

# JavaScript

Laurent Réveillère

Enseirb  
Département Télécommunications

`Laurent.Reveillere@enseirb.fr`  
`http://www.enseirb.fr/~reveille`

# Introduction

---

- ◆ JavaScript est un langage de programmation
  - interprété
  - non typé
  - intègre des éléments de POO
  - syntaxiquement proche de C, C++ et Java
- ◆ Introduit dans la version 2.0 de Netscape
- ◆ Code exécuté coté client (par le navigateur)
  - ➔ comportement dépendant du navigateur ...

# Principaux objectifs

- ◆ Répondre localement à certaines interactions utilisateur (animation graphique)
- ◆ Valider les données d'un formulaire avant de les transmettre
- ◆ Gérer dynamiquement des pages multi-cadres (modification de plusieurs cadres sur un clic)
- ◆ Agrémenter les pages Web de dates, heures, messages dans la barre de défilement, etc.
- ◆ Concevoir de petites applications...

# JavaScript et HTML

## ◆ Interne au document

- Définition du script dans le document html grâce aux balise `<script>...</script>`

## ◆ Externe au document

- Définition du script dans un fichier à part
- Liaison entre le fichier HTML et le script par la balise `<script>`

## ◆ Local à un élément

- Définition de code JavaScript dans un élément HTML et liaison grâce aux gestionnaires événement `<balise eventHandler="code JavaScript">`

# Exemple (script interne)

exemple1.html

```
<html>
  <body>
    <h1>Table des factorielles</h1>
    <script language="JavaScript">
      var i;           // Indice de boucle
      var fact = 1;   // Valeur de 0!
      for (i=1; i<=9; i++) {
        fact *= i;
        document.write (i + "! = " + fact + "<br>");
      }
    </script>
  </body>
</html>
```



# Exemple (script externe)

## exemple2.js

```
var i;           // Indice de boucle
var fact = 1;    // Valeur de 0!

for (i=1; i<=9; i++) {
    fact *= i;
    document.write (i + "! = " + fact + "<br>")
}
```

## exemple2.html

```
<html>
  <body>
    <h1>Table des factorielles</h1>
    <script language="JavaScript" src="exemple2.js"> </script>
  </body>
</html>
```



# Commentaires

- ◆ Mettre en commentaire une ligne (à la C++)

```
// Tous les caractères qui suivent sont ignorés
```

- ◆ Mettre en commentaire plusieurs lignes (à la C)

```
/* Toutes les lignes comprises entre ces repères  
sont ignorées par l'interpréteur de code */
```

- ◆ Ne pas confondre

- les balises de commentaires html (<!-- -->)
- les caractères de commentaires JavaScript (//, /\* \*/)

# Notion de variables

- ◆ Affectation d'une valeur à un identificateur

```
x = 2.5;  
var y;
```

- ◆ Préférable de déclarer explicitement une variable à l'aide du mot clé `var`
- ◆ Système de portée similaire au langage C
  - Les variables implicites sont globales
- ◆ Typage faible
  - une variable peut à tout moment changer de type
  - le type est déterminé au moment de l'affectation

# Principaux types de données

## ◆ number

- entiers signés, hexadécimaux (0x...), octal (0...), réels

## ◆ string

- suite de caractères encadrée par des guillemets simples ( ' ) ou doubles ( " )
- caractère spéciaux

`\b` touche de suppression

`\n` retour à la ligne

`\t` tabulation

`\'` guillemets simples

`\f` formulaire plein

`\r` appui sur la touche *ENTREE*

`\"` guillemets doubles

`\\` caractère contre-oblique

## ◆ boolean

- `true` **ou** `false`

# Opérateurs

## ◆ Comparaison

== != === !== < > <= >=

== != Test d'identité (conversion de type)

=== !== Test d'égalité (pas de conversion de type)

