http://www-adele.imag.fr/users/Didier.Donsez/cours

Le Langage JavaScript / ECMAScript

Didier DONSEZ
Université Joseph Fourier
PolyTech'Grenoble – LIG/ADELE
Didier.Donsez@imag.fr,
Didier.Donsez@ieee.org

Motivations

- Ajout du contrôle au niveau du client
  - Exemple :
    - contrôle des entrées d'un formulaire avant sa soumission.

- Scripts embarqués dans des documents HTML
  - Contrôle des ressources du client
    - documents, formulaires, applets, …
    - ⇒Moins d'intervention du Serveur WWW
  - Génération dynamique de documents DHTML
    - layers
    - ⇒Evite les GIFs animés et les animations Shockwave

Plusieurs propositions

- 4 langages de scripting
  - JavaScript *(Netscape)*
    - initialement LiveScript, rien à voir avec Java
  - VBScript, Jscript *(MicroSoft)*
    - syntaxe Visual Basic pour VBScript
    - JScript a une syntaxe compatible avec JavaScript
  - ECMAScript
    - *(ECMA : European Computer Manufacturers Association)*
- DOM : Document Object Model *(W3C)*
  - description « objet » d 'un document HTML ou XML
  - exploitable par les langages de scripting
    - mais IE et Navigator ont chacun leur modèle de document

L ’élément HTML <Script>

- Scripting local au document apparaît
  - dans l’élément HEAD ou dans l’élément BODY

```
<SCRIPT LANGAGE="JavaScript">
<!-- Commentaire HTML  qui dissimule le contenu du script aux browsers
  … Instruction; ...
// qui ne supporte pas JS -->
</SCRIPT>
```

- Scripting externe
  - inclusion du code dans l’élément SCRIPT

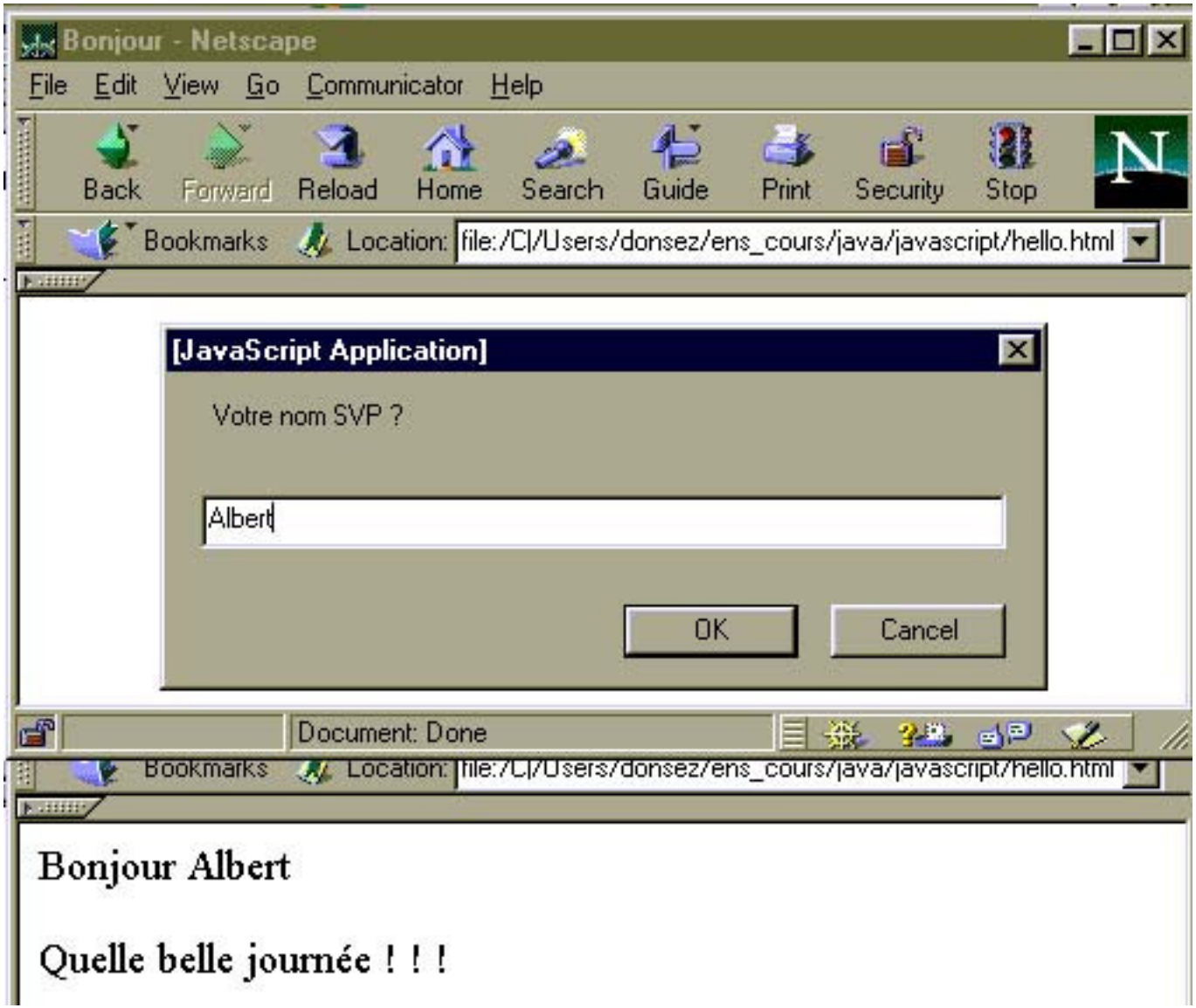
```
<SCRIPT LANGAGE="JavaScript" SCR="url/script.js">
// inclut le fichier script.js puis l’execute
// Ne pas mettre d’instructions : elles ne seraient pas exécutées !
</SCRIPT>
```

L ’élément HTML <Script>

```
<HTML><HEAD><TITLE>Bonjour</TITLE>
</HEAD><BODY>
<SCRIPT LANGAGE="JavaScript">
<!-- Dissimule le contenu du script aux browsers
  var nom = window.prompt("Votre nom SVP ?","");
  document.write("Bonjour " + nom);
// qui ne supporte pas JS -->
</SCRIPT>
<BR>
Quelle belle journée ! ! !
</BODY></HTML>
```

```
<HTML><HEAD><TITLE>Bonjour Script Externe</TITLE>
</HEAD><BODY>
<SCRIPT LANGAGE="JavaScript" SRC="http://www.mycomp.com/libscript.js">
// execute le contenu de libscript
</SCRIPT>
<BR>
Quelle belle journée ! ! !
</BODY></HTML>
```

Exemple


Bonjour - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: file:/C|/Users/donsez/ens_cours/java/javascript/hello.html
[JavaScript Application]
Votre nom SVP ?
Albert
OK
Cancel
Document: Done
Bookmarks Location: file:/C|/Users/donsez/ens_cours/java/javascript/hello.html
Bonjour Albert
Quelle belle journée ! ! !

