Langage Javascript

Professeur Mostafa JEBBAR
Laboratoire C3S
Ecole Supérieure de Technologie, Université Hassan II de Casablanca, Maroc
Ex Chef de Département de la Recherche Scientifique de l'Université Hassan II de Casablanca
Email : mostafajebbar@gmail.com
https://fr.slideshare.net/mostafaJebbar
Chaîne vidéo : https://youtube.com/user/mostafajebbar
Recherche : https://www.researchgate.net/profile/Mostafa-Jebbar/research

Plan

- Gestion des minuteries
- Le modèle DOM
- Gestion des événements

Gestion des Minuteries

Le découpage du temps
Les minuteries

Déclenchement des actions au bout d'un certain laps de temps

- **setInterval() :**
  - **Déclenchement d'une action répétitive périodique et régulière**
  - **L'appel de la méthode renvoie un identificateur unique**
  - **L'arrêt de l'action est produit par appel de la méthode clearInterval() ou par la destruction de la fenêtre.**

Syntaxe :

setInterval(action,période[,arg1[arg2…argn]]);

action : expression javascript (formule de calcul, appel de fonction)
période : période de déclenchement de l'action(en millisecondes)
arg1,arg2,… : arguments de la fonction

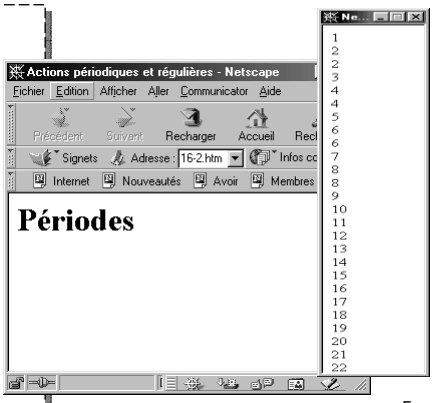
Le découpage du temps
Les minuteries

- **clearInterval() :**
  - **Arrête une minuterie lancée par un appel à setInterval()**

  syntaxe : clearInterval(identificateur de minuterie);

  identificateur de minuterie : identificateur renvoyé lors de la création de la minuterie

```
<HTML><HEAD>
<TITLE>Actions périodiques et
régulières</TITLE>
<script language="JavaScript">
test = 1;
w = open("", "","height=500,width=100");
function affiche()
{ w.document.write(test++ + "<BR>");
  if (test > 10) clearInterval(minut1);
}
setInterval("affiche()", 700, test);
minut1 = setInterval("--test", 2000);
</script></HEAD>
<BODY onUnload w.close()>
<H1>Périodes</H1>
</BODY></HTML>
```



Le découpage du temps
Les minuteries

- **setTimeout()** :
  - Évalue une fonction ou une expression pendant un délai indiqué comme paramètre
  - L'appel de la méthode renvoie un identificateur unique.

  syntaxe :

  setTimeout(action,délai[,arg1[arg2…argn]]);

  action : expression javascript (formule de calcul, appel de fonction)

  délai : délai au bout duquel sera évaluée l'action (en millisecondes)

  arg1,arg2,… : arguments de la fonction

Le découpage du temps
Les minuteries

- **clearTimeout() :**
  - Arrête une minuterie lancée par un appel à setTimeout() avant l'expiration du délai fixé

  syntaxe : clearTimeout(identificateur de minuterie);

  identificateur de minuterie : identificateur renvoyé lors de la création de la minuterie

```
<HTML><HEAD><TITLE>Lancement de deux minuteries et arrêt de l'une d'elles</TITLE>
<script language="JavaScript">
resultat = 252;
function maFonction(){
resultat = 252252
document.forms[0].Champ2.value = "Après 3 secondes : " + resultat;}
</script> </HEAD>
<BODY onLoad="seconde = setTimeout('maFonction()', 5000)">
<FORM>
<INPUT TYPE="text" NAME="Champ1" SIZE=30>
<INPUT TYPE="text" NAME="Champ2" SIZE=30>
<INPUT TYPE="button" VALUE="Cliquez ici pour arrêter" onClick=clearTimeout(seconde)>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
document.forms[0].Champ1.value = "Immédiatement : " + resultat;
</SCRIPT></BODY></HTML>
```