### – Exemple

» 2 == "2" true

» 2 === "2" false

## ◆ Arithmétiques + - \* / % ++ --

– La division entre deux entiers produit un réel

### – Exemple

» 1/2 → 0.5

# Opérateurs (suite)

- ◆ Manipulation de bits `& | ^ ~ << >> >>>`
  - `>>` **décalage à droite en conservant le signe**
  - `>>>` **décalage à droite en ajoutant des 0**
- ◆ Affectations `= += -= *= /= <<= >>= >>>= &= ~= |=`
- ◆ Logiques `&& || !`
- ◆ Sur les chaînes `+ +=`
- ◆ Sur les variables `typeof variable`
  - Retourne `number`, `boolean`, `string`, `object`, `function`, **ou** `undefined` **selon le type de la variable**

# Instructions

## ◆ Identiques à C

## ◆ Conditions

```
if (condition) { ... }  
if (condition) { ... } else { ... }  
condition ? ... : ...  
switch (expression) {  
    case constante : ... break;  
    default: ...  
}
```

## ◆ Boucles

```
for ( initialisation ; test ; increment ) { ... }  
while (condition) { ... }  
do { ... } while (condition)
```

# Instructions (suite)

## ◆ Déroulements

`break`

**Interruption**

`for, case, while, do-while`

`continue`

**passage à l'itération suivante**

`for, while, do-while`

# Fonctions

- ◆ Définition identique aux fonctions en C mais
  - pas de typage des arguments
  - pas de typage de la valeur de retour

```
function fact (n) {  
  if (n < 2)  
    return 1;  
  else return n * fact (n-1);  
}
```

# Fonctions (suite)

- ◆ Non déclaration exhaustive des arguments possible (accès par le tableau prédéfini `arguments`)

```
function produit () {  
    var i, prod = 1;  
    for (i = 0; i < arguments.length; i++) {  
        prod *= arguments[i];  
    }  
    return prod;  
}
```

# Fonction prédéfinies

## ◆ Évaluation d'expression

```
expr = "2 + 3 * 4";  
document.write (expr + " = " + eval(expr));
```

## ◆ Conversion d'une chaîne de caractère

```
parseInt("12", 10), parseInt("1100", 2),  
parseInt("0xC", 16)           retournent 12  
parseFloat("12.2")           retourne 12.2  
parseFloat("f12,2")          retourne NaN  
if (isNaN(...)) ...
```

## ◆ Codage des URLs et des cookies

```
escape("#intro")              retourne %23intro  
unescape("%23intro")          retourne #intro
```

# Modèle objet de JavaScript

- ◆ Modèle de prototype
  - Différent du modèle de classe (Java,C++, ...)
- ◆ Constructeur
  - Un objet instancié est « vierge » de propriétés
  - La fonction « constructeur » ajoute des propriétés

```
function printPersonne() {  
    document.write ("Personne: "+  
this.nom + ", " + this.age);  
}
```

```
function Personne (nom, age) {  
    this.nom = nom;  
    this.age = age;  
    this.print = printPersonne;  
    return this;  
}
```

```
function printVehicule() {  
    document.write ("Vehicule: "+  
this.marque + ", " + this.age);  
}
```

```
function Vehicule (marque, age) {  
    this.marque= marque;  
    this.age = age;  
    this.print = printVehicule;  
    return this;  
}
```

# Instanciación

- ◆ `new`            Instancie un objet vierge et appelle un constructeur

```
pers1 = new Personne ("Laurent", 28);  
pers2 = new Personne ("Clementine", 0);  
voit1 = new Vehicule ("Toyota", 1);
```

- ◆ `this`            Donne accès aux propriétés de l'objet courant
- ◆ `with`            Raccourci pour désigner les propriétés et les méthodes

```
with (document) {  
    write ("Bonjour");  
}
```

