Partie 2 – Le langage JavaScript

Olivier GLÜCK
Université LYON 1/UFR d'Informatique
Olivier.Gluck@ens-lyon.fr
http://www710.univ-lyon1.fr/~ogluck

Copyright

Remerciements

- Quelques transparents sont directement tirés des supports de cours de :
  - Dominique Bouillet (INT)
- Merci à eux !
- Des figures et exemples sont issus des livres ou sites Internet cités en bibliographie

Bibliographie

- « *Webmaster in a nutshell* », S. Spainhour & R. Eckstein, 3ième édition, O'REILLY, ISBN 0-596-00357-9
- « ***JavaScript, La référence*** », D. Flanagan, 4ième édition en français, O'REILLY, ISBN 2-84177-212-8
- « *Création d'un site Web du débutant à l'expert* », Daniel Ichbiah, Eska, ISBN 2-7472-0227-5
- « *HTML et JavaScript* », P. Chaléat et Daniel Charnay, Eyrolles, ISBN 2-212-11157-6
- Internet et JavaScript…
  - http://www.aidejavascript.com/sommaire.php3
  - http://conceptnet.online.fr/accueil.htm
  - http://www.ac-creteil.fr/util/programmation/javascript/Welcome.html
  - http://www.le-webmestre.net/web/cours/javascript/
  - http://www.henri-ruch.ch/javascript/cours.asp
  - http://developpementweb.online.fr/index1.html
  - http://www.w3.org/

Plan de la partie 2 (2 séances)

- Le langage JavaScript
- Les objets prédéfinis
- Les événements
- Les objets du noyau
- Quelques exemples classiques

Le langage JavaScript

Motivations

- Insertion d'instructions de programmation directement dans le code des pages HTML
- Exécution de code sur le poste client pour
  - améliorer l'interactivité (temps de réponse plus court)
  - améliorer les débits sur le réseau (éviter des envois erronés)
  - proposer des pages dynamiques (animation, personnalisation,…)
- Exemples
  - test d'un formulaire avant envoi
  - animation type texte défilant
  - affichage dynamique

Le langage JavaScript

- Créé à l'origine par Netscape (Netscape 2.0)
- Conçu pour traiter localement des événements provoqués par le client
  - déplacement du pointeur de souris ou click de souris
  - soumission d'un formulaire, …
- JavaScript est un langage
  - interprété (le code du script est analysé et exécuté au fur et à mesure par l'interprète, partie intégrante du navigateur) - problème en cas d'erreurs !
  - à base d'objets
  - multi-plateforme

Le langage JavaScript

- JavaScript permet
  - de programmer des actions en fonction d'événements
    - si la zone de saisie contient un nombre alors enregistrer la valeur sinon afficher une erreur
  - d'effectuer des calculs (sans recours au serveur)
    - lire la valeur saisie, la multiplier par 3,14 et afficher le résultat

Le langage JavaScript

- Domaines d'applications
  - petites applications simples (calculette, outils de conversions, édition automatique de devis, jeu, …)
  - aspects graphiques de l'interface (modification d'images lors du passage de la souris, gestion de fenêtres, modification locale de la page HTML, modification de menus…)
  - test de validité des données sur les éléments de l'interface de saisie
    - vérifier qu'une valeur considérée comme obligatoire a bien été saisie
    - vérifier que le champ saisi correspond bien au format demandé

Autres langages analogues

- Le langage VBScript
  - proposé par Microsoft
  - dérivé de Visual Basic
- Le langage Jscript (microsoft)

Le noyau JavaScript

- Au niveau du langage, on distingue
  - le noyau JavaScript (cœur du langage) comportant des objets prédéfinis, des opérateurs, des structures, …
  - un ensemble d'objets associés au navigateur
    - fenêtres,
    - documents,
    - images, …
- Il existe aussi la possibilité d'exécuter du code JavaScript sur le serveur (communications avec une BD…) mais l'usage est plus restreint

Normalisation

- ECMA (European Computer Manufacturers Association) a défini un standard ECMAScript basé sur JavaScript 1.1
- Ce standard, repris par l'ISO, définit les caractéristiques du noyau du langage
- JavaScript 1.3 et Jscript 3.0 sont conformes à cette norme mais ils ont aussi tous les deux
  - leurs propres extensions
  - des différences au niveau du modèle objet du navigateur

JavaScript n'est pas JAVA !

- JavaScript
  - interprété
  - à base d'objets prédéfinis (pas d'héritage)
  - code intégré dans HTML (visible)
  - typage faible
  - n'existe pas en dehors du Web
  - *débuggage* difficile
- Java
  - compilé
  - orientés objets (définition de classes, héritage)
  - code dans applets (non visible)
  - typage fort
  - langage à part entière
  - environnement de développement

Communication possible entre Java et JavaScript grâce au plug'in LiveConnect (Netscape) ou aux contrôles Active X (Microsoft)

Insertion de code JavaScript

- 3 méthodes
  - utilisation de la balise <script>…</script>
    - déclaration de fonction dans l'en-tête entre <head> et </head>
    - appel de fonction ou exécution d'une commande JavaScript dans <body>…</body>
    - insertion d'un fichier JavaScript "externe"
  - utilisation dans une URL
    - une URL peut être une exécution de fonction JavaScript (entre <a>… ou <form>…</form>
  - utilisation de nouveaux attributs de balise pour la gestion d'événements utilisateur
    - <BALISE onEvenement="code JavaScript">

Insertion de code JavaScript

```
<!-- index.html -->
<HTML><HEAD>
        <SCRIPT LANGUAGE="JAVASCRIPT">
                function fin() { window.close(); }
        </SCRIPT>
</HEAD><BODY>
        <SCRIPT LANGUAGE="JAVASCRIPT">
                document.write('Pour fermer la fenêtre');
        </SCRIPT>
        <br><a href="javascript:fin();">cliquez ici</a>
        <br> ou passez la souris sur
        <a href="" onMouseOver="fin();">ce lien</a>
</BODY></HTML>
```