Constantes

- 4 types de base
  - entier
    - 127 (base 10), 0755 (base 8), 0xFA15 (base 16)
  - flottant
    - 0.123, -0.4e5, .67E-89
  - booléen
    - true, false
  - chaîne
    - "L'étoile" // L'étoile
    - 'filante' // filante
    - "'filante'" // 'filante'
    - \b, \f, \n, \r, \t, \"

Typage et Variable

- Pas de déclaration des Variables
  - nbr = 10;
  - fl =3.141;
  - str1 = "L'étoile";
  - str2 = 'brille';
  - lien = "Home";
- Portée des Variables
  - Local (uniquement dans le script ou la fonction)
    - var vloc = 0 ;
  - Global (en tout point du document)
    - vglob = 0 ;

Expressions et Opérateurs

- Expressions
  - **Arithmétique**
    - (3+4) * (56.7 / 89)
  - **Chaîne**
    - "L'étoile" + " " + "filante"
  - **Logique**
    - temp == 37
    - h2o = (temp<100) ? "eau" : "vapeur";
    - h2o = (temp>0) ? ((temp<100) ? "eau" : "vapeur") : "glace";
- Opérateurs
  - **Affectations**
    - +=, -=, *=, /=, %=, &=, |=, <<=, >>=
  - **Comparaisons**
    - ==, !=, <, <=, >, >=
  - **Arithmétiques**
    - %, ++, --
  - **Logiques**
    - &&, ||, !
  - **Bit**
    - &, |, ^ (XOR), <<, >>, >>>

Structures de Contrôle

- *bloc* { }
- if else, switch case, for, while, break, continue, do while, export, import, labeled
- boucle for in
  - liste les propriétés d'un objet

Fonctions

- Définition et Appel

```
function nomfonction( param1, ..., paramN) {
  // code JavaScript
  return variable_ou_valeur ;
}
var res = nomfonction(var1, val2, varN) ;
```

  - **Remarque : passage des paramêtres par valeur**

- Exemple

```
function isHumanAge(age) {
 if ((age < 0) || (age > 120)) { return false ; } else { return true ; }
}
if ( ! isHumanAge(age) ) {
  alert("Vous ne pouvez pas avoir " + age + " ans !");
}
```

- Arguments

```
function somme() {
  var argv = somme.arguments;      var argc = somme.arguments.length;
  var result=0 ; for(var i=0 ; i < argc ; i++) { result += argv[i];  }  return result; }
somme(1,2,3) retourne 6 et  somme(2) est retourne 2
```

Fonctions prédéfinies

- **eval : évaluation d'expression**

```
angle = Math.PI / 2 ;  expr = "1 - Math.cos(angle)" ;
document.write( expr + " = " + eval(expr) )
```

- **parseInt, parseFloat**
  - convertion d'une représention chaîne d'un entier (dans une base) et d'un flottant
    - **parseInt("12", 10)), parseInt("1100", 2)), parseInt("0xC", 16)) // retournent 12**
    - **parseFloat("123") // 123**
    - **parseFloat("456abc") // 456**
    - **parseFloat("1abc23") // 1**
    - **floatValue= parseFloat("abc789"); // NaN**
    - **if (isNaN(floatValue)) { notFloat() } else { isFloat() }**
- **escape() , unescape()**
  - utilisées pour le codage des URL ou des Cookies

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

Le modèle objet de JavaScript

- Modèle de Prototype
  - Différent du modèle de classe (Java,C++, …)
- Contructeur
  - un objet instancié est « vierge » de propriétés
  - la fonction « constructeur » ajoute des propriétés d'un objet

```
function printPersonne() {
  document.write("Personne: "
     +this.nom+","+this.age);
}
function Personne(nom, age) {
  this.nom = nom ;
  this.age = age ;
  this.print = printPersonne;
  return this ; // OBLIGATOIRE
}
```

```
function printVehicule() {
  document.write("Vehicule: "
   +this.marque+","+this.age);
}
function Vehicule(marque, age) {
  this.marque = marque ;
  this.age = age ;
  this.print = printVehicule;
  return this ; // OBLIGATOIRE
}
```

Instanciation

- **`new`**
  - instancie un objet “ vierge ” et appelle un constructeur

```
pers1      = new Personne("Albert", 77);
pers2      = new Personne("Leo", 56);
voit3      = new Vehicule("Renault", 2);

// Construction dans JS1.2
voit3 = {nom:"Renault", age:2, print:printVehicule};
```

- **`this`**
  - donne accès aux propriétés de l’objet courant
- **`with`**
  - raccourci pour désigner les propriétés et les méthodes

```
with(document) {  write("La longueur du document est :" + length) ; }
// OU
document.write("La longueur du document est :” + document.length) ;
```

Accès aux propriétés d 'un objet (i)

- Propriété = Objet, Valeur ou Méthode
  - Opérateur de référence . et []
    - l'objet doit avoir les propriétés désignées
    - Dans une méthode, les propriétés de l'objet sont accessible par this.

```
pers1.age = 20;
pers1.salaire=5000;
pers1["salaire"]=10000;
voit3.age = 2;

function unAnDePlus() { this.age++ ; }
pers1.incrAge=unAnDePlus;              // ajout d'une nouvelle propriété
voit3.incrAge= pers1.incrAge;          // ajout d'une nouvelle propriété
pers1.incrAge();                       // appel de unAnDePlus() sur pers1
voit3. incrAge();                      // appel de unAnDePlus() sur voit3
pers1.print();                         // appel de printPersonne() sur pers1
voit3.print();                         // appel de printVehicule() sur voit3
```