# Accès aux propriétés

- ◆ propriétés  $\equiv$  objet, valeur ou méthode
  - Référence à une propriété par l'opérateur `.` ou `[]`
- ◆ Exemples

```
function unAnDePlus() {  
    this.age++;  
}  
  
pers1.incrAge = unAnDePlus; // Ajout d'une nouvelle propriété  
voit1.incrAge = unAnDePlus; // Ajout d'une nouvelle propriété  
pers1.incrAge(); // appel de unAnDePlus() sur pers1  
voit1.incrAge(); // appel de unAnDePlus() sur voit1  
pers1.print(); // appel de printPersonne() sur pers1  
voit1.print(); // appel de printVehicule() sur voit1
```

# Accès aux propriétés (suite)

## ◆ Structure de contrôle `for in`

```
function dumpObject (obj) {  
  var result = "";  
  for (field in obj) {  
    result += field + " = " + obj[field] + "<br/>";  
  }  
  result += "<hr/>";  
  return result;  
}  
  
with (document) {  
  write (dumpObject(pers1));  
  pers1.incrAge();  
  write (dumpObject(pers1));  
  write (dumpObject(voit1));  
}
```

# Objet String

- ◆ Représentation des chaînes de caractères
- ◆ Longueur maximale à priori comprise entre 50 et 80 caractères
- ◆ Premier caractère en position 0 et dernier en position n-1, où n est la taille de la chaîne
- ◆ Possède une seule propriété
  - `length` **retourne la longueur de la chaîne**

```
var taille = ("petit exemple").length;  
var chaine = new String ("Un autre exemple");  
var taille = chaine.length;
```

# Principales propriétés

- ◆ `charAt(pos)`  
Retourne le caractère situé à la position `pos`
- ◆ `indexOf(sub, pos)`  
Retourne la position de la sous-chaîne `s` en effectuant la recherche de gauche à droite à partir de la position `pos`
- ◆ `lastIndexOf(sub, pos)`  
Similaire à `indexOf()`, à la différence que la recherche se fait de droite à gauche
- ◆ `substring(pos1, pos2)`  
Retourne la sous-chaîne comprise entre les positions `pos1` et `pos2`
- ◆ `toLowerCase()`  
Convertit tous les caractères en minuscules
- ◆ `toUpperCase()`  
Convertit tous les caractères en majuscules

# Objet Array

- ◆ Premier élément à l'indice 0
- ◆ Accès aux éléments avec la notation `tab[pos]`
- ◆ Peut contenir des éléments de types hétérogènes
- ◆ La taille peut être augmentée à la demande
- ◆ Déclaration (formes équivalentes)

```
t = new Array ("laurent", "", 28);  
t = [ "laurent", "", 28 ];  
t = new Array (); t[0] = "laurent"; t[1] = ""; t[2] = 28;  
t = new Array (3); t[0] = "laurent"; t[1] = ""; t[2] = 28;
```