La balise <SCRIPT>

- Introduite pour permettre l'exécution de code par le navigateur
- Syntaxe générale

<SCRIPT LANGAGE="nom" SRC="URL" ARCHIVE="fichier.jar" ID="entier">…</SCRIPT>

- Attribut LANGAGE
  - "JavaScript" par défaut dans Netscape
  - permet de préciser la version
- Attribut SRC
  - permet de charger du code présent dans un autre fichier
- Attributs ARCHIVE et ID
  - utilisés pour la sécurisation (signature digitale)

Insertion d'un fichier externe

```
<!-- index.html -->
<HTML><HEAD>
        <SCRIPT LANGUAGE="JAVASCRIPT" SRC="fin.js">
        </SCRIPT>
</HEAD><BODY>
        <SCRIPT LANGUAGE="JAVASCRIPT">
                document.write('Pour fermer la fenêtre');
        </SCRIPT>
        <br><a href="javascript:fin();">cliquez ici</a>
        <br> ou passez la souris sur
        <a href="" onMouseOver="fin();">ce lien</a>
</BODY></HTML>
```

```
// fin.js
function fin() {
        window.close();
}
```

La balise <NOSCRIPT>

- Bonne utilisation de la balise <SCRIPT>

```
<SCRIPT LANGAGE="JavaScript1.2">
<! --
Code JavaScript
-->
</SCRIPT>
<NOSCRIPT>
Attention, ce document contient du code JavaScript non
   interprété par votre navigateur…
</NOSCRIPT>
```


Le langage JavaScript

- Des variables faiblement typées
- Des opérateurs et instructions (ceux du langage C)
- Des méthodes
  - globales (associées à tous les objets)
  - fonctions définies par l'utilisateur
- Des objets
  - prédéfinis (String, Date, Math, …)
  - liés à l'environnement (window, document, …)
- Commentaires comme en langage C
  // une seule ligne ou /* plusieurs lignes */
- Séparateur d'instructions : **;** (ou retour-chariot)

Opérateurs JavaScript

- Ceux du langage C
  - arithmétiques : + - * / %
  - in/décrémentation (pré/post indexée) : k++ ++k
  - logiques : && (ET) | | (OU) ! (NON)
  - bit à bit : & (AND) | (OR) ^ (XOR) ~ (Not)
  - décalages : >> (à droite) << (à gauche) >>> (non signé)
  - comparaisons: == != <= >= < >
  - concaténation de chaînes : +

ATTENTION : L'opérateur + est l'addition ou la concaténation

Affectations

- affectation simple
  nom=valeur;
- affectation conditionnelle
  var = (condition) ? exp_alors : exp_sinon;
  X = (a > b) ? "plus" : "moins ";
- affectation avec opération : += -= *= ...
  X +=3; // équivaut à X=X+3;
- ATTENTION : distinguer l'affectation (=) et la comparaison (==)

Variables JavaScript

- JavaScript est sensible à la casse
- Déclaration de variables
  - optionnelle mais fortement conseillée
  - avec l'instruction *var*
  - le type n'est pas précisé lors de la déclaration
  - initialisation possible lors de la déclaration sinon valeur *undefined*
- Notion de variables locales et globales
  - locales à une fonction
  - globales au document HTML
- Utilisation d'une variable globale d'un autre document (autre frame)
  window.parent.droite.nomvar

Variables JavaScript

```
<!-- index.html -->
<HTML><HEAD>
        <SCRIPT LANGUAGE="JAVASCRIPT1.2">
                var Age=Math.round(70*Math.random());
                var Nom;
                var Prenom='Florent';
                function affiche() {
                   var Nom='Dupont';
                   var Age=Math.round(10*Math.random());
                   document.write(Prenom + ' ' + Nom + ' ' + Age + ' ans<br>');
                }
        </SCRIPT>
</HEAD><BODY>
        <SCRIPT LANGUAGE="JAVASCRIPT1.2">
                document.write(Prenom + ' ' + Nom + ' ' + Age + ' ans<br>');
                affiche();
        </SCRIPT>
</BODY></HTML>
```

Variables et types

- Le typage a lieu lors de l'initialisation ou d'une affectation
- Le type d'une variable peut changer si on lui affecte une valeur d'un autre type
- Les types de données simples
  - Nombre (***Number***)

    Entier : décimal ou hexa (0x4F) ou octal (075)

    Réel (-2.3452E-12)
  - Booléen (***Boolean***) : true ou false
  - Chaîne de caractères (***String***)

    'chaine' ou "chaine"

    Les codes \t (tabulation) \n (à la ligne) \r (retour chariot) \b (backspace) \f (saut de page) sont reconnus

Conversions de type

- Type String = type dominant
- JavaScript fait des conversions implicites selon les besoins
- Exemples
  - N=12; // N numérique
  - T="34"; // T chaîne de caractères
  - X=N+T; // X est la chaîne de caractères "1234"
  - Il existe des types particuliers : null, undefined, objet, function
  - et des nombres particuliers : Infinity, -Infinity, NaN (Not a Number)

Instructions classiques

- Instructions de branchement

  if (condition) { instructions; } [else { instructions; }]
- Boucles

  for (i=1 ; i<N ; i++) { instructions; }

  while (condition) { instructions; }

  do { instructions; } while (condition)

  for (p in objet) { instructions; }
  - Sortie d'une boucle

  break;
  - Itération suivante d'une boucle

  continue;
- typeof(entite) retourne le type de l'entité

Instructions classiques

- Le bloc switch

```
switch (variable) {
   case 'valeur 1':
   Code à exécuter si "variable == 'valeur 1'";
   break;
   case 'valeur 2':
   Code à exécuter si "variable == 'valeur 2'";
   break;
   case 12:
   Code à exécuter si "variable == 12";
   break;
   default:
   Code à exécuter si tous les autres ont échoué;
   break;
   }
```


