion[0]=row0; morpion[1]=row1; morpion[2]=row2;
...
morpion[1][2]="X";
...
```

O	X	
X	O	X
	O	X

DÉCLARATION ET CRÉATION D'OBJETS

- Deux types d'objets
  - Objets prédéfinis
  - Objets propres
- Création d'objets avec des initialiseurs d'objets (objets littéraux).

```
var obj = {
   propriété_1: valeur_1,
   propriété_2: valeur_2,
   …
   propriété_n: valeur_n
};
```

```
var obj = new Object();
obj.popriété_1=valeur_1;
obj.propriété_2=valeur_2,
   …
obj.propriété_n= valeur_n;
```

DÉCLARATION ET CRÉATION D'OBJETS

- Les objets sont créés de la même façon qu'avec new Object().
- les objets créés à partir d'une expression littérale seront des instances d'Object.
- Les objets peuvent également être créés en utilisant la méthode Object.create().

```
var Animal = {
type: "Invertébrés", // Valeur par défaut
afficherType : function() { // Une méthode pour afficher le type Animal
console.log(this.type); }
}
// On crée un nouveau type d'animal, animal1
var animal1 = Object.create(Animal);
animal1.afficherType(); // affichera Invertébrés
// On crée un nouveau type d'animal, animal2
var animal2 = Object.create(Animal);
Anim12["type"]= "poisson";
animal2.afficherType(); // affichera poisson
```

DÉCLARATION ET CRÉATION D'OBJETS

Création d'objets propres en utilisant un constructeur

- Par appel d'une fonction qui va créer les propriétés de l'objet.
- Utilisation de **this** pour faire référence à l'objet courant
- On crée une instance de l'objet avec **new**.

```
var Etudiant=new Etudiant("Mohamed", "Ben Ali", "1298742046");
function Etudiant(Le_nom,Le_prenom,Le_CNE){
            this.nom=Le_nom;
            this.prenom=Le_prenom;
            this.CNE=Le_CNE;
}
alert("Votre nom est: "+Etudiant.prenom);
```

DÉCLARATION ET CRÉATION D'OBJETS

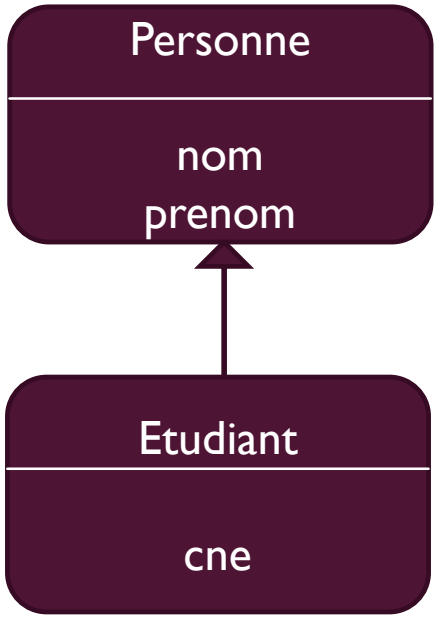
- Déclaration de méthodes
  - Association de fonctions dans la création de l'objet.

```
function Etudiant(Le_nom,Le_prenom,Le_CNE){
            this.nom=Le_nom;
            this.prenom=Le_prenom;
            this.CNE=Le_CNE;
            this.afficher=affiche_Etudiant;
}
function affiche_Etudiant(){
            document.write("Votre nom et prénom est: "+ Etudiant.nom+"
            "+Etudiant.prenom+"<br/>" );
            document.write("Votre CNE est: "+ Etudiant.CNE );
}
var Etudiant=new Etudiant("Mohamed", "Ben Ali", "1298742046");
Etudiant.afficher();
```

LE MODÈLE OBJET JAVASCRIPT

- JavaScript est un langage objet basé sur des prototypes et non pas sur des classes.
- JavaScript possède que des objets.
- Les objets *prototypiques* agissent comme un modèle sur lequel on pourrait obtenir des propriétés initiales pour un nouvel objet.
- Un objet peut être associé comme le *prototype* d'un autre objet (le second objet partage les propriétés du premier).

LE MODÈLE OBJET JAVASCRIPT




```
function Personne(){
        this.nom="Mahraz";
        this.prenom="Mohamed";
}
function Etudiant(){
        this.CNE="22222222";
}
```

```
Etudiant.prototype=new Personne()
Var p1=new Personne();
var e1=new Etudiant();
e1.__proto__.nom="toto";
Etudiant.prototype.nom="dodo";
```

On peut modifier la propriété nom du constructeur Etudiant pour que les objets instanciés changent leurs valeurs

OBJETS PRÉDÉFINIS

- Plusieurs objets prédéfinis en JavaScript:
  - *Array*, *Boolean*, *Date*, *Function*, *Image*, *Number*, *Object*, ou *String*.
- **L'opérateur Typeof**
  - *L'opérateur typeof renvoie une chaîne de caractères indiquant quel est le type de l'opérande.*

```
var titre="Les raisins de la colère";
typeof titre; //retourne string
```

L'OBJET STRING

L'objet **String** permet de manipuler les chaînes de caractères

- **Propriété** :
  - **length** : retourne la longueur de la chaîne de caractères;
- **Méthodes** :

1. Opérations sur les chaînes

- **concat(str)** : retourne la chaîne concaténée avec **str**
- **split(str)** : retourne, sous forme de tableau, les portions de la chaînes délimitées par **str**
- **substring(debut,fin)** : extrait une sous-chaîne, depuis la position **debut** (incluse) à **fin** (exclusive).
- **substr(debut,i)** : extrait une sous-chaîne, depuis la position **debut**, en prenant **i** caractères

L'OBJET STRING

2. Opérations sur les caractères

- **charAt(i)** : retourne le **i**ième caractère
- **indexOf(str)** : retourne la position de **str** dans la chaîne (**-1** si elle n'est pas trouvée)
- **lastIndexOf(str)** : idem, mais renvoie la position de la dernière occurrence de **str**
- **toLowerCase()** : retourne la chaîne en minuscules
- **toUpperCase()** : retourne la chaîne en majuscules

L'OBJET ARRAY

L'objet **Array** permet de créer et manipuler des tableaux.

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

L'OBJET MATH

- 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

- 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 IMAGES

- **Propriétés**
  - complete
  - width
  - height
  - src
- **Méthodes**
  - constructeur
    - Image()
    - Image(largeur, hauteur)