# Principales propriétés

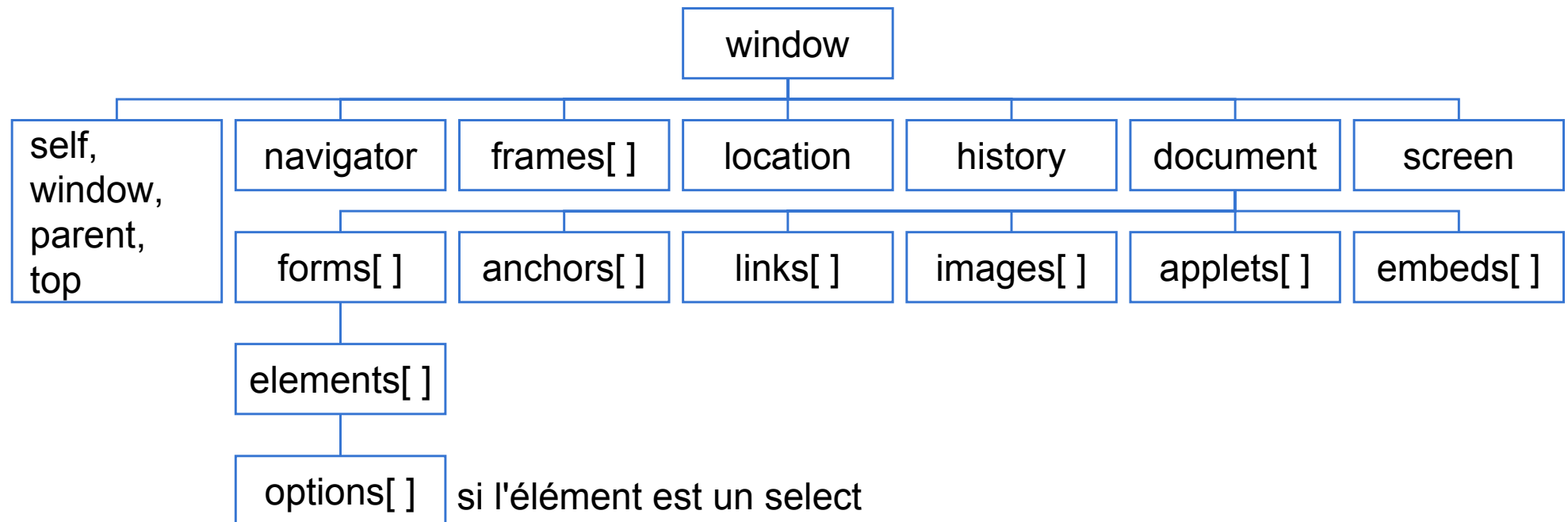
- ◆ `t.length` Retourne la taille du tableau
- ◆ `t.concat(t1, ...)` Retourne un tableau qui est la concaténation de `t`, `t1`, ...
- ◆ `t.join(sep)` Retourne la concaténation des éléments de `t` séparés par `sep`
- ◆ `t.reverse()` Renverse les éléments de `t`
- ◆ `t.slice(from, to)` Retourne un sous-tableau des positions `from` (inclus) à `to` (exclus)
- ◆ `t.sort()` Tri lexicographique

# JavaScript et HTML

---

- ◆ Description en objets JavaScript
  - du navigateur
  - des fenêtres
  - des documents qui les composent
- ◆ Consultation et modification des propriétés de ces objets

# Hiérarchie d'objets



# Exemple

Formulaire - Microsoft Internet Explo

Fichier Edition Affichage Favoris Outil: >>

← Précédente →

Adresse C:\Documents and Sett OK Liens >>

## Sondage sur le cour

Nom

Prénom

Je viens en cours:

toujours

souvent

qu'en j'y pense

jamais

Votre opinion

Entrez votre commentaire ici

Merci pour votre réponse

Terminé Poste de travail

Objet fenêtre

Objet document

Objet formulaire

# Exemple (suite)

Form titled "Sondage sur le cour" with the following elements:

- 1st text input field: "Nom"
- 2nd text input field: "Prénom"
- Radio button group: "Je viens en cours:" with options "toujours", "souvent", "qu'en j'y pense", "jamais"
- Dropdown menu: "Votre opinion" (selected: "Satisfait")
- Text area: "Entrez votre commentaire ici"
- Buttons: "ENVOYER" and "ANNULER"