Les fonctions

```
function nom_f (arg1, …, argN) {
   instruction1;
   …
   instructionN;
   return valeur;
}
```

- arguments non typés
- nombre d'arguments non fixé par la déclaration

Les fonctions - exemple

```
<!-- index.html -->
<HTML><HEAD>
  <SCRIPT LANGUAGE="JAVASCRIPT1.2" SRC="function.js">
  </SCRIPT>
</HEAD><BODY>
  <SCRIPT LANGUAGE="JAVASCRIPT1.2">
    document.write('Aujourd\'hui, on est ' + jour() + '<br>');
  </SCRIPT>
  <FORM name="naissance"
        onSubmit="document.write('Vous êtes nés un ' +
        jour(this.date_naissance.value));">
    <center>
      Entrez votre date de naissance sous la forme jj/mm/aaaa : <br>
      <input type="text" name="date_naissance">
      <input type="submit" value="Je veux voir !">
    </center>
  </FORM>
</BODY></HTML>
```

Les fonctions - exemple

```
// function.js
/* retourne le jour de la semaine si pas d'argument sinon le jour de la semaine
correspondant à la date passée en argument jj/mm/aaaa */
function jour(arg_date) {
   var date_AMJ;
   var Semaine = new Array('Dimanche', 'Lundi', 'Mardi', 'Mercredi', 'Jeudi',
                           'Vendredi', 'Samedi');
   if (arg_date) {
      // on récupère arg_date dans un tableau d'entiers
      date_JMA = arg_date.split('/');
      // on crée un objet Date en fournissant année, mois (0-11), jour
      date_AMJ = new Date(date_JMA[2], date_JMA[1]-1, date_JMA[0]);
   } else {
      // on crée un objet Date avec la date du jour
      date_AMJ = new Date();
   }
   return Semaine[date_AMJ.getDay()];
}
```


Les objets

- Pas de classe mais des pseudo-classes
  - pas de sous-classe
  - pas d'héritage
  - uniquement des créations d'objets et la possibilité de définir des propriétés "prototype"
  - syntaxe largement inspirée de la programmation objet
- Objets prédéfinis
  - accès à une propriété (objet.propriété)
  - accès à une méthode (objet.méthode)
- Création d'un objet par la définition de son constructeur

Les objets - exemple

```
// constructeur de l'objet individu
function individu(N, P, D) {
   // propriétés d'un individu
   this.nom = N;
   this.prenom = P;
   this.date = D;
   // méthodes d'un individu
   this.age = calcul_age;
}
```

```
// définition de la méthode calcul_age
function calcul_age() {
   // date du jour
   var date_today = new Date();
   // date de naissance de l'individu
   date_indiv = this.date.split('/');
   ...
   return age_indiv;
}
```

Complétez !

```
// créer une instance de l'objet individu - appel du constructeur avec trois arguments
Pauline = new individu('Pauline', 'Dupont', '3/12/75');
// modifier une propriété
Pauline.date = '3/12/78';
// appel de la méthode age()
document.write(Pauline.age());
// suppression de l'instance Pauline
delete Pauline;
```


Les objets - la propriété prototype

- Toutes les classes JavaScript ont une propriété particulière "**prototype**" permettant d'ajouter une nouvelle propriété ou méthode à une classe

Nom_classe.prototype.New_propriete = Valeur_par_defaut;

  - Nom_classe est une classe prédéfinie ou définie par l'utilisateur
  - New_propriete est le nom de la nouvelle propriété ; elle est créée pour tous les objets déjà instanciés ; si elle existe déjà, cela permet de lui affecter une valeur par défaut
  - Valeur_par_defaut peut être une référence vers une fonction si on désire ajouter une nouvelle méthode

Les tableaux

- Construire un tableau sans préciser le contenu
  var Tab = new Array();
- Construire un tableau en précisant la taille
  var Tab = new Array(3);
- Initialisation du tableau lors de sa création
  var Tab = new Array(t1, ..., tN);
  -> les indices varient de 0 à N-1
  -> les ti peuvent être de types différents
- propriété *length* : taille du tableau
  Tab.length = N;
- La taille du tableau est dynamique

Tableaux et objets

- Un tableau est un objet prédéfini qui possède des propriétés et des méthodes
- Tableaux associatifs : l'indice peut être une chaîne de caractères
  Tab['nom'] est équivalent à **Tab.nom**
- Un objet peut être considéré comme un tableau associatif - exemple de parcours de l'ensemble des propriétés d'un objet :

```
for (p in window) { // p est une chaîne de caractères
  document.write('window.'+p+'='+window[p]+'<br>');
}
```

Tableaux et objets

```
<!-- propriete_obj.html -->
<HTML><HEAD>
</HEAD><BODY>
  <SCRIPT LANGUAGE="JAVASCRIPT1.2">
   // p est une chaîne de caractères
   for (p in window.location) {
     document.writeln('<b>location.' + p + '</b>=' + window.location[p] +
         '<br>');     }
   document.writeln("<hr>");
   for (p in window.document) {
     document.writeln('<b>document.' + p + '</b>=' + window.document[p] +
         '<br>');     }
   document.writeln("<hr>");
   for (p in window) {
     document.writeln('<b>window.' + p + '</b>=' + window[p] + '<br>');      }
  </SCRIPT>
</BODY></HTML>
```


Les objets prédéfinis

Les objets prédéfinis (1)

- L'objet **Global** qui définit un ensemble de propriétés et méthodes communes à tous les objets
  - les méthodes et propriétés de cet objet n'appartiennent à aucune classe et cet objet n'a pas de nom
  - la seule façon de faire référence à cet objet est *this*
  - chaque variable ou fonction globale est une propriété de *Global*
  - sur un navigateur client, l'objet **window** est l'objet *Global* auquel il a été ajouté certaines propriétés et méthodes
  - propriétés de *Global* : **Infinity, NaN, undifined**

Les objets prédéfinis (2)