Le Modèle DOM

DOM & Javascript

- Document Object Model, standard du W3C
- Une API permettant de manipuler les différents éléments d'une page web grâce à des **objets, propriétés et méthodes** standardisés.
- Couplé Javascript, il permet de gérer tout **l'aspect évènementiel** d'un document.
- DOM fournit :
  - **Une représentation structurée sous forme d'une arborescence hiérarchisée d'objets (où chaque balise de la page Web est lui associée un objet dans l'arborescence)**
  - La manière dont un script peut accéder à cette structure

DOM & Javascript

- Les "Document Object Models" (DOMs) définissent une série de méthodes et propriétés avec lesquelles on peut interroger et modifier n'importe quel élément d'un document HTML ou XML.
- Un DOM est une description structurée sous forme d'**arbre** d'un document Web (HTML ou XML)
- Un DOM permet d'offrir **une interface qui modifie la structure**, le contenu et le style d'un document Web
- Un DOM est un arbre avec des nœuds. Les nœuds de cet arbre peuvent être de types différents, les plus importants sont :
  - Un **noeud élément** associé à une balise dans le document Web
  - Un **noeud attribut** désignant un attribut dans un document Web
  - Un **noeud texte** désignant le contenu d'un élément ou d'un attribut dans un document web
- Chaque **nœud est un objet** ayant des méthodes et des propriétés
- Ces interfaces sont implémentées pour plusieurs langages : javascript, java, …

DOM & Javascript : les Nœuds

- Exemple :

```
<DIV id="Div0" style = "background-color:#FFCC33; width:400">
<CENTER> javascript et Dom <BR>
<IMG name="image01"src="images/photo_printemps.jpg"> <BR>
Image du printemps
</CENTER>
</DIV>
```

  - **DIV, IMG** : sont des nœuds de type élément
  - **style, SRC** : sont des nœuds de type attribut
  - **Background-color, javascript et Dom , images/photo_printemps.jpg**, **Image du printemps** : sont des noeuds de type texte
  - Javascript et Dom, Image du printemps, IMG sont des trois **noeuds enfant** de CENTER
  - style est un **noeud associé** au noeud DIV
  - SRC est un **noeud associé** au noeud IMG
  - IMG ne dispose pas de noeud enfant

Accès aux éléments

- Accès aux éléments, il existe 3 méthodes

  - **getElementById("id")** : retourne un objet HTML à partir de son id, défini dans la propriété id de la balise de l'objet ;

  - **getElementsByName("nom")** : retourne un tableau d'objets HTML ayant nom défini dans la propriété name de la balise de l'objet;

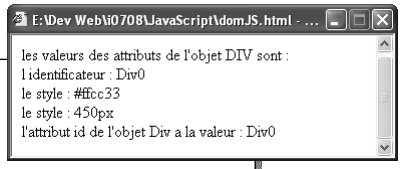
  - **getElementsByTagName()** : retourne un tableau d'objets HTML à partir de nom de la balise. Il existe déjà des tableaux définis pour certains types d'objets par exemple :
    - document.getElementsByTagName("IMG")=document.images
    - document.getElementsByTagName("llinks")=document.links
    - document.getElementsByTagName("script")=document.scripts

Accès aux éléments

```
<SCRIPT LANGUAGE="JavaScript">
obj1=document.getElementById("Div0")
win01=open("","","width=380,height=420")
win01.document.write(obj1.innerHTML)

win01.document.write("<hr>")
T=document.getElementsByName('image0')
win01.document.write("l'URL de l'image est :
" + T[0].src)

win01.document.write("<hr>")
T2=document.getElementsByTagName("BR")
win01.document.write("Le nombre de balise
BR dans le document est : " + T2.length)
</SCRIPT>
```

Accès aux attributs

- **getAttribute**("nomAtt") : permet de retourner la valeur d'un attribut de nom donné si l'attribut n'est pas un objet. Si l'attribut n'existe pas, la valeur renvoyée sera soit null soit "" (une chaîne vide)
- **attributes("**nomElt**")** : retourne un tableau d'attributs un élément




```
obj=document.getElementById("Div0")
valAttId=obj.getAttribute("id")
valAttStyle=obj.getAttribute("style")

win01=open("","","width=380,height=420,scrollbars=yes")
win01.document.write("les valeurs des attributs de l'objet " +
obj.tagName + " sont : <BR>")

win01.document.write("l identificateur : " + valAttId +" <BR>")
win01.document.write("le style : " + valAttStyle.backgroundColor+"
<BR>")
win01.document.write("le style : " + valAttStyle.width+" <BR>")

win01.document.write("l'attribut " + obj.attributes[18].name+ " de l'objet
Div a la valeur : " + obj.attributes[18].value+" <BR>")
```