1<sup>er</sup> objet texte

`elements[0]`

2<sup>ème</sup> objet texte

`elements[1]`

1<sup>er</sup> objet radio

`elements[2]`

2<sup>ème</sup> objet radio

`elements[3]`

3<sup>ème</sup> objet radio

`elements[4]`

4<sup>ème</sup> objet radio

`elements[5]`

1<sup>er</sup> objet liste déroulante

`elements[6]`

1<sup>er</sup> objet champ de texte

`elements[7]`

1<sup>er</sup> objet bouton

`elements[8]`

2<sup>ème</sup> objet bouton

`elements[9]`

# Document simple

```
<html><body>
<center></center>

<form name="formulaire">
<table border="0">
  <tr> <td align="right">Nom</td>
  <td><input name="nom" type="text"
          value="" size="8" /></td>
</tr>

  <tr> <td align="right">Mot de passe</td>
  <td><input name="motpasse" type="password"
          value="" size="15" /></td>
</tr>
</table>
<center><input type="button" value="Valider"
          name="bouton"></center>

</form>

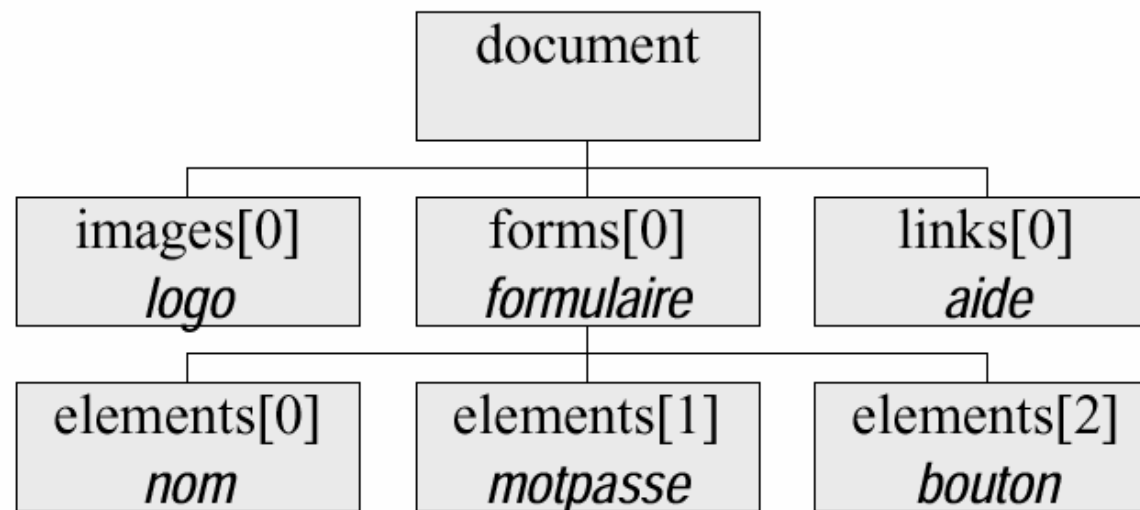
<div align="right"><i>
<a href="aide1.html" name="aide">Aide</a>
</i></div>
</body></html>
```



# Réprésentation arborescente

## ◆ Exemple de script relatif au document

```
passwd = document.forms[0].elements[1].value;
if( passwd == "toto") {
    document.images[0].src = "salut.gif";
} else {
    document.logo.src = "youps.gif";
}
```