- L'objet **Global** - quelques méthodes
  - **parseFloat(s)** et **parseInt(s,base)** pour les conversions string->réel ou entier
  - **isNaN(expr)** pour tester une expression numérique
  - **eval(s)** permet d'évaluer une expression JavaScript contenu dans la chaîne de caractère s
  - **escape(s)** et **unescape(s)** pour le codage de type URL-encodé

```
escape("Hello World!"); // retourne "Hello%20World%21"
```

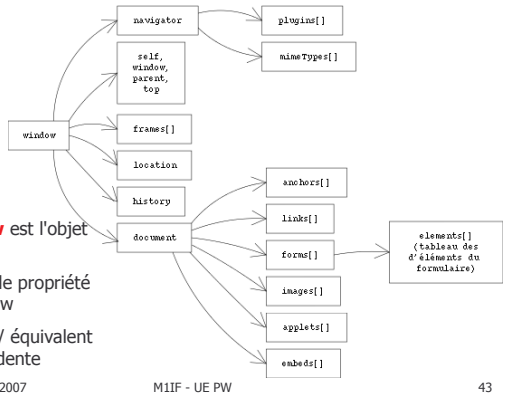

Les objets prédéfinis (3)

- Les classes prédéfinies du noyau JavaScript
  - **Array** - objet pour créer les tableaux
  - **Boolean** - objet pour créer un Booléen (true ou false)
  - **Date** - contient des méthodes de traitement de la date
  - **Function** - objet pour créer des fonctions
  - **Math** - contient des méthodes et des propriétés mathématiques (sinus, cosinus, racine carrée...)
  - **Number** - objet pour créer des nombres
  - **Image** - objet de gestion dynamique des images
  - **Option** - permet la gestion des listes créées avec <select> dans les formulaires
  - **RegExp** - pour l'utilisation des expression régulières
  - **String** - objet pour créer des chaînes

Les objets prédéfinis (4)

- Les objets instanciés automatiquement lors du démarrage du *browser*
  - permettent d'accéder à des informations concernant le navigateur client, les documents HTML affichés, l'écran de la machine
  - la classe **Navigator**
    - une seule instance -> objet *navigator*
    - infos sur nom, version, plug-ins installés,...
  - la classe **Window**
    - une instance par fenêtre et *frame* du document HTML
    - accès à tous les objets créés par des balises HTML
  - la classe **Screen**
    - une seule instance -> objet *screen*
    - infos sur largeur et hauteur en pixels, nombre de couleurs disponibles,...

Hiérarchie des objets du navigateur



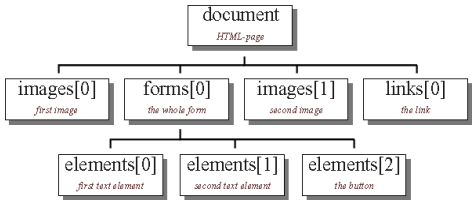
L'objet **window** est l'objet racine :

var t=10; // nelle propriété de l'objet window

window.t=10; // équivalent à la ligne précédente

Hiérarchie des objets du navigateur

- Une image est contenue dans un document qui est lui même contenu dans une fenêtre...
- Un champ de saisie est contenu dans un formulaire qui lui même est contenu dans un document...



L'objet *navigator* (1)

- **navigator.appName** = *Netscape* nom du browser permet de différencier les navigateurs
- **navigator.appVersion** = *4.7 [fr] (Win95; I)* informations sur la plate-forme d'exécution
- **navigator.language** = *fr* (2 caractères)
- **navigator.platform** = *Win32* type de machine
- **navigator.appCodeName** = *Mozilla* nom de code
- **navigator.userAgent** = *Mozilla/4.7 [fr] (Win95; I)* informations générales envoyées au serveur HTTP à chaque requête du navigateur

L'objet *navigator* (2)

- **navigator.plugins** = [object PluginArray] tableau des plug-ins installés sur le navigateur

```
var p0 = navigator.plugins[0].name;
document.write(navigator.plugins[p0].description);
```

- **navigator.mimeTypes** = [object MimeTypeArray] tableau des types mimes reconnus par le navigateur
  - mimeTypes[0].type nom du type MIME ('image/gif')
  - mimeTypes[0].description description du type
  - mimeTypes[0].enabledPlugin référence vers le plug-in qui gère ce type
  - mimeTypes[0].suffixes listes des suffixes de fichiers gérés par ce type

Les problèmes d'incompatibilité

```
// Vérifie que le navigateur soit au minimum IE4 ou Netscape4
function verifieNav() {
 var oldNav = true;
 navName = navigator.appName;
 navVer = parseInt(navigator.appVersion);
 if (navName == "Netscape" && navVer >= 4)
  oldNav = false;
 else if (navName == "Microsoft Internet Explorer" && navVer >= 4)
  oldNav = false;
 if (oldNav)
  alert("Passez à une version plus récente de votre navigateur !");
}
```

```
if (navigator.appName == "Netscape") {
// Code à exécuter avec le navigateur Netscape
} else {
// Code à exécuter avec les autres navigateurs
}
```


L'objet *screen*

- **screen.height** hauteur de l'écran en pixels
- **screen.width** largeur de l'écran en pixels
- **screen.availHeight** nombre de pixels disponibles verticalement (sans les barres de tâches, ...)
- **screen.availWidth** idem horizontalement
- **screen.pixelDepth** nombre de bits utilisés pour coder la couleur en pixels
- **screen.colorDepth** nombre de couleurs disponibles

La classe Window (1)

- Un objet *window* pour chaque fenêtre ou cadre (zone) ouverte par le navigateur
- 3 propriétés de base
  - ***window.history*** qui contient un tableau des URL déjà visitées dans la zone (historique)
  - **window.location** qui contient les caractéristiques de l'URL de la zone
  - **window.document** qui contient les caractéristiques et tous les objets de la zone

La classe Window (2)