Informations sur les nœuds

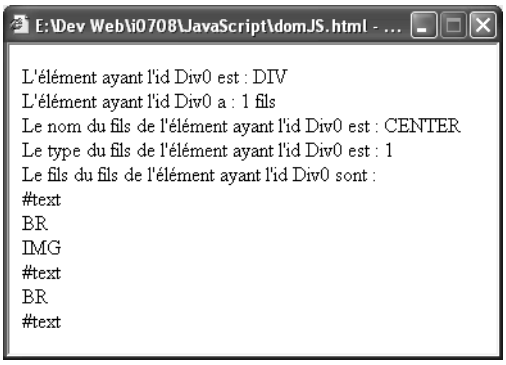
- Informations sur un nœud
  - **nodeName** : renvoie le nom de l'élément
  - **nodeType** : renvoie le type de nœud (1 : élément, 2 : attribut, 3 : texte, ….)
  - **nodeValue** : Texte contenu dans le nœud de type texte
  - **hasChildNodes** : retourne si un nœud à des éléments enfants
  - **hasAttributes()** : méthode qui retourne une valeur booléenne si un nœud à des attributs : nomElt.attributes.length > 0)
  - **hasAttribute()** : méthode qui retourne une valeur booléenne qui indique si l'attribut existe ou pas.

Informations sur les nœuds

- **Exemple**

```
obj1=document.getElementById("Div0")
win01=open("","","width=380,height=240")
win01.document.write("L'élément ayant l'id Div0 est : " + obj1.nodeName
+"<BR>")
if(obj1.hasChildNodes()){
win01.document.write("L'élément ayant l'id Div0 a : " +
obj1.childNodes.length+" fils<BR>")
win01.document.write("Le nom du fils de l'élément ayant l'id Div0 est : "
+ obj1.childNodes[0].nodeName+"<BR>")
win01.document.write("Le type du fils de l'élément ayant l'id Div0 est : " +
obj1.childNodes[0].nodeType+"<BR>")
obj2=obj1.childNodes[0]
lesFilsObj2=""
for(i=0;i<obj2.childNodes.length;i++){
lesFilsObj2+=obj2.childNodes[i].nodeName
lesFilsObj2+="<BR>"
}
win01.document.write("Le fils du fils de l'élément ayant l'id Div0 sont :
<BR> " + lesFilsObj2)
}
```

Informations sur les nœuds



Accès aux enfants d'un élément

- Accès aux enfants :
  - nomElt.**childNodes** tableau donnant la liste de tous les nœuds-enfants de l'élément nomElt.
  - nomElt.**firstChild** objet retornant le premier nœud-enfant de l'élément nomElt (~ nomElt.childNodes[0]).
  - nomElt.**lastChild** objet retournant le dernier nœud-enfant de l'élément nomElt.

```
obj1=document.getElementById("Div0")
win01=open("","","width=380,height=240")
win01.document.write("L'élément ayant l'id Div0 est : " + obj1.nodeName +"<BR>")
obj2=obj1.firstChild
win01.document.write("Le premier fils de l'élément CENTER est : " +
obj2.firstChild.nodeName)
}
```

Accès aux nœuds ancêtres/frères

- Accès aux nœuds ancêtres :
  - **parentNode** : permet de remonter d'un cran dans la hiérarchie et se positionner au nœud père de l'élément
- Accès aux noeuds frères :
  - **previousSibling** : permet d'accéder au frère précédent ;
  - **nextSibling** : permet d'accéder aux frère suivant ;

```
obj1=document.getElementById("Div0")
win01=open("","","width=380,height=240")
win01.document.write("L'élément ayant l'id Div0 est : " + obj1.nodeName +"<BR>")

obj2=obj1.firstChild
win01.document.write("Le pére de l'élément " + obj2.nodeName + " est : " +
obj2.parentNode.nodeName + "<BR>")

obj3=obj2.firstChild
obj3Frere=obj3.nextSibling
win01.document.write("Le frére suivant de l'élément " + obj3.nodeName + " est : " +
obj3Frere.nodeName)
```

Accès aux enfants par collections

- Accès par collections: aux enfants :
  - **window.frames** : la liste des frames d'un document
  - **window.document.forms** : liste des formulaires présents dans un document
  - **window.images** : liste des images présentes dans un document
  - **window.links** : liste des liens présents dans un document

Modification du style

- Modification du style d'un élément :
  - En changeant la valeur de l'identifiant ou bien la valeur de la classe d'un élément, en rapport avec une feuille de style CSS. Les propriétés correspondantes aux attributs id et class sont respectivement **id** et **classNam**e
  - En faisant appel à la collection style nomElt.**style.nomPropriete** (ex. monDiv.style.zIndex=2)
  - En faisant appel à la méthode **setAttribute** (ex. monDiv.setAttribute ("z-Index", "2").