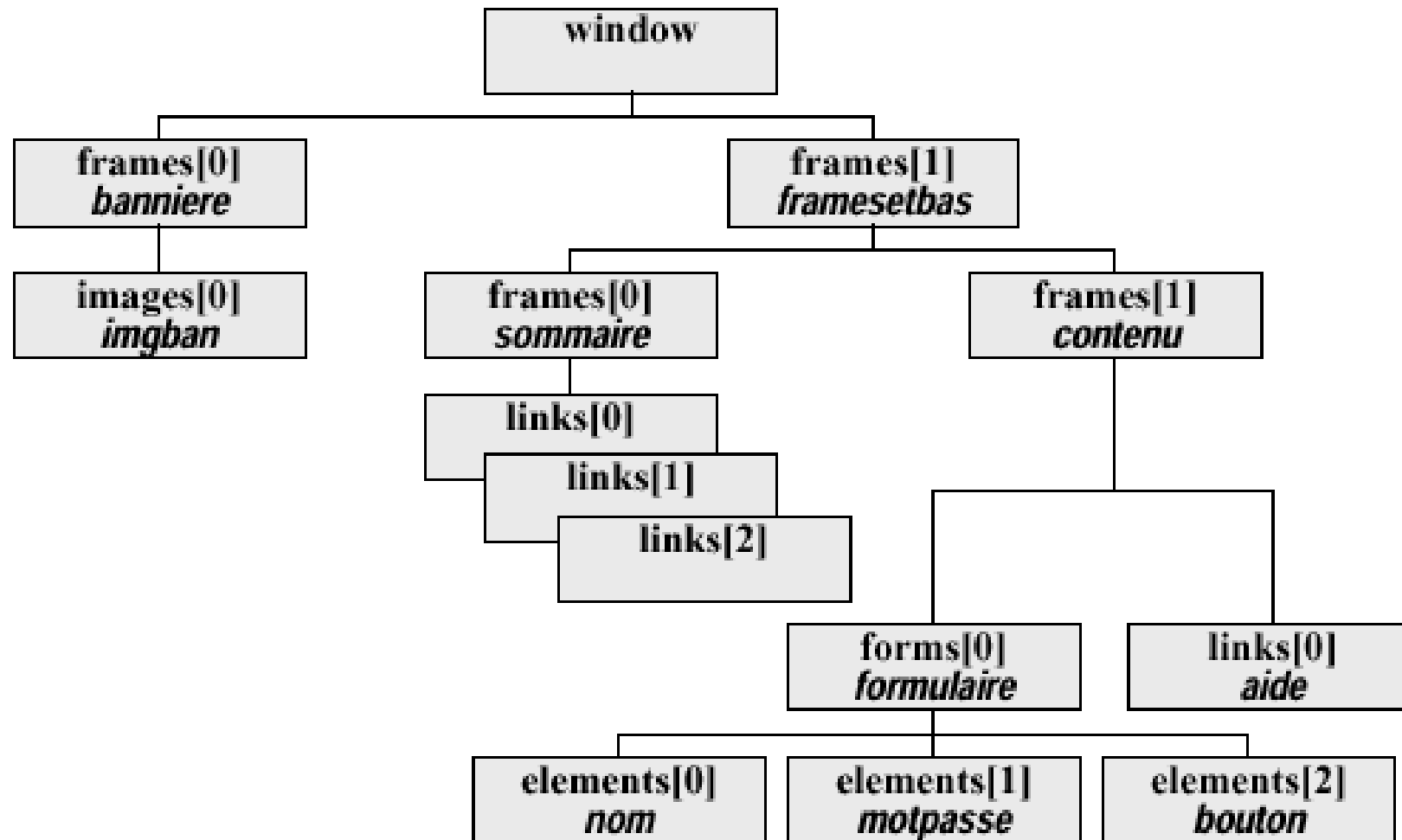
# Document avec cadres

```
<html>
  <frameset rows="100,*">
    <frame name="banniere"
          src="f_banniere.html">

    <frameset cols="100,*"
              name="framesetbas">
      <frame name="sommaire"
            src="f_sommaire.html">
      <frame name="contenu"
            src="f_ident.html">
    </frameset>
  </frameset>
</html>
```



# Réprésentation arborescente



# Notion d'événement

- ◆ Définition: action de l'utilisateur pouvant donner lieu à une interactivité
- ◆ JavaScript permet de gérer des événements tels que le passage de la souris au-dessus d'une zone, le changement d'une valeur, ...
  - Html ne gère que l'événement *clic de souris*
- ◆ Un gestionnaire d'événement associe une action à un événement

```
onEvent = "Action JavaScript à réaliser"
```

# Liste des événements

Évènements	Signification	Objets
<code>onblur</code>	Élément n'est plus la cible de saisie	<code>button</code> , <code>input</code> , <code>label</code> , <code>select</code> , <code>textarea</code>
<code>onchange</code>	Valeur modifiée depuis que l'élément est cible de saisie	<code>input</code> , <code>select</code> , <code>texttearea</code>
<code>onclick</code>	Click de la souris <sup>1</sup>	Tous les éléments <sup>2</sup>
<code>ondblclick</code>	Double click de la souris <sup>1</sup>	Tous les éléments <sup>2</sup>
<code>onfocus</code>	Élément devient la cible de saisie	<code>button</code> , <code>input</code> , <code>label</code> , <code>select</code> , <code>textarea</code>
<code>onkeydown</code>	Touche enfoncée <sup>1</sup>	Éléments de formulaire et <code>body</code>
<code>onkeypress</code>	Touche enfoncée et relâchée <sup>1</sup>	Éléments de formulaire et <code>body</code>
<code>onkeyup</code>	Touche relâchée <sup>1</sup>	Éléments de formulaire et <code>body</code>
<code>onload</code>	Chargement document ou cadre terminé	<code>body</code> , <code>frameset</code>