- Accès aux objets contenus dans l'objet window
  - référence directe à l'objet window de la zone
    ```
    window.document.bgcolor = 'blue';
    ```
  - référence à un objet window d'une autre zone par l'intermédiaire d'une propriété
    ```
    window.top.cadre_droit.document.bgcolor = 'blue';
    window.parent.frames[1].document.bgcolor = 'blue';
    ```
  - référence à un objet window d'une autre zone par l'intermédiaire d'une variable
    ```
    var new_window=window.open('test.html');
    new_window.document.bgcolor = 'blue';
    ```


Principales propriétés de window (1)

- **document** contient des méthodes et informations sur le document (voir après)
- **history** permet d'avoir accès à l'historique d'une page (voir après)
- **location** permet d'avoir accès à toutes les informations de l'URL (voir après)
- **external** cet objet permet d'avoir accès à certaines propriétés du navigateur et de les modifier, telles que la page d'accueil, les favoris (IE uniquement)

Principales propriétés de window (2)

- **frames[]** permet d'avoir accès aux cadres ; on peut également utiliser directement le nom du cadre
- **parent, self, top** permettent d'accéder aux autres cadres
- **defaultStatus** permet de préciser la valeur par défaut du texte à afficher dans la barre d'état (quand la souris est sur le fond de la zone)
- **status** permet de préciser la valeur du texte à afficher dans la barre d'état à un instant donné

Principales propriétés de window (3)

- **closed** booléen qui permet de savoir si la zone a été fermée
- **opener** retourne l'objet window qui a ouvert cette fenêtre (avec la méthode open())
- Autres propriétés relatives aux caractéristiques de la zone (dimensions, positionnement)

  innerHeight, innerWidth, outerHeight, outerWidth,

  pageXOffset, pageYOffset, screenX, screenY

Principales méthodes de window (1)

- La méthode **open()**

  win = window.open(URL, nom_zone, options);
  - options booléennes (yes ou no) : fullscreen, menubar, toolbar, scrollbars, status, resizable
  - options en pixels : height, width, left, top
- La méthode **close()**

  window.close();
  - demande de confirmation à l'utilisateur si la zone n'a pas été ouverte avec open()
- La méthode **alert()**

  window.alert(message);
  - ouvre une fenêtre d'alerte où s'affiche message avec un bouton OK qui permet de fermer la fenêtre

Principales méthodes de window (2)

- La méthode **focus()**
  window.focus(); rend la zone active
- La méthode **blur()**
  window.blur(); rend la zone non active
- La méthode **prompt()**
  chaine = window.prompt(message, valeur_defaut);
  - ouvre une fenêtre composée d'un message et d'un champ de saisie et retourne la valeur saisie
- La méthode **confirm()**
  bool = window.confirm(message);
  - ouvre une fenêtre avec message, boutons OK et Annuler et retourne *true* (OK) ou *false* (Annuler)

Principales méthodes de window (3)

- La méthode **back()**
  window.back(); renvoie sur la page précédente
- La méthode **forward()**
  window.forward(); renvoie sur la page suivante
- La méthode **home()**
  window.home(); renvoie sur la page d'accueil du browser
- La méthode **stop()**
  window.stop(); stoppe le chargement de la zone
- La méthode **find()**
  chaine = window.find(chaine, casse, sens);
  - recherche la chaîne dans la zone

Principales méthodes de window (4)

- La méthode **print()**
  permet d'imprimer le document courant comme si l'utilisateur avait cliqué sur le bouton "imprimer"
- Les méthodes **setTimeout()** et **clearTimeout()**
  timeout = window.setTimeout(codeJS, msec);
  timeout = window.setTimeout(func, msec, arg1,...,argN);
  window.clearTimeout(timeout);
- Les méthodes **setInterval()** et **clearInterval()**
  idem mais exécutions périodiques toutes les msec
  interval = window.setInterval(codeJS, msec);
  interval = window.setInterval(func, msec, arg1,...,argN);
  window.clearInterval(interval);

DHTML !

Principales méthodes de window (4)

- Les méthodes **moveBy()** et **moveTo()**
  permettent de déplacer la zone
  window.moveBy(dx,dy); // dx pixels vers la droite, dy vers le bas
  window.moveTo(x,y); // (x,y) = coin supérieur gauche
- Les méthodes **resizeBy()** et **resizeTo()**
  permettent de redimensionner la zone
  window.resizeBy(dw,dh); // dw pixels d'augmentation de la largeur...
  window.resizeTo(l,h); // largeur et hauteur en pixels
- Les méthodes **scrollBy()** et **scrollTo()**
  permettent de faire défiler la zone
  window.scrollBy(dx,dy); // défilement de dx pixels vers la droite...
  window.scrollTo(x,y); // le point (x,y) du document est affiché dans le coin supérieur gauche de la fenêtre

Les événements associés à window

- **onFocus** - exécuter du code JS quand la fenêtre devient active
  <BODY onFocus="codeJS"> ou
  window.onfocus=f; // f est une fonction ou null
- **onBlur** - idem mais quand la fenêtre devient non active
- **onerror** - exécuter une fonction quand survient une erreur dans le code JavaScript
  window.onerror=f; // f est une fonction ou null
- Et aussi : **onload**, **onunload** , **onmove**, **onresize**

La propriété *history* de window

- **window.history** est un objet qui permet d'avancer ou reculer dans l'historique du browser associée à la zone
- Trois méthodes : **back()**, **forward()** et **go(int)**
  window.history.back(); ou window.history.forward();
  window.history.go(-2); retourne 2 documents en arrière
  window.history.go(3); retourne 3 documents en avant
  window.history.go(0); recharge le document courant
  Retour

La propriété *location* de window (1)

- **window.location** est un objet qui contient des informations sur l'URL de la zone
- Deux méthodes : **reload()** et **replace(URL)**
  window.location.reload(); recharge le document courant
  window.location.replace(URL); remplace le document courant par celui dont l'URL est passé en paramètre mais sans changer l'historique du navigateur
- Les propriétés
  - **window.location.href** contient la totalité de l'URL ; permet de charger une URL dans une autre fenêtre