Accès aux propriétés d 'un objet (ii)

- Structure de contrôle for in
  - Liste les propriétés de l'objet

```
function dumpobj(obj){
   var result= "", i= "";
   for(i in obj){ result += i + " = " + obj[i] + "<BR>" ; }
   return result ;
}

document.writeln( dumpobj(pers1));
document.writeln( dumpobj(pers1.incrAge));
document.writeln( dumpobj(voit3));
```

L ’héritage (i)

- La propriété `prototype`

```
function Employee () {
    this.name = "";
    this.dept = "general";
    return this;
}

function Manager () {
    this.reports = [];
    return this;
}
Manager.prototype = new Employee;

function WorkerBee () {
    this.projects = [];
    return this;
}
WorkerBee.prototype = new Employee;
```

```
paul = new Employee;
mark = new WorkerBee;
mark.name = "Doe, Mark";
mark.dept = "admin";
mark.projects = ["navigator"];
mark.bonus = 3000;
  // propriété ajoutée à mark uniquement
Employee.prototype.specialty = "none";
  // propriété ajoutée à mark et à paul

function Employee (name, dept) {
    this.name = name || "";
    this.dept = dept || "general";
}
noname = new Employee
carla = new Employee("Carla, Smith", "R&D");
```

L 'héritage (ii)

- Le test d'appartenance

```
function instanceOf(object, constructor) {
  while (object != null) {
    if (object == constructor.prototype) // constructeur de l'objet
      return true;
    object = object.__proto__;          // chainage de prototype
  }
  return false;
}
```

- Variable globale

```
var idCounter = 1;                      // variable globale
function Employee (name, dept) {
  this.name = name || "";               // valeur par défaut
  this.dept = dept || "general";
  this.id = idCounter++;
}
```

Les objets Tableaux

- Ce sont des objets avec les propriétés qui sont les indices

```
function Array(nbElements) {
// constructeur
  this.length = nbElements;
  for( var e = 0 ; e < this.length ; e++)
     { this[e] = 0 ; }
  return this ;
}

marque = new Array(10);
marque[0]= "Renault";
marque[1]="Peugeot";
marque[2]="Citroen";
marque[10]="Ford"; marque.lenght++;
```

```
var i;

for( i=0;i <marque.length;i++ ){
  document.write(i + " = " + marque[i] + "<BR>")
}

document.write("<HR>");
for( i in marque ){
  document.write(i + " = " + marque[i] + "<BR>")
}
```

Les objets Tableaux (JS1.2)

- Prototype prédéfinie
- Constructeur

```
marque = Array("Renault", "Peugeot", "Citroen");
```

- Méthodes d'Array
  - **concat, pop, push, shift, unshift, slice, splice, sort**
- Tableau d'objets

```
var vehicules = new Array(4);
vehicules[1] = new Vehicule(marque[1], 2);
vehicules[2] = new Vehicule(marque[2], 4);
vehicules[3] = new Vehicule(marque[3], 8);
vehicules[4] = new Vehicule("Ford", 8);
for( i=1;i <= vehicules.length;i++ ){
  document.write(i + " = "); vehicules[i].print();
  document.write("<BR>");
}
```

Les objets Math et RegExp

- Prototypes prédéfinis

Math

constantes mathématiques : `PI`,..., fonctions mathématiques :
`sin(x),cos(x),abs(x),random()...`

```
circonference = Math.PI * rayon * rayon;        abscisse     = r * Math.sin(angle);
ordonnee      = r * Math.cos(angle);
with(Math) {
   circonference = PI * rayon * rayon;
   abscisse      = r * sin(angle);
   ordonnee      = r * cos(angle);
}
```

RegExp

expression régulière

```
re = /ab+c/
re = new RegExp("ab+c")
regexp = /pattern/[i|g|gi]
regexp = new RegExp("pattern", ["i"|"g"|"gi"])
```

Objets Function et Date

- Function
  - Définition des méthodes des objets
    - évite de passer par une fonction nommée

```
Employee.setdept = function(dept) {  this.dept = dept; }
```

- Date
  - référentiel 1er Janvier 1970

```
aujourdhui                  = new Date();
jour_an2000_moins_1sec  = new Date("December, 1999 23:59:59");
jour_an2000                 = new Date("January 1, 2000 00:00:00");
jour_an2000                 = new Date("January 1, 2000");
jour_an2000                 = new Date("January 1, 2000");
jour_avant_an2000       = new Date(99,11,31,0,0,0); // Jan(=0)
jour_avant_an2000       = new Date(99,11,31);      // Dec(=11)
msParJour = 24*60*60*1000;
nbjour_avant_an2000 = new Date();
nbjour_avant_an2000 =   Math.round(( jour_an2000.getTime() - aujourdhui.getTime()
                                )/msParJour);
noel99= new Date();       noel99.setTime(Date.parse("Dec 25, 1999"));
```

Objets String (i)

- Création d'un objet sur affectation

```
objstr = "Le nouveau monde !";
```

- Manipulation
  - les caractères sont indicés de 0 à objstr.length-1

```
"Le nouveau monde !".charAt(1)                  // "e"
"Le nouveau monde !".indexOf("e")               // 1
"Le nouveau monde !".indexOf("e",2)             // 7
"Le nouveau monde !".lastIndexOf("e")           // 15
"Le nouveau monde !".substring(3,7)             // "nouveau"
"Le Nouveau Monde !".toLowerCase()              // "le nouveau monde !"
"Le Nouveau Monde !".toUpperCase()              // "LE NOUVEAU MONDE !"
```

  - Mais aussi
    - charCodeAt, concat,fromCharCode, match, search, slice, split

Objets String (ii)

