Javascript

Eléments de base du langage

1990 : début du Web

- HTML
- **1995 : Scripts Clients**
  - **Javascript, Jscript**
  - **Applets Java, ActiveX, Flash et**
- **2000: scripts Serveurs**
  - **PHP, ASP et ASP.NET, JSP.**
- **2005: cohabitation des scripts clients et serveurs**
- **Une application Web 2:**
  - **Xhtml**
  - **DOM**
  - **CSS**
  - **Ajax**

Javascript

- Principales utilisations:
  - Animations
  - Contrôle de saisie dans les formulaires.
  - Gestion des cookies .
  - Contrôle du navigateur : détection du navigateur utilisé, création de fenêtres …

Le langage Javascript

- Un script javascript peut être placé
  - Dans la partie head d'une page html
  - Dans la partie body
  - Dans un fichier f.js
  - <script src="f.js" />
- L'objet document représente le document en cours dans le navigateur.

Syntaxe

- Proche de celle du langage C
- Javascript est sensible à la casse
- Variables et types
  - Les variables ne possèdent pas de type statique, mais possèdent un type dynamique qui dépend de leur contenu (null: null, Boolean: true ou false, Number, String).

Les tableaux

- Exemple 1

```
var tab = new Array(3);
    tab[0] = 5;// l'indexation commence à partir de 0
    tab[1] = "Valeur";// chaque élément peut être de type différent
    tab[15] = 12.5;// le tableau sera automatiquement redimensionné
```

- On peut également ne pas déclarer de taille :

```
var tab = new Array;
var i;
for (i = 0 ; i<5 ; i++) tab[i] = 2*i;
for (i = 0 ; i<5 ; i++) document.write(tab[i]);
```

- Un tableau peut être indexé à l'aide d'une chaîne de caractères.

Fonctions

- Définition
  - function f1() {

```
    var i;
    for (i=0 ; i<arguments.length; i++)
document.write("Argument  " i+ ":"+ arguments[i] + "<br />");
return arguments.length

}
```

- Les fonctions doivent être définies dans la partie head
- Une fonction peut accepter un nombre variable de paramètres accessibles à partir du tableau arguments.
- Transmission de paramètres:
  - Par valeur: les types de base
  - Par référence: les tableaux et les objets (types Array et Object).
  - Une fonction peut retourner un résultat à l'aide d'une instruction return.

objets

- Création
  - Exemple1:
    - var produit = new Object;
    - produit.ref = "p01";
    - produit.des = "Chemise";
    - alert(produit.ref + ":" + produit.des);
  - Exemple 2: utiliser une fonction constructeur
    - function produit() {
    - this.ref=reference;
    - this.des=designation;
    - }
  - Un objet peut être utilisé avec la syntaxe des tableaux associatifs

objets

- Notion de prototype

```
function MonPrototype() {
this.a = 1;
this.b = function() { return 'prototype'; } this.e = 3; }
function MaClasse() {
this.c = 2;
this.d = function() { return 'classe'; }
this.e = 4; }
MaClasse.prototype = new MonPrototype();
monObjet = new MaClasse();
 monObjet.a; // 1
monObjet.b(); // 'prototype'
monObjet.c; // 2
monObjet.d(); // 'classe'
monObjet.e; // 4
```

Les boîtes de dialogue

```
alert("message d'alerte");
```

Confirmation

```
        if (confirm(" Action ?"))
                //action 1
        else
                alert("Action 2");

<script language="javascript">
        var reponse;
        reponse=prompt("Voulez-vous continuer?","oui");
        if (reponse==null)
                alert("Vous avez annulé la question");
        else
                if (reponse=="oui")
                        alert("On continue");
                else
                        alert("Choix incorrecte");
</script>
```

Les événements

- Une URL javascript dans un lien
  - cliquez ici.
- Une URL comme action d'un formulaire
  - <script>
  - function message() {
  - alert("Message");
  - Alert
  - }
  - </script>
  - </head>
  - <body>
  - <form name="saisie" action="javascript:message()">
  - <input type="submit" value="envoyer">
  - </form>

Les événements

- Exemple 1

```
<img     src="ordichapeau.gif"
     onClick="message('Vous avez cliqué');"
     onMouseOver="message('vous passez');"
     onMouseOut="message('vous repartez');"
/>
```

- L'événemet timeout

```
<script>
function changeImage() {
     document.zone.src=tabImages[numImage];
     if (numImage==2)
          numImage=0;
     else
          numImage++;
     setTimeout("changeImage()",1000);
}
</script>
<body>
<script>
var tabImages=new Array(3);
var numImage=1;
tabImages[0]="zone1.gif";
tabImages[1]="zone2.gif";
tabImages[2]="zone3.gif";
setTimeout("changeImage()",1000);
</script>
<img name="zone" src="zone1.gif" />
```

Les événements

Événement	Description
blur	Le curseur quitte l'élément : champs de formulaire, liens hypertexte et body (ou objet window).
focus	Le curseur entre dans l'élément : champs, liens hypertexte et body (ou objet window).
Change	Le contenu d'un champ de formulaire est modifié champ select ou texte (text ou textarea)
readystatechange	Changement d'état de la requête : objet XMLHttpRequest
resize	L'élément est retaillé.
scroll	L'utilisateur fait défiler la page.
reset	Annulation d'un formulaire, clic sur un bouton reset (balise form)
select	Sélection de texte champs textearea, select
submit	Envoi d'un formulaire (balise form)