La propriété *location* de window (2)

- Les propriétés (suite…)
  - **window.location.hash** partie de l'URL située après le #
  - **window.location.host** nom du serveur et port
  - **window.location.hostname** nom du serveur
  - **window.location.port** numéro du port sur le serveur
  - **window.location.pathname** entre le nom d'un script CGI et le ?
  - **window.location.protocol** nom du protocole
  - **window.location.search** partie de l'URL située après le ? ; permet de récupérer les données d'un formulaire en méthode GET

La propriété *document* de window (1)

- Les principales propriétés
  - titre du document : **document.title**
  - couleur du texte : **document.fgColor**
  - couleur du fond : **document.bgColor**
  - couleur de liens : **document.linkColor**
  - couleur de liens visités : **document.vlinkColor**
  - couleur de liens activés : **document.alinkColor**
  - adresse du document : **document.URL**
  - URL du document précédent : **document.referrer**
  - date de dernière modification : **document.lastModified**
  - il y a beaucoup d'autres propriétés liées aux balises <body>, <img>, <form>, <a>, <area>, …

La propriété *document* de window (2)

- Les principales propriétés (de type collection)
  - **anchors[]** tableau des liens internes <A NAME=...>
  - **applets[]** tableau des applets <APPLET CODE=...>
  - **embeds[]** tableau des objets insérés <EMBED SRC=...>
  - **forms[]** tableau des formulaires <FORM>
    - **elements[]** tableau des composants du formulaire <INPUT...>, <SELECT...>, ...
  - **images[]** tableau des images <IMG SRC=...>
  - **links[]** tableau des liens (hypertextes et images cliquables) <A HREF=...>
  - all[] référence tous les éléments HTML du document dans l'ordre d'apparition dans le source (IE seulement)
  - layers[] référence tous les blocs DIV (Netscape seulement)

La propriété *document* de window (3)

- Accès à un objet du document

```
<form name="mon_form">
  <input type="text" name="t1" value="z1">
  <input type="submit" name="b1" value="go">
</form>
<!-- l'attribut id permet de donner un nom à un élément -->
<a id="mon_lien" href="f1.html">lien1</a>
<script>
// forms[0], forms['mon_form'] et mon_form sont équivalents
  document.forms[0].elements[0].value = 'z2';
  document.mon_form.b1.value = 'Envoyer'; // car attribut name
  document.links[0].href = document.URL; // standard
  if (navigator.appName == "Netscape")
    document.links['mon_lien'].target='_blank'; // Netscape seulement
  if (navigator.appName == "Microsoft Internet Explorer")
    document.all.mon_lien.target='_blank'; // IE seulement
</script>
```

La propriété *document* de window (4)

- Accès au contenu de <p>, , … ?
- Principales méthodes d'accès à un objet du document (DOM niveau 2)
  - **getElementById("id1")** retourne **l'**élément du document dont l'attribut `id` vaut "id1" (null sinon) ; id1 doit être unique
    document.getElementById("mon_lien").target='_blank';
  - **getElementsByName("mon_form")** retourne le tableau des éléments dont l'attribut `name` vaut "mon_form" (tableau de longueur nulle sinon)
  - **getElementsByTagName("h1")** retourne le tableau des éléments du document de type "h1" (l'argument est un nom de balise)
    - permet d'atteindre tous les liens du document, ...

La propriété *document* de window (5)

- Principales méthodes
  - **write(arg1, ..., argN)** affiche les chaînes ou les arguments traduits en chaînes
  - **writeln(arg1, ..., argN)** idem que write() mais ajoute un retour chariot non interprété par HTML
  - **open()** crée un nouveau document (vide l'ancien) dans lequel on peut écrire avec write(), ne pas confondre avec la création d'une fenêtre !
  - **close()** ferme le document et provoque l'affichage de ce qui a été écrit avec write(), ne pas confondre avec la fermeture d'une fenêtre !
  - **getSelection()** retourne une chaîne de caractères contenant le texte sélectionné dans le document

La propriété *document* de window (6)

```
<!-- index.html -->
<HTML><HEAD>
  <SCRIPT LANGUAGE="JAVASCRIPT" SRC="function.js">
  </SCRIPT>
</HEAD><BODY>
  <div id="title1" style="position:absolute; top:50; left:20;">
    <h1>DHTML...</h1>
  </div>
  <div style="position:absolute; top:10; left:250;">
    <a href="javascript:changeText();">Modifier le texte</a><br>
    <a href="javascript:changeColor();">Modifier la couleur</a><br>
    <a href="javascript:changePosition();">Déplacer le bloc</a>
    <a href="javascript:AfficheBloc();">Affiche le bloc</a>
    <a href="javascript:MasqueBloc();">Masque le bloc</a>
  </div>
  <script language="JavaScript">
    var elm = document.getElementById('title1');
  </script>
</BODY></HTML>
```

La propriété *document* de window (7)

```
// function.js
function changeText() {
        elm.innerHTML="<b><i>... est dynamique !</i></b>";
        /* outerHTML permet de tout réécrire (marqueurs compris)
        elm.outerHTML="<div id=\"title1\" style=\"background-color:
                gold;\">... est dynamique !</div>" */
}

function changeColor() {
        elm.style.color="#00FF00";
}

function changePosition() {
        elm.style.top="400";
        elm.style.left="400";
}
```


Les événements

Les événements (1)

- Des événements sont associés aux actions de l'utilisateur et à certaines balises
  - notion de couple (objet, événement)
  - permet d'exécuter du code JavaScript lorsque l'événement se produit sur l'objet associé
- Exemples d'événements
  - un lien hypertexte est sensible au clic ou au passage de la souris
  - un formulaire est sensible au fait d'être soumis
  - une page est sensible au fait d'être chargée

Les événements (2)

- Deux façons d'associer une action à un couple (objet, événement)
  - utiliser des attributs de balise spécifiques associés aux événements

```
<a href="index.html" onMouseOver="alert('bonjour');">cliquez ici</a>
```