- Formatage HTML
  - insertion de balisage (tag) HTML :
    - <BIG>,<BLINK>, <B> …
    - big(), blink(), bold(), fixed(), fontsize(), fontcolor(), italics(), link(URL) ou anchor(URL), small(), strike(), sub(), sup()

var html += "la " + "SNCF".anchor("http://www.sncf.fr")

var html += "la <A HREF=\"http://www.sncf.fr\">SNCF</A>"

JavaScript, le navigateur et les documents HTML

- Description en objet JavaScript
  - du navigateur
  - des fenêtres
  - des documents qui les composent
- Consultation et Modification des propriétés de ces objets par scripting
- Future direction :
  - DOM

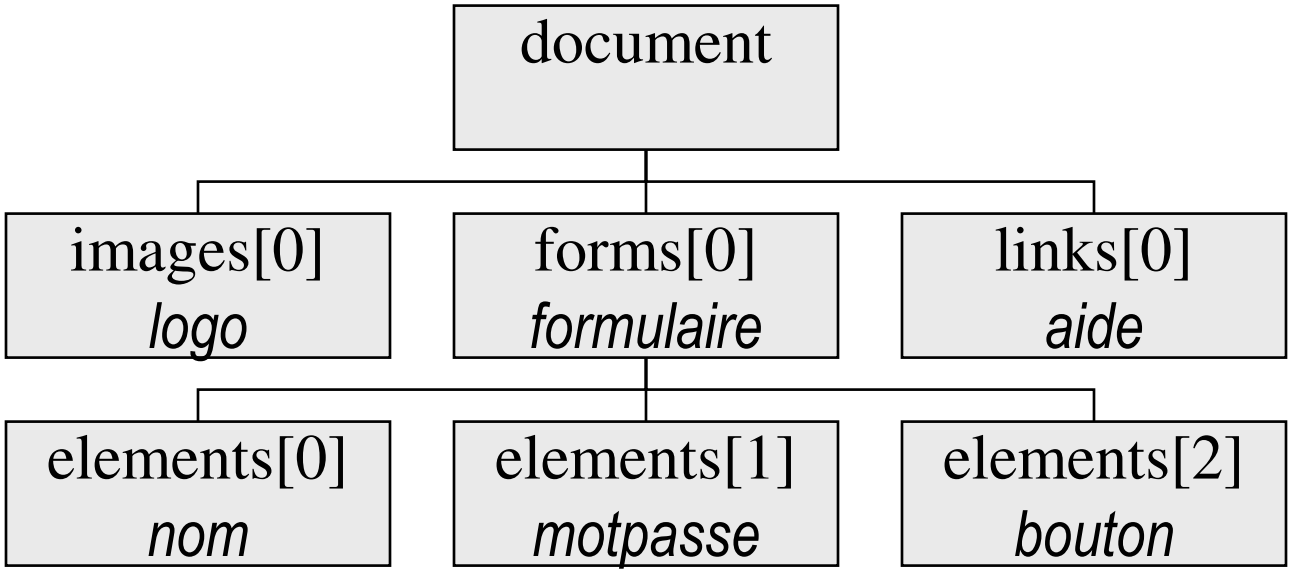
Document « Simple »

- HTML

```
<HTML><BODY>
<CENTER><IMG SRC="bienvenu.gif"
    HEIGHT=69 WIDTH=184
    name="logo"></CENTER>
<FORM name="formulaire">Nom:
<BR> <INPUT type="text"
    name="nom" value="">
<BR>Mot de Passe:
<BR> <INPUT type="password"
    name="motpasse" value="">
<CENTER><INPUT type="button"
    value="Valider" name="bouton"
    onClick="verif(this.parent)"></FORM>
</CENTER>
<DIV ALIGN=right><I><A HREF="aide1.htm »
    name="aide">Aide</A></I></DIV>
</BODY></HTML>
```

- Affichage

La représentation arborescence d 'objets d 'un document



- Exemple (script relatif au document)

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

- Propriété parent
  - référence l 'objet englobant

Les formulaires

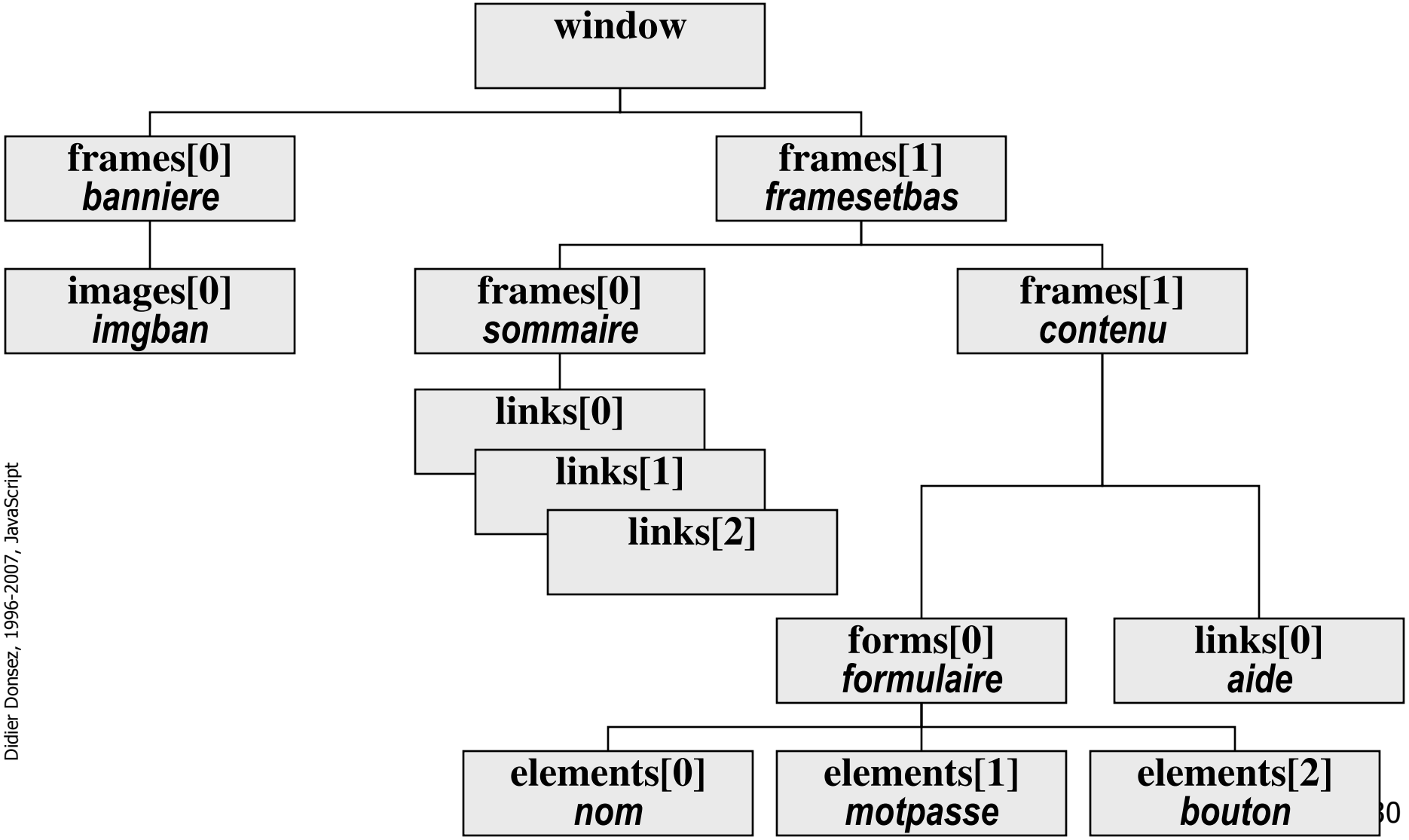
- l 'objet Form
  - action
  - elements[ ]
  - encoding
  - attribute
  - method
  - target
  - submit()
  - onSubmit
- Les éléments
  - un type d 'objet pour chaque type d 'entrée
  - Hidden
  - Password
  - Radio
  - Checkbox
  - Select
  - Text
  - Textarea
  - Submit
  - Reset