```
// Exemple
img = new Image() ;
img.src = 'image.gif' ; // Préchargement
img.onload = function(){
   // Modification de la 13e image de la page Web
        document.images[12].src = img.src ;
}
```

L'OBJET DATE

- Méthodes
  - Constructeur
  - getDay(), attention de 0 (dimanche) à 6 (samedi)…
  - getDate() / setDate()
  - getMonth() / setMonth(), attention de 0 à 11…
  - getYear() / setYear() / getFullYear() / setFullYear()
  - getHours()/ setHours()
  - getMinutes() / setMinutes()
  - getTime() / setTime()

```
var jour = new Date();
alert(jour.getFullYear());
// 2011

var anniversaire= new Date(2015, 10, 25);
alert(anniversaire.toLocaleString());
// dimanche 25 octobre 2015 00:00
```

FONCTIONS SUPÉRIEURES

- **`eval(`*`chaine`*`)`**
- **`isFinite(`*`nombre`*`)`**
- **`isNaN(`*`objet`*`)`**
- **`parseFloat(`*`chaine`*`)`**
- **`parseInt(`*`chaine`*`)`**

FONCTIONS SUPÉRIEURES

➔ 25

document.write(isFinite(Math.log(0))) ;

➔ false

document.write(isNaN("abcd")) ;

➔ true

document.write("12.34"+2) ;

➔ 12.342

document.write(parseFloat("12.34")+2) ;

➔ 14.34

DÉCLENCHEMENT D'INSTRUCTIONS JAVASCRIPT

- Programmation événementielle
  - JavaScript = langage réactif
  - L'interaction avec l'utilisateur est gérée via des événements
  - Événement = tout changement d'état du navigateur

DÉCLENCHEMENT D'INSTRUCTIONS JAVASCRIPT

- Événements JavaScript
  - blur : le focus est enlevé d'un objet
  - focus : le focus est donné à un objet
  - change : la valeur d'un champ de formulaire à été modifiée par l'utilisateur
  - mouseover : la souris est déplacée sur un objet
  - click : un clic souris est déclenché sur un objet
  - select : un champ de formulaire est sélectionné (par tabulation)
  - submit : un formulaire est soumis
  - load : la page est chargée par le navigateur
  - unload : l'utilisateur quitte la page

DÉCLENCHEMENT D'INSTRUCTIONS JAVASCRIPT

Il est possible de baser l'exécution de fonctions sur des événements

Événements détectables

- Nom de l'événement précédé de on :
  onBlur, onChange, onClick, onFocus, onLoad, onMouseover, onSelect, onSubmit, onUnload

Association événement - action

- Dans le code HTML, identique à la déclaration d 'une propriété :
- <nom_élément attribut$_i$ = propriété$_i$ événement$_j$ = "action$_j$" >

DÉCLENCHEMENT D'INSTRUCTIONS JAVASCRIPT

```
<html>
    <head>
        <title>Exemples de déclenchements</title>
        <script>
            function saluer() {
                alert("Bonjour tout le monde");
            }
        </script>
    </head>
    <body onLoad="saluer()">
        <h1 onMouseover="saluer()">Survoler le pointeur pour exécuter l'événement</h1>
        <form>
            <input type="button" name="bouton"  value="salut" onClick="saluer()">
        </form>
        <h1>Exécution sur protocole javascript:</h1>
        <a href="javascript:saluer()">pour saluer</a>
    </body>
</html>
```

CHANGER L'ASPECT DU FORMULAIRE

Ecrire une page HTML contenant un formulaire (deux zones de texte et le bouton envoyer). La bordure de la zone du texte est changée en vert s'elle est sélectionnée, sinon, elle devient en gris.

CHANGER L'ASPECT DU FORMULAIRE

```
<form>
    <input type="text" value="" name="texte1" onBlur="unchanger(this)" onFocus="changer(this)"/>
    <input type="text" value="" name="texte2" onBlur="unchanger(this)" onFocus ="changer(this)"/>
    <input type="submit"/>
</form>
<script>
    var changer=function(texte){
        texte.style.border="2px solid green";
    }
    var unchanger=function(texte){
         texte.style.border="";
    }
</script>
```

CONTRÔLE DU FORMULAIRE

Ecrire une page HTML contenant un formulaire (zone de texte et le bouton envoyer). Un message d'erreur est affiché si la zone de texte est vide au moment de la soumission.

CONTRÔLE DU FORMULAIRE

```
<form onSubmit="return verifier()">
    <input type="text" name="texte" value="" name="texte1" />
    <input type="submit" />
</form>
<script>
    var verifier=function(){
        if (document.forms[0].elements["texte"].value==""){
            alert("zone vide");
            return false;
        }
    }
</script>
```

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" : Il permet de spécifier que le *cookie* sera envoyé uniquement si la connexion est sécurisée 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'instruction d'écriture de cookie est:
    - **document.cookie = Nom + "=" + Contenu + "; expires=" + expdate.toUTCString() ;**

```
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.toUTCString() ;
```

LES COOKIES (7)

- **Lecture d'un cookie**

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

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

- **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() ;
```