  - indiquer pour un élément et un événement donnés la fonction qui devra être exécutée

```
<body><a id="l1" href="index.html">cliquez ici</a></body>
<script>
  function bonjour() { alert('bonjour'); };
  document.links[0].onmouseover=bonjour;
</script>
```

Les événements (3)

- La classe *event*
  - quand un événement se produit, le navigateur crée un objet *event* avec les propriétés suivantes :

  type chaîne indiquant le type de l'événement
  target objet sur lequel l'événement est déclenché
  layerX, layerY position de la souris en pixels dans la couche
  pageX, pageY position de la souris en pixels dans la page
  screenX, screenY position de la souris en pixels dans l'écran
  - on accède à l'objet *event* par
    - *event*.xxxx dans l'attribut de la balise capturant l'evt
    - *event* qui peut être passé en paramètre de la fonction associée à l'evt

Attribut	Balises concernées	Déclenchement de l'événement
onAbort	IMG	L'utilisateur interrompt le chargement d'une image
onBlur	BODY, FRAMESET, <INPUT txt,pwd>, <TEXTAREA>	Un élément n'est plus actif (perd le focus)
onChange	<INPUT txt,pwd>, <TEXTAREA>	L'utilisateur à changer la valeur d'un champ de saisie
onClick	<A>, <AREA>, <INPUT others>	L'utilisateur clique sur un objet
onDbClick	<A>, <AREA>, <INPUT others>	L'utilisateur double-clique sur un objet
onError	IMG ou window.onerror	Une erreur se produit lors du chargement d'une image ou de l'exécution d'un code JavaScript
onFocus	BODY, FRAMESET, <INPUT txt,pwd>, <TEXTAREA>	Un élément devient actif (reçoit le focus)
onKeyDown	<INPUT txt,pwd>, <TEXTAREA>	L'utilisateur appuie sur une touche
onKeyPress	<INPUT txt,pwd>, <TEXTAREA>	L'utilisateur appuie, puis relâche une touche
onKeyUp	<INPUT txt,pwd>, <TEXTAREA>	L'utilisateur relâche une touche
onLoad	BODY, FRAMESET, IMG	Un document ou une image commence à se charger
onMouseDown	<A>, <AREA>, <INPUT others>	L'utilisateur appuie sur un bouton de la souris
onMouseMove	seulement interceptable	L'utilisateur déplace la souris
onMouseOut	<A>, <AREA>, <INPUT others>	L'utilisateur fait ressortir la souris de l'objet
onMouseOver	<A>, <AREA>	L'utilisateur place la souris sur un objet
onMouseUp	<A>, <AREA>	L'utilisateur relâche un bouton de la souris
onMove	BODY, FRAMESET	L'utilisateur déplace une fenêtre
onReset	FORM	Un formulaire est réinitialisé par un bouton reset
onResize	BODY, FRAMESET	L'utilisateur redimensionne une fenêtre ou un cadre
onScroll	BODY, FRAME	L'utilisateur se sert des barres de défilement de la page
onSelect	<INPUT txt,pwd>, <TEXTAREA>	L'utilisateur sélectionne du texte dans un champ
onSubmit	FORM	Un formulaire est soumis (bouton submit)
onUnload	BODY, FRAMESET	Un document se décharge (chargement d'un autre doc)

Les événements (5)

- Pour la plupart des événements, le navigateur possède déjà un comportement par défaut (ex: chargement du document lors du click sur un lien)
- Quand un événement est intercepté, le navigateur exécute d'abord le traitement JavaScript défini dans la page puis le traitement par défaut
- Pour empêcher le traitement par défaut, il faut retourner la valeur *false* :

```
<a href="l.html" onClick="return confirm('continue ?');">lien</a>
```


Les objets du noyau JavaScript

L'objet String

- Propriété : length
- Principales méthodes de manipulation de chaînes
  - indexOf(chaine,index),
  - substring(débuf,fin+1), charAt(n),
  - lastIndexOf(chaine), toLowerCase(),
  - toUpperCase(), split(), toString()

```
var T = Bonjour;
T.indexOf ("o"); --> 1
T.lastIndexOf("o"); --> 4
T.charAt(3); --> j
T.substring(3,7); --> jour
T.toUpperCase(); --> BONJOUR
```


L'objet Math

- Propriétés : constantes mathématiques
  - E, PI, SQRT2, SQRT1_2 (1/SQRT2), LN2, LN10, LOG2E, LOG10E
- Méthodes : fonctions usuelles
  - trigonométriques : cos, sin, tg, acos, atan, asin, atan2
  - abs (valeur absolue)
  - ceil (entier sup), floor (entier inf), round (entier + proche)
  - exp, log, sqrt, pow(x,a)
  - max (a,b), min (a,b)
  - random --> 0 < r < 1

L'objet Array

- Propriété : length (nombre d'éléments)
- Principales méthodes :
  - **join** concatène tous les éléments en une chaîne de caractères séparés par une virgule ou un séparateur passé en argument
  - **reverse** transpose le tableau (1<->N 2<-> N-1...)
  - **sort** trie les éléments dans l'ordre lexicographique ou selon la relation d'ordre passée en argument (fonction de comparaison fournie par l'utilisateur)

Quelques exemples classiques

Changement de la couleur de fond

```
<!-- ex1.html -->
<HTML><BODY><CENTER>
<FORM>
<SELECT NAME="couleur"
onChange="document.bgColor=this.options[this.selectedIndex].value;">
  <OPTION selected VALUE="" >Choisissez une couleur de fond
  <OPTION VALUE="blue">Bleu
  <OPTION VALUE="aquamarine">Vert clair
  <OPTION VALUE="darkred">Marron
  <OPTION VALUE="gold">Or
  <OPTION VALUE="red">Rouge
  <OPTION VALUE="yellow">Jaune
  <OPTION VALUE="hotpink">Rose
  <OPTION VALUE="#000000">Noir
  <OPTION VALUE="#FFFFFF">Blanc
</SELECT></FORM></CENTER></BODY>
```

alert() et formulaires

```
<!-- ex2.html -->
<HTML><HEAD><SCRIPT LANGUAGE="Javascript">
<!--
function pop_up(msg)
{
 alert (msg);
} -->
</SCRIPT></HEAD><BODY><CENTER>
<A HREF="javascript:pop_up('Message 01 !');">MESSAGE 01</A>
<FORM ACTION="toto.html" onSubmit="confirm('Soumission');">
  <INPUT type="button" value="MESSAGE 02" onClick="pop_up('Message
02 !');">
  <INPUT type="submit" value="Envoyer" onClick="pop_up('Message 03
!');">
</FORM>
</CENTER></BODY></HTML>
```