Les frames

- HTML

```
<FRAMESET ROWS="100,*">
    <FRAME NAME="banniere"
                SRC="bann.htm">
    <FRAMESET COLS="100,*"
      NAME="framesetbas">
      <FRAME NAME="sommaire"
                SRC="somm.htm">
      <FRAME NAME="contenu"
                SRC="cont1.htm">
    </FRAMESET>
</FRAMESET>
```

La représentation arborescence d 'objets d 'un frameset



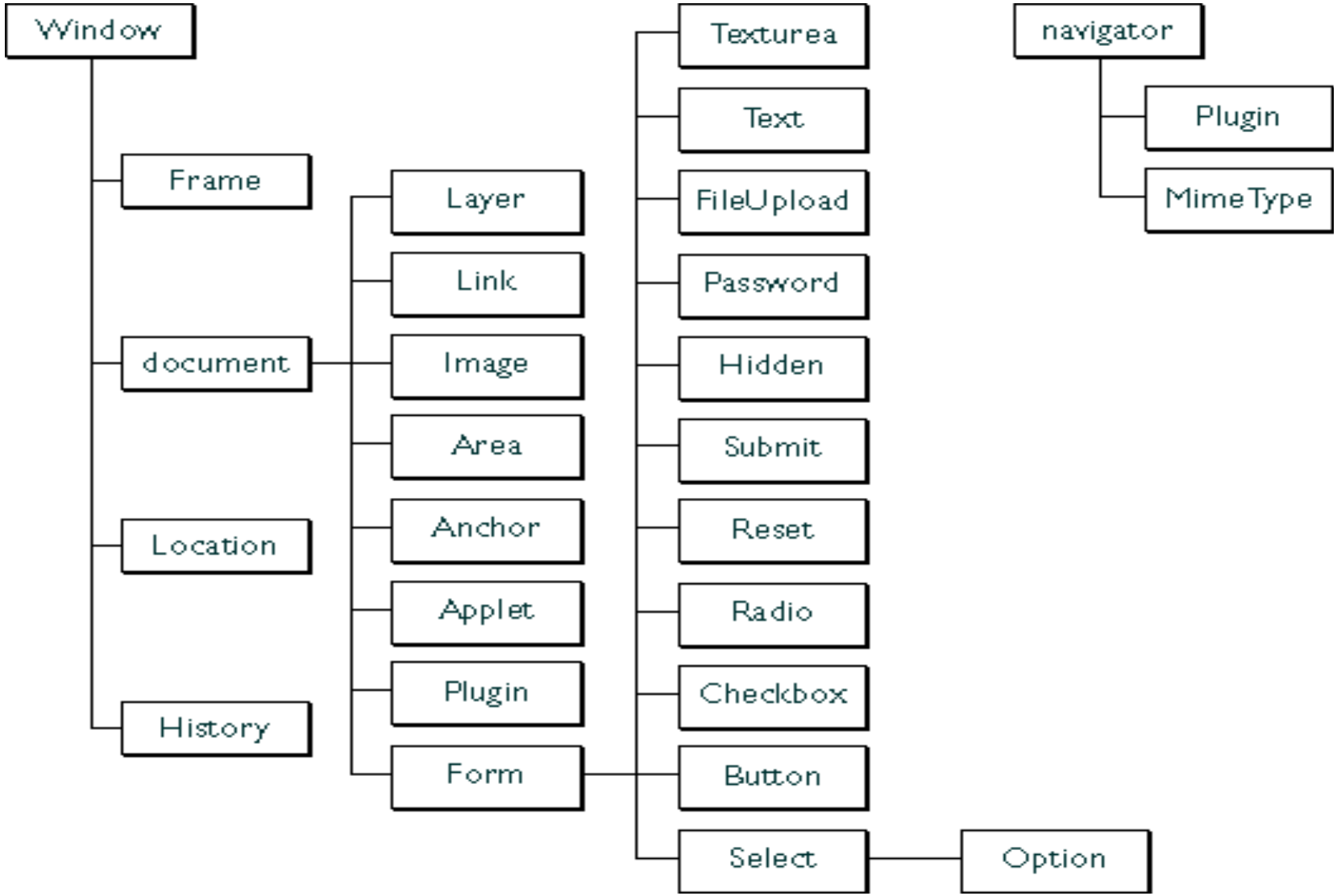
Exemple

```
<HTML><HEAD>
    <BASE TARGET="contenu">
<SCRIPT LANGUAGE="JavaScript">
function loaddoc(urlbanndoc, urlsommdoc, urldocfirst) {
    parent.parent.banniere.location.href= urlbanndoc;
    parent.sommaire.location.href= urlsommdoc;
    parent.contenu.location.href= urldocfirst;
}
</SCRIPT>
</HEAD><BODY>
 <A HREF="cont1.htm"> Accueil</A><BR>
 <A HREF="cont2.htm" TARGET="_top">Présentation</A><BR>
 <A HREF="javascript:loaddoc('banndoc.htm', 'sommdoc.htm', 'docdeb.htm')"
      TARGET="_self">Documentation</a>
</BODY>
</HTML>
```