# Liste des événements (suite)

Évènements	Signification	Objets
<code>onmousedown</code>	Bouton de la souris enfoncé	Tous les éléments <sup>2</sup>
<code>onmousemove</code>	Déplacement du curseur	Tous les éléments <sup>2</sup>
<code>onmouseout</code>	Sortie du curseur de l'élément	Tous les éléments <sup>2</sup>
<code>onmouseover</code>	Curseur au dessus de l'élément	Tous les éléments <sup>2</sup>
<code>onmouseup</code>	Bouton de la souris relâché	Tous les éléments <sup>2</sup>
<code>onreset</code>	Réinitialisation de formulaire	<code>form</code>
<code>onselect</code>	Sélection d'un texte	<code>input</code> , <code>textarea</code>
<code>onsubmit</code>	Soumission d'un formulaire	<code>form</code>
<code>onunload</code>	Déchargement document ou cadre	<code>body</code> , <code>frameset</code>

<sup>1</sup> : false annule l'action

<sup>2</sup> : La plupart des éléments seulement

# Méthode `write`

- ◆ Modification dynamique du contenu d'une page html
- ◆ Insertion d'une chaîne de caractère (pouvant contenir des balises html) à l'endroit où est placé le script
- ◆ Différentes utilisation possibles

```
document.write ("Salut");  
  
var msg="Salut"; document.write (msg);  
  
var msg="Salut"; document.write (msg+ "!");  
  
var msg="Salut"; document.write ("" + msg+ "");
```

# Création de fenêtre

- ◆ `open()` et `close()` permettent d'ouvrir et de fermer des fenêtres

```
window.open(URL, "nom de la fenetre","liste d'options");  
window.close ("nom de la fenetre");
```

- ◆ Exemple

```
function newWindow () {  
    var win = window.open ("", "exemple",  
        "width=500,height=400,status=yes,toolbar=no,menubar=yes");  
    win.document.open();  
    win.document.writeln("<html><head><title>Exemple de open</title></head>");  
    win.document.writeln("<body>");  
    win.document.writeln("Ceci est un document généré par un script<br />");  
    win.document.writeln("</body>");  
    win.document.close();  
    return win;  
}
```

# Manipulation de fenêtre

```
<html>
<head>
  <script language="JavaScript" src="open.js"></script>
</head>

<body>
<form>
  <input type="button" value="Ouvrir" onClick="win = newWindow()">
  <input type="button" value="Fermer" onClick="win.close()">
</form>
</body>
</html>
```

# Boîte de dialogue

- ◆ Fenêtre qui s'affiche au premier plan suite à un événement, et qui permet
  - d'avertir l'utilisateur
  - de le confronter à un choix
  - de lui demander de compléter un champ pour récupérer une information
- ◆ Trois différents types
  - alert
  - confirm
  - prompt

# Boîte de dialogue : alert

- ◆ Permet d'afficher dans une boîte composée d'une fenêtre et d'un bouton "OK" le texte qu'on lui fournit en paramètre
- ◆ Exemple

```
<html><head><script language="JavaScript">
function verifAge (form) {
  if ((form.age.value < 0) || (form.age.value > 150)) {
    alert ("Votre age semble incorrect!");
    form.age.value = "";
  }
}
</script></head>

<body><form name="exa">
Nom <input type="text" name="nom" value=""> <br />
Prenom <input type="text" name="age" onChange="verifAge(exa)" value=""><br />
<input type="submit" value="Envoyer"> <input type="reset" value="Annuler">
</form></body></html>
```