Modification du contenu

- Modification du contenu d'un élément :
  - **innerHTML()** : permet de modifier le contenu d'un élément donné (ex : monDiv.innerHTML = " <B>Ceci est un texte qui sera affecté à monDiv en style gras<B> ")
  - **nodeValue**: permet d'accéder à la valeur d'un nœud donné dépendamment de son type
    - attribut ⓪ Valeur de l'attribut
    - texte ⓪ Contenu du nœud texte
    - document ⓪ null
    - element ⓪ null

Modification de la structure

- Création de nœuds :
  - **createElement()** : permet de créer un élément (ex: monDiv02 = document.createElement ("div") ~)
  - **createTextNode()** : permet de créer un nœud de type texte (ex. monParagraphe = document.createTextNode ("texte de mon paragraphe")

- Affectation d'un noeud à un autre :
  - **appendChild()** : permet d'affecter un nœud à un autre (ex. monDiv01.appendChild(monDiv02) ~)
  - **createAttribute**() : permet de créer et d'affecter un attribut à un élément (ex. idMonID02 = monDiv02.createAttribute("id"); idMonID02.value ="monDiv02" ~)
  - **setAttributeNode()** : Ajoute un nouvel attribut. Si un attribut avec un même nom est déjà présent, il est remplacer par le nouveau (ex.

Modification de la structure

- Exemple

```
obj1=document.getElementById("Div0")
win01=open("","","width=380,height=440,scrollbars=yes")

oTable=document.createElement("TABLE")
oTable.setAttribute('border','1');
oTable.setAttribute('width','200');

//création de la première ligne du tableau
oTR1=document.createElement("TR")
oTR1.setAttribute('bgcolor','green');

oTD11=document.createElement("TD")
oTEXT11=document.createTextNode("Nom Image")
oTD11.appendChild(oTEXT11)
oTR1.appendChild(oTD11)

oTD12=document.createElement("TD")
oTEXT12=document.createTextNode("Type")
oTD12.appendChild(oTEXT12)
oTR1.appendChild(oTD12)

oTable.appendChild(oTR1)
```

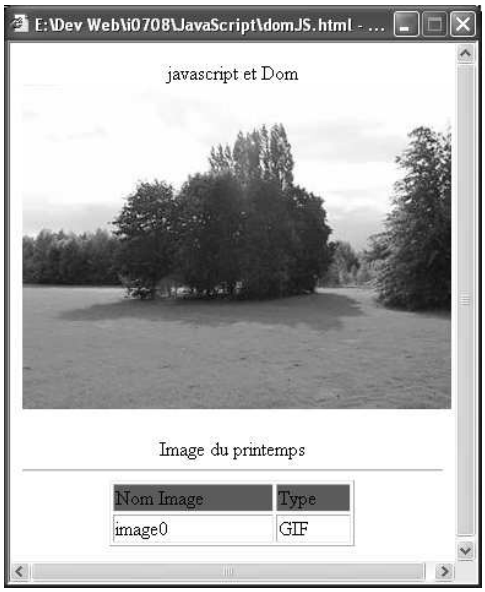
Modification de la structure

- Exemple

```
//création de la deuxième ligne du tableau
oTR2=document.createElement("TR")
oTD21=document.createElement("TD")
oTEXT21=document.createTextNode("image0")
oTD21.appendChild(oTEXT21)
oTR2.appendChild(oTD21)

oTD22=document.createElement("TD")
oTEXT22=document.createTextNode("GIF")
oTD22.appendChild(oTEXT22)
oTR2.appendChild(oTD22)

oTable.appendChild(oTR2)

oHR=document.createElement("HR");


obj1.firstChild.appendChild(oHR)
obj1.firstChild.appendChild(oTable)

win01.document.write(obj1.innerHTML)
```