L ’objet Window

- frames[ ]
- parent
- top
- self, window
- alert(message)
- close()
- confirm(message)
- open(url,name,features)
- toolbar=[yes,no,1,0]
- location=[yes,no,1,0]
- directories=[yes,no,1,0]
- status=[yes,no,1,0]
- menubar=[yes,no,1,0]
- scrollbars=[yes,no,1,0]
- resizable=[yes,no,1,0]
- width=pixels
- height=pixels
- prompt(message,response)
- id=setTimeout(expression,time)
- clearTimeout(id)

L 'objet Window



Création de fenêtre

- Création d 'une fenêtre

```
myWin= open("nouv.htm", "Nouvelle fenêtre",
    "width=400,height=300,status=no,toolbar=no,menubar=no");
```

- Création à la volée

```
<html><head><script language="JavaScript">
function openOnTheFly() {
  myWin= open("", "displayWindow", "width=500,height=400,"
                                  +"status=yes,toolbar=yes,menubar=yes");
  myWin.document.open();
  myWin.document.writeln("<html><head><title>On-the-fly</title></head><body>");
  myWin.document.writeln("Ce document HTML est généré par un script<BR>");
  myWin.document.writeln("<form><input type=button value='Close' onClick='Close()'>");
  myWin.document.writeln("</form></body></html>");
  myWin.document.close();
}
</script></head><body>
<form><input type=button value="On-the-fly" onClick="openOnTheFly()"></form>
</body></html>
```

Les timers

- 2 méthodes
  - ID=setTimeout(expr,delai);
  - clearTimeout(ID);
- Exemple : *un message glissant dans la Statusbar*

```
<html><head> <script language="JavaScript">
var scrtxt =  "Bienvenue sur ce site !";
var length = scrtxt.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 + " ";}
    scroller = scroller + scrtxt.substring(0, width - i + 1);
  } else { scroller = scroller + scrtxt.substring(pos, width + pos); }
  window.status = scroller; setTimeout("scroll()", 100);
}
</script>
</head><body onLoad="scroll()"><h1>Bienvenue</h1></body></html>
```

Les zones positionnables

- Motivations
  - Fragments du document HTML
    - Positionnable
    - Clippable
    - En couche superposable (dimension z)
    - Visible/Caché
    - Opaque/Transparence
  - Ces propriétés sont modifiables dynamiquement par scripting
- Eléments
  - LAYER (NSC/Mozilla seulement)
  - DIV (MSIE et NSC/Mozilla)

Les layers

- Propriétés
  - name="layerName", left=xPosition, top=yPosition z-index=layerIndex, width=layerWidth, clip="x1_offset, y1_offset, x2_offset, y2_offset", above="layerName", below="layerName", Visibility=show|hide|inherit, bgcolor="rgbColor", background="imageURL »
- Exemple

```
<html><head><script language="JavaScript">
function move() {
  if (pos < 0) direction= true; if (pos > 200) direction= false;
  if (direction) pos++ else pos--;
  document.layers["layer0"].left= pos; setInterval('move()', 20);
}
</script>
</head><body onLoad="setInterval('move()', 20)">
<ilayer name=layer0 left=0>
<font size=+1 color="#0000ff"><i>Bienvenue</i></font>
</ilayer></body></html>
```

Les layers imbriquées *(nested)*

```
<html><head>
<script language="JavaScript">
var pos= 0; var direction= false;
function moveNclip() {
  if (pos<-180) direction= true;
  if (pos>40) direction= false;
  if (direction) pos+= 2 else pos-= 2;
  document.layers["clippingLayer"].layers["imgLayer"].top= 100 + pos;
  setInterval('moveNclip()', 20);
}
</script>
</head><body onLoad="setInterval('moveNclip()', 20);">
<ilayer name="clippingLayer" z-index=0 clip="20,100,200,160" top=0 left=0>
  <ilayer name="imgLayer" top=0 left=0>
  <img name=davinci src="davinci.jpg" width=209 height=264>
  </ilayer></ilayer></body></html>
```

Les DIV

- Exemple
  - TODO

- DIV Imbriqués
  - TODO

History

- Objet permettant de naviguer dans l'historique

- Exemple 1

```
history.back();
window.history.back();
parent.frames[1].history.back();
```

- Exemple 2 (dans du HTML)

```
<a href="javascript:history.go(-5)">5 pages en arri&egrave;re</a>
<a href="javascript:history.back()">page pr&eacute;c&eacute;dente</a>
<a href="javascript:history.forward()">page suivante</a>
<a href="javascript:history.go(5)">5 pages en avant</a>
```

Cookies

```
<HTML><HEAD><SCRIPT>
function register(name) {
  var today = new Date(); var expires = new Date();
  expires.setTime(today.getTime() + 1000*60*60*24*365)
  setCookie("TheCoolJavaScriptPage", name, expires)
}
</SCRIPT></HEAD>
<BODY>
<H1>Register Your Name with the Cookie-Meister</H1><P>
<SCRIPT>
var yourname = getCookie("TheCoolJavaScriptPage")
if (yourname != null)          document.write("<P>Welcome Back, ", yourname)
else                           document.write("<P>You haven't been here in the last year...")
</SCRIPT>
<P> Enter your name. When you return to this page within a year,
you will be greeted with a personalized greeting. <BR>
<FORM onSubmit="return false">
Enter your name: <INPUT TYPE="text" NAME="username" SIZE= 10><BR>
<INPUT TYPE="button" value="Register" onClick="register(this.form.username.value);
    history.go(0)">
</FORM>
```