# Boîte de dialogue : confirm

- ◆ Similaire à la méthode `alert()` , si ce n'est qu'elle permet un choix entre "OK" et "Annuler"
- ◆ Comportement des boutons
  - `ok` **retourne la valeur true**
  - `cancel` **retourne la valeur false**
- ◆ Exemple

# Boîte de dialogue : prompt

- ◆ Fournit un moyen de récupérer une information provenant de l'utilisateur
- ◆ Requier deux arguments
  - le texte d'invite
  - le texte par défaut dans le champ de saisie
- ◆ Exemple

# Timer (code JavaScript)

```
var txt = "Ceci est un message qui bouge!";
var length = txt.length;
var width = 100;
var pos = -(width + 2);
function scroll() {
  var scroller = "";
  pos++;
  if (pos == length) {
    pos = -(width + 2);
  }
  if (pos < 0) {
    for (var i = 1; i <= Math.abs(pos); i++) {
      scroller += " ";
    }
    scroller += txt.substring (0, width - i + 1);
  } else {
    scroller += txt.substring (pos, width + pos);
  }
  window.status = scroller;
  x = setTimeout ("scroll()", 100);
}
```

[...]

```
function clearScroll () {
  window.status = "";
}
```

# Timer (code html)

```
<html>
<head>
  <script language="JavaScript" src="timer.js">
  </script>
</head>

<body onLoad="scroll()">
<a href="javascript:clearTimeout(x); clearScroll()">
Suppression du scroll</a>
<h1>Exemple du scroll</h1>
</body>
</html>
```

# Cookies

- ◆ Permettent au navigateur de stocker des informations de manière à pouvoir les récupérer lors d'une prochaine visite
- ◆ Les cookies sont stockés sur l'ordinateur client et non sur le serveur
- ◆ Exemple d'information à mettre dans un cookie
  - nom d'utilisateur
  - langue de l'utilisateur
  - mot de passe

# Fonctionnement d'un cookie

- ◆ Contenu automatiquement chargé dans l'objet `document.cookie`
- ◆ Par défaut, durée de vie d'un cookie égale à la session
  - Possibilité de définir une date d'expiration
    - » `expires=date`
  - Date convertie dans l'heure de Greenwich
    - » `toGMTString()`
- ◆ Les paramètres d'un cookie sont séparés par un point-virgule (;)

# Fonctions de manipulations

## ◆ Création

```
function setCookie (nom, valeur, secondes) {  
    var expireDate = new Date();  
    expireDate.setTime(expireDate.getTime() + (secondes * 1000));  
    // création du cookie  
    document.cookie = nom + "=" + escape(valeur) + ";expires=" +  
        expireDate.toGMTString();  
}
```

# Fonctions de manipulations

## ◆ Lecture

```
function getCookie (nom) {
    // on vérifie si il y a un cookie
    if (document.cookie.length > 0) {
        debut = document.cookie.indexOf(nom + "=");
        if (debut != -1) {
            // on vérifie si la valeur qu'on recherche est dans le cookie
            debut += nom.length + 1;
            fin = document.cookie.indexOf(";", debut);
            if (fin == -1) {
                fin = document.cookie.length;
            }
            return unescape(document.cookie.substring(debut, fin));
        }
    }
    return null; /* la valeur n'a pas été trouvée */
}
```

# Fonctions de manipulations

## ◆ Suppression

```
function delCookie(nom) {  
    if (getCookie(nom)) {  
        /* en mettant cette date, le cookie sera désactivé */  
        document.cookie = nom + ";expires=Thu, 01-Jan-70 00:00:01  
GMT";  
    }  
}
```

# Exemple

```
<html><body>
<script language="JavaScript">
var yourname = getCookie ("exemple_cookie");
if (yourname != null) {
    document.write ("Bonjour " + yourname);
} else {
    document.writeln ("
```