Modification de la structure

- Exemple



Modification de la structure

- Création de nœuds :
  - **insertBefore()** : permet d'inseret un nœud enfant avant un autre nœud enfant (ex. monDiv01.insertBefore(monParagraphe, , monDiv02) ~ texte de mon paragraphe)
  - **cloneNode(booléen)** : Établit une copie identique d'un nœud, avec (booléen=true) ou sans (booléen=false) la structure de sous-nœuds qui en fait partie (ex: monDiv03=monDiv01.cloneNode(true) ~mondiv03 à la valeur texte de mon paragraphe)

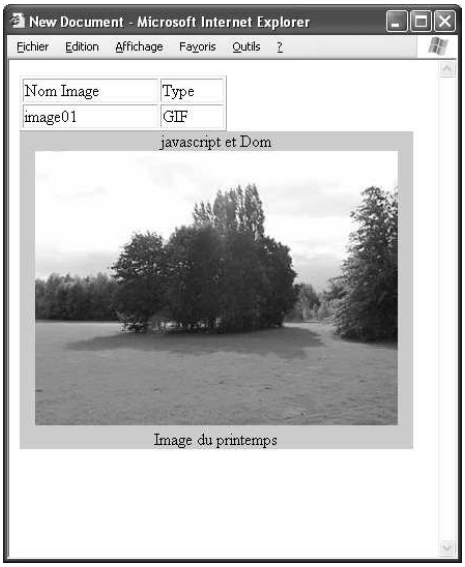
Modification de la structure

- Exemple

```
obj1=document.getElementById("Div0")
oTable=document.createElement("TABLE")
oTBody=document.createElement("TBody")
oTable.setAttribute('border','1');
oTable.setAttribute('width','200');
//création de la première ligne du tableau
oTR1=document.createElement("TR")
oTD11=document.createElement("TD")
oTEXT11=document.createTextNode("Nom Image")
oTD11.appendChild(oTEXT11)
oTR1.appendChild(oTD11)
oTD12=document.createElement("TD")
oTEXT12=document.createTextNode("Type")
oTD12.appendChild(oTEXT12)
oTR1.appendChild(oTD12)
oTBody.appendChild(oTR1)
 //création de la deuxième ligne du tableau
oTR2=oTR1.cloneNode(true)
oTR2.firstChild.firstChild.nodeValue="image01"
oTR2.firstChild.nextSibling.firstChild.nodeValue="GIF"
oTBody.appendChild(oTR2)
oTable.appendChild(oTBody)
oDocBody=document.getElementsByTagName("body")
oDocBody[0].insertBefore(oTable,obj1)
```

Modification de la structure

- Exemple



Modification de la structure

- Suppression de contenu :
  - **removeChild ()** : permet de supprimer un enfant d'un élément donné (ex. monDiv01.removeChild(elt_inclus))
  - **removeAttribute()** : permet de supprimer un attribut d'un élément donné (ex. monDiv01.removeAttribute("id"))
  - **removeAttributeNode()** : supprimer le nœud attribut d'un élément (ex. monDiv01.removeAttributeNode(monID))

Gestion des événements

Gestion des événements

- **Les événements sont des actions de l'utilisateur, qui vont pouvoir donner lieu à une interactivité**
- Chaque **composant implémente des méthodes** répondant aux **événements.**
- Les gestionnaires d'événements qui permettent d'associer une action à un événement.
- Différentes méthodes peuvent associer un événement sur un objet:
  - Evénement dans le tag HTML
  - Evénement directement lié à l'objet.
  - Ajouter un événement sur un élément

Gestion des événements

- **Evénements dans le tag HTML**
  - Définis comme attributs de la balise
- La syntaxe générale :

```
onEvenement="Action_Javascript_ou_Fonction();"
```

- Exemple :

```
<HTML>
<HEAD>
<TITLE> Gestion des événements </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<span style="background-color: yellow"
onmouseover="this.style.backgroundColor='blue'"
onmouseout="this.style.backgroundColor='yellow'">
Un élément avec un gestionnaire de base.</span>
</BODY>
</HTML>
```


Gestion des événements

- **Evénements dans le tag HTML**

Event	Objets	Description
abort	Image	Se produit lorsque l'utilisateur interrompt le chargement de l'image
blur	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window	L'élément perd le focus, l'élément n'est plus actif
change	FileUpload, Select, Submit, Text, TextArea	modification le contenu d'un champ de données.
click	Button, document, Checkbox, Link, Radio, Reset, Select, Submit	click sur l'élément associé à l'événement
dblclick	document, Link	double-click sur l'élément associé à l'événement
dragdrop	window	un glisser-déposer sur la fenêtre du navigateur
error	Image, window	Se déclenche lorsqu'une erreur apparaît durant le chargement de la page
focus	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text,TextArea, window	L'élément reçoit le focus