Les événements dans le document

- Différents événements

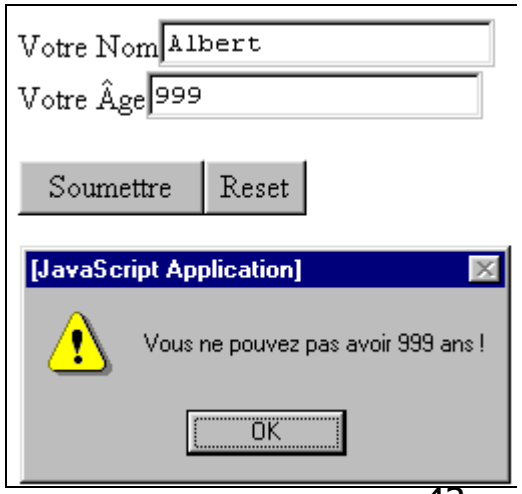
Evénement	Action de l'Utilisateur	Balise HTML
onBlur	*désélection d'un champ de saisie*	*INPUT*
onFocus	*activation d'un champ de saisie*	*INPUT*
onSelect	*sélection d'un champ de saisie*	*INPUT*
onChange	*changement de valeur d'un champ* *changement de valeur d'une sélection*	*INPUT* *SELECT*
onMouseOver	*passage de la souris sur un lien*	*A*
onClick	*clique sur un lien* *clique sur un élément du formulaire*	*A* *FORM*
onSubmit	*soumission du formulaire*	*FORM*
onLoad	*chargement de la page*	*BODY*
onUpload	*sortie du document*	*BODY*

- Handler

```
< TAG PARAMETRES ÉVÉNEMENT=FONCTION >
<INPUT   TYPE="BUTTON" NAME="monButton"
  VALUE="Appuyer"
         onClick="document.alert('Bien cliqué !')">
```

Exemple de handler

```
<HTML><HEAD><TITLE>Gestion d'événement</TITLE>
<SCRIPT LANGAGE="JavaScript">
function VerifAge(form) {
  if ((form.sonAge.value < 0) || (form.sonAge.value > 120)) {
    alert("Vous ne pouvez pas avoir " + form.sonAge.value+" ans !");
    form.sonAge.value = 0;
  }}
</SCRIPT></HEAD><BODY>
<FORM NAME="formulaire">
Votre Nom<INPUT TYPE=TEXT NAME="sonNom"><BR>
Votre Âge<INPUT TYPE=TEXT NAME="sonAge"
      onChange="VerifAge(formulaire)"> <P>
<INPUT TYPE=SUBMIT VALUE="Soumettre">
<INPUT TYPE=RESET>
</FORM></BODY></HTML>
```



Les événements dans les Fenêtres

- Propriétés de l 'objet Event
  - target, type, data (url des objets droppé)
  - height, width
  - layerX, layerY
  - modifiers: ALT_MASK, CONTROL_MASK, SHIFT_MASK, and META_MASK.
  - pageX, pageY, screenX, screenY
  - which (touche ASCII ou numéro du bouton de la sourie
- Capture des événements dans la fenêtre

```
window.captureEvents(Event.CLICK);
window.onclick= displayCoords;
function displayCoords(e) {
      if(confirm("x: " + e.pageX + " y: " + e.pageY){
                  window.releaseEvents(Event.CLICK); }
```

Exemple de capture

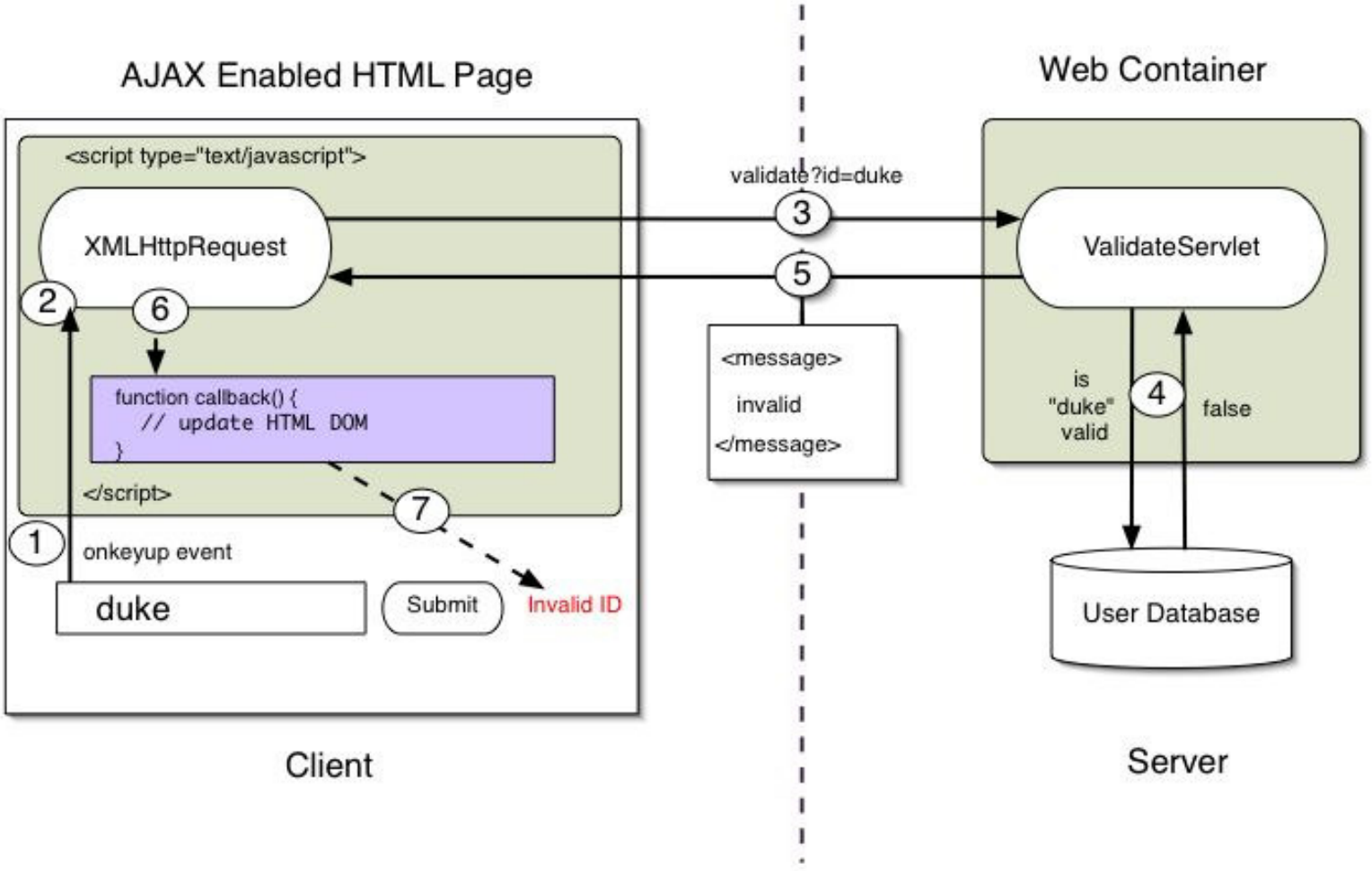
```
<html><head>
<script language="JavaScript">

window.captureEvents(Event.MOUSEDOWN | Event.MOUSEUP);
window.onmousedown= startDrag;
window.onmouseup= endDrag;
window.onmousemove= moveIt;

function startDrag(e)   { window.captureEvents(Event.MOUSEMOVE);}
function moveIt(e)      { status= "x: " + e.pageX + " y: " + e.pageY; }
function endDrag(e)     { window.releaseEvents(Event.MOUSEMOVE); }

</script></head><body>

Push the mouse button and move the mouse. The coordinates are being
displayed on the statusbar.
</body></html>
```