Les événements

Événement	Description
click	L'utilisateur a cliqué sur l'élément.
keydown	L'utilisateur a enfoncé une touche du clavier.
keypress	L'utilisateur a pressé sur une touche (voir plus loin la différence avec keydownet keyup).
keyup	L'utilisateur a relâché une touche du clavier.
mousedown	L'utilisateur a enfoncé un bouton de la souris. lien, bouton, image
mousemove	Mouvement de la souris . lien, bouton, image
mouseout	La souris est sortie de l'élément (de la boîte qui le représente à l'écran). . lien, bouton, image
mouseover	La souris est entrée dans l'élément (dans la boîte qui le représente à l'écran). . lien, bouton, image
mouseup	Relâchement d'un bouton de la souris . lien, bouton, image

Evénements

- Association d'une réaction à un événement:
  - document.getElementById("unDiv").onclick = uneFonction;
  - Ou bien (DOM 2, non supportée par IE)
    - unElement.addEventListener('click', uneFonction, false);
- Suppression d'une réaction à un événement:
  - document.getElementById("unDiv").onclick =null;
  - Ou bien (DOM 2, non supportée par IE)
    - unElement.removeEventListener('click', uneFonction, false);//false -> phase de capture
- **Les gestionnaires d'événement Internet Explorer**

Remarque: IE

Les méthodes attachEvent et detachEvent fonctionnent de la même façon que addEventListener et removeEventListener, à quelques différences près :

- Il n'y a pas de troisième paramètre (seules les phases de cible et de bouillonnement sont supportées).
- Le type d'événement (premier paramètre) est préfixé par "on" (par exemple : "onload").
- Il n'y a aucun équivalent à this ou event.currentTarget, ce qui est peut être parfois très gênant (la cible n'étant pas systématiquement le même nœud que celui auquel on a attaché le gestionnaire d'événement).

Pour stopper la propagation de l'événnement:

- evenement.cancelBubble = true;
- et dans un navigateur W3C : evenement.stopPropagation();

L’objet événement

Chaque événement produit est un objet, qui porte plusieurs propriétés :

- Données relatives à la souris (position, bouton enfoncé).
- Données relatives au clavier (touche enfoncée, présence de l’un des modificateurs Alt, Ctrl, majuscule).
- Élément du DOM sur lequel survient l’événement: ev.target (w3c) et event.srcElement .
- Type de l’événement

L’événement possède aussi des méthodes, par exemple pour empêcher la réaction par défaut (pour l’événement submit: la soumission du formulaire, pour l’événement click sur un lien: le chargement de la page correspondante).

Récupération de l’événement dans la réaction:

- IE : window.event
- Pour le W3C l’événement doit être passé en paramètre de la réaction :

```
function f(evenement) {
// Recuperer l'evenement
var evenement = window.event || evenement;
// et le traitement a effectuer
}
```

Propriétés de l'événement

Propriété		Description
Internet Explorer	W3C	
altKey		Booléen indiquant si la touche Alt a été pressée.
cancelBubble=true	stopPropagation()	Empêcher l'événement d'être transmis aux éléments ancêtres de l'élément sur lequel est survenu l'événement.
clientX		Coordonnées de la souris par rapport au coin gauche haut de la fenêtre
clientY		
ctrlKey		Booléen indiquant si la touche Ctrl a été pressée.
fromElement	relatedTarget	Pour les événements de souris, élément d'où vient la souris.
keyCode		Code Unicode de la touche pressée pour les événements clavier
returnValue=false	preventDefault()	Empêche l'action par défaut associée à l'événement.
screenX		Coordonnées de l'événement par rapport à la fenêtre
screenY		
shiftKey		Booléen indiquant si la touche Maj a été pressée.
srcElement	target	Cible de l'événement (élément situé le plus en bas de l'arbre)
toElement	currentTarget	Pour les événements souris, élément destination de la souris

Evénements clavier

- Quand l'utilisateur presse une touche au clavier, trois événements sont produits, dans l'ordre suivant : keydown, keypress et keyup.
- Si l'utilisateur saisit plusieurs fois le même caractère en gardant la touche enfoncée, plusieurs événements keydown et keypress sont produits, contre un seul événement keyup.
- Cela peut notamment se produire quand l'utilisateur navigue, par exemple dans une liste, au moyen des touches fléchées.
- L'événement keypress n'est pas produit pour toutes les touches, en particulier pas pour les touches fléchées, ni pour les touches Alt, Ctrl et Maj.
- L'objet événement indique le numéro Unicode du caractère saisi à travers sa propriété keyCode. Nous obtenons le caractère par String.fromCharCode(keyCode). Nous pouvons utiliser cette propriété keyCode pour contrôler la saisie, ou la suggérer.
- La valeur du champ ne prend en compte le caractère saisi qu'avec l'événement keyup. Le dernier caractère saisi par l'utilisateur, « x », n'apparaît donc dans la valeur du champ que sur cet événement. Il serait plus court d'utiliser keyup pour une suggestion de saisie.
- Dans Mozilla, toujours pour keypress, keyCode vaut 0 s'il s'agit d'un caractère affichable, le code de celui-ci se trouvant alors dans charCode. Internet Explorer renseigne quant à lui toujours keyCode mais ne connaît malheureusement pas charCode.Le caractère saisi est renvoyé en majuscule pour les événements keydown et keyup. Par contre, avec keypress, nous avons la bonne valeur.

Evénements souris

- Position de la souris
  - ev.clientX et ev.clientY: donnent le nombre de pixels de la souris par rapport à la fenêtre.
  - Pour tenir compte du défilement de la page:
    - var left = event.clientX + document.body.scrollLeft;
    - var top = event.clientY + document.body.scrollTop;

Document HTML