Gestion des événements

- **Evénements dans le tag HTML**

Event	Objets	Description
keydown	document, Image, Link, TextArea	appuyer sur une touche de clavier
keypress	document, Image, Link, TextArea	Maintenir une touche de clavier enfoncée
keyup	document, Image, Link, TextArea	Relâcher une touche du clavier
Load	Image, Layer, window	Chargement de la page
MouseOver	Area, Layer, Link	positionne le curseur de la souris au-dessus d'un élément
MouseOut	Layer, Link	le curseur de la souris quitte un élément
move	window	
Reset	form	Effacement des données d'un formulaire
Resize	window	Redimensionner de la fenêtre du
Select	text, Textarea	sélectionne un texte (ou une partie d'un texte) dans un champ de type "text" ou "textarea"
Submit	Form	clique sur le bouton de soumission d'un formulaire
Unload	window	quitter la page en cours

Gestion des événements

- **Evénements directement lié à l'objet.**
  - Sépare le code HTML de son comportement
  - La syntaxe générale :

```
document.getElementById('iDObjet').onclick=action_JS_ou_Fonction;
```

  - Exemple :

```
<HTML><HEAD><TITLE> Gestion des événements </TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<span style="background-color: yellow" id='mySpan' >
Un élément avec un gestionnaire séparé</span>
<script >
document.getElementById('mySpan').onmouseover=BGBlue ;
document.getElementById('mySpan').onmouseout=BGYellow ;

function BGBlue() { this.style.backgroundColor='blue' }
function BGYellow() { this.style.backgroundColor='yellow' }

</script>
</BODY></HTML>
```

Gestion des événements

- **Ajouter un événement sur un élément**
  - Ajout d'un événement à un élément créé dynamiquement en utilisant DOM
  - La syntaxe générale :

```
 // Mozilla, Firefox
if(window.addEventListener){
        object.addEventListener('mouseover', fctEvt, false);
        object.myProp = "valProp";
         }
// IExplorer
else { object.attachEvent('onmouseover', fctEvent);
        object.myProp = "valProp";
      }
```

```
 // Mozilla, Firefox (target)
 // IE (srcElement  function testevent(myEvt){
if (myEvt["target"]){
        e_out = evt["target"]["myProp"];
else    e_out = evt ["srcElement"]["myProp"];
alert(e_out);
}
```

Gestion des événements

- **Ajouter un événement sur un élément**
  - Exemple

```
<HTML><HEAD><TITLE> Gestion des événements </TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<span id='mySpan'>Un élément avec un gestionnaire séparé</span>
<SCRIPT LANGUAGE="JavaScript">
objSpan=document.getElementsByTagName("span");
oDIV=document.createElement("DIV")
oTxT=document.createTextNode("TxT")

oDIV.appendChild(oTxT)
objSpan[0].appendChild(oDIV)

oDIV.attachEvent('onmouseover', BGBlue)
oDIV.attachEvent('onmouseout', BGYellow)

function BGBlue(e) {
var monObj = e["srcElement"]
monObj.style.backgroundColor='blue' }
function BGYellow(e) {
var monObj = e["srcElement"]
monObj.style.backgroundColor="yellow" }
</SCRIPT></BODY></HTML>
```

Gestion des événements

- **Liste des Propriétés les plus utilisées dans MS**

Propriétés	Description	Exemple
Id	identification	
position	relative : l'objet est dépendant des modifs relatif à son parent. absolute : l'objet peut prendre des valeurs indépendamment de son parent	oIMG.style.position=absolute
posLeft	Abscisse du coin supérieur gauche par rapport à son parent	oIMG.style.posLeft=200;
posTop	ordonnée du coin supérieur gauche par rapport à son parent	oIMG.style.posTop=200;
left	Abscisse du coin supérieur gauche par rapport à la page	
top	ordonnée du coin supérieur gauche par rapport à la page	
style.visibility	Détermine la visibilité	Inherit, visible, hidden
style.zIndex	Ordre d'empilement des plans	
overflow	définit ce qui peut arriver à une zone lorsque son contenu est trop grand pour être correctement affiché	Les valeurs possibles : visible, hidden, scroll, auto