Conversions Francs/Euros

```
<!-- ex3.html -->
<HTML><HEAD><SCRIPT LANGUAGE="JavaScript">
function f2e (form)
{ form.eur.value = form.fra.value/6.55957; }
function e2f (form)
{ form.fra.value = form.eur.value*6.55957; }
</SCRIPT></HEAD><BODY>
<FORM name="fe">
  <INPUT TYPE="text" name="fra" size="15"
onChange="f2e(document.fe);"> F<br>
  <INPUT TYPE="text" name="eur" size="15"
onChange="e2f(document.fe);"> Euros
</FORM>
Calculs effectués sur la base de<BR>
1F = 6.55957 Euros
</BODY></HTML>
```

Affichage de l'heure

```
<!-- ex4.html -->
<HTML><HEAD>
<SCRIPT LANGUAGE="JavaScript1.2">
var t=null;
function affiche_heure() {
  var now = new Date();
  var hrs = now.getHours();
  var min = now.getMinutes();
  var sec = now.getSeconds();
  document.heure.h.value = hrs;
  document.heure.m.value = min;
  document.heure.s.value = sec; }
function start() { t = setInterval(affiche_heure,1000); }
function stop() { clearInterval(t); }
function restart() { window.location.reload(); }
</SCRIPT></HEAD>
<BODY onLoad="start();">
<FORM NAME="heure">
  <INPUT TYPE="text" NAME="h" SIZE=2>
  <INPUT TYPE="text" NAME="m" SIZE=2>
  <INPUT TYPE="text" NAME="s" SIZE=2>
  <INPUT TYPE="button" VALUE="STOP" onClick="stop();">
  <INPUT TYPE="button" VALUE="CONTINUE" onClick="restart();">
</FORM></BODY></HTML>
```

Test d'une adresse mail

```
<!-- ex5.html -->
<HTML><HEAD>
<SCRIPT LANGUAGE="JavaScript">
function verifMail(adresse) {
  if  (adresse == "" ) {
    alert ("Vous n'avez pas saisi votre adresse");
    document.forms[0].email.focus(); return false;
  }
  arob = adresse.indexOf('@');
  if (arob == -1) {
    alert ("Adresse incorrecte (pas de @)"); document.forms[0].email.value="";
    document.forms[0].email.focus(); return false; }
  site = adresse.substring(arob+1);
  if (site == "ens-lyon.fr" ) { return confirm('c\'est bon ! On envoie ?'); }
  else { alert ("site : "+ site +" incorrect"); return false; }
}
</SCRIPT></HEAD><BODY>
<FORM name="formulaire" action="toto.html" onSubmit="return
verifMail(document.formulaire.email.value);">
Entrez votre adresse e-mail : <BR>
<INPUT TYPE="text" NAME="email" SIZE="20" MAXLENGTH="40">
<INPUT TYPE="submit" VALUE="Envoyer">
</FORM></BODY></HTML>
```


Création d'une fenêtre

```
<!-- ex6.html -->
<HTML><HEAD><SCRIPT LANGUAGE="JavaScript">
var maFen=null;
function ouvrir() {
  var options = "width=300,height=30";
        // window.open(url, nom-pour-target, caractéristiques)
        maFen = window.open("", "Ma_Fenetre", options);
        maFen.document.open();
        maFen.document.write("Exemple JavaScript");
        maFen.document.close();
}
function fermer() {
        maFen.close();
}
</SCRIPT></HEAD><BODY>
Passez sur ce <A HREF="" onMouseOver="ouvrir();"
                         onMouseOut="fermer();">lien </a>
sans cliquer<br>
Il doit se passer quelque chose... et ensuite sortez du lien
</BODY></HTML>
```


Texte défilant

```
<!-- ex7.html -->
<HTML><HEAD><SCRIPT LANGUAGE="JavaScript">
var posBan1=0, ban1, msgBan1, sp="";
function banniere1(delai) {
  for (i=1;i<35;i++) sp=sp+" ";
  msgBan2=sp+msgBan1+sp;
  ban1 = setInterval(banniere2,delai);
}
function banniere2(delai) {
  if (posBan1 >= 2*msgBan1.length)    posBan1 = 0;
  document.formBan1.Fbanniere1.value =
          msgBan2.substring(posBan1,posBan1+msgBan1.length);
  posBan1++;
}
</SCRIPT></HEAD><BODY onLoad="msgBan1='Voici un exemple de
texte defilant '; banniere1(500);" onUnload="clearTimeout(ban1);">
<FORM NAME="formBan1">
  <INPUT TYPE="text" NAME="Fbanniere1" SIZE="35">
</FORM></BODY></HTML>
```


JavaScript et les images

```
<!-- ex8.html -->
<HTML><HEAD><SCRIPT LANGUAGE="JavaScript">
 var img = new Image(400,400);
 // permet d'accélérer le chargement de l'image
 // (pré-chargement de img1.gif dans le cache du navigateur)
 img.src = "img1.gif"
</SCRIPT></HEAD><BODY>
<IMG src="img0.gif" width="100" height="100"
    onMouseOver="this.src='img1.gif';this.width=400;this.height=400;"
    onMouseOut="this.src='img2.gif';this.width=200;this.height=200;">
</BODY></HTML>
```