Asynchronous JavaScript and XML (AJAX)

- Motivations
  - Réalisation d'interfaces HTML dynamiques (Web 2.0) et interactives incluant des éléments/des chargements récupérés dynamiquement sur un/des serveurs
- API
  - Principalement objet XmlHttpRequest
    - Soumission de requêtes HTTP
      - Méthodes GET ou POST
      - Synchrone -> retourne le document entièrement chargé dans un chaîne
      - Asynchrone -> écoute de la progression du chargement
  - En conjonction avec JavaScript/DOM ou JSON

AJAX

Exemple



D'après http://java.sun.com/developer/technicalArticles/J2EE/AJAX/

AJAX
Exemple

- Objet XMLHttpRequest

```
function evalJSON()
{
   if (req.readyState == 4) {
        var doc = eval('(' + req.responseText + ')');
   }
}

var req = new XMLHttpRequest();
req.open("GET", "config.json", true);
req.onreadystatechange = evalJSON;   // la fonction de prise en charge
req.send(null); Le code JavaScript:
```

Utilisation des données

```
var nomMenu = document.getElementById('jsmenu');   // trouver un champ
nomMenu.value = doc.menu.value;          // assigner une valeur au champ

doc.commands[0].title
doc.commands[0].action
```

JSON JavaScript Object Notation

http://www.json.org/

- Format simple de fichier d'échange de données entre script AJAX et serveur
  - Avantage: évite la manipulation d'un DOM depuis JavaScript
- Syntaxe JSON
  - Un objet {} : contient des objets ou des variables.
  - Une variable scalaire: Number, String, Boolean.
  - Un tableau []
  - Les valeurs litérales: *null*, *true*, *false*, "*chaîne de caractères*", et les valeurs numériques.

JSON Exemple

```
{
  "menu": "Fichier",
  "commandes": [
      {
          "title": "Nouveau",
          "action":"CreateDoc"
      },
      {
          "title": "Ouvrir",
          "action": "OpenDoc"
      },
      {
          "title": "Fermer",
          "action": "CloseDoc"
      }
   ]
}
```

JavaScript et les Plug-ins

- Pilotage d ’un document inclus par un plug-in
  - QuickTime, RealPlayer, LiveAudio, Cosmo
- Plug-ins installés sur le navigateur
  - **navigator.plugins[] ou navigator.components[]**

Exemple avec un Plug-in

```
function rembobiner() { // d 'après Mac Farlane Chap 4
   // ... rembobine un film quicktime à la première image
}
var connu = true;
if ( ! navigator.plugins["QuickTime Plug-In"].enabledPlugin )
   connu = false;                          // le plug-in souhaité est absent
else if ( ! navigator.plugins["QuickTime Plug-In"]["video/quicktime"])
   connu = false;                          // il n'est pas configuré pour notre type MIME
if ( connu ) {
 document.write('<EMBED SRC="film.mov" HEIGHT=260 WIDTH=320'
          + 'CONTROLLER=true LOOP=false AUTOPLAY=TRUE>');
 document.write('<INPUT TYPE="BUTTON" VALUE="Play" ONCLICK="rembobiner()">');
} else { // utilise seulement les paramètres HTML standards
 document.write('<EMBED SRC="film.mov" HEIGHT=260 WIDTH=320');}
 document.write('Better with <IMG SRC="installez_quicktime.gif" HEIGHT=260
    WIDTH=320>');
}
```

LiveConnect : *Communication JavaScript avec Java*

- Une fonction JavaScript peut contrôler une applet (i.e. appeler une méthode de l'applet)

- Exemple

```
<html> <head><title>Cercle</title></head><body>
<applet code=Cercle.class id=IdCercle width=60 height=60 >
   <param name=r value=0> <param name=v value=0> <param name=b value=0>
</applet>
<script language="JavaScript">
 function setROUGE()        { document.IdCercle.setCercleColor(255,0,0); }
 function setVERT()         { document.IdCercle.setCercleColor(0,255,0); }
 function setBLEU()         { document.IdCercle.setCercleColor(0,0,255); }
</script>
<hr>
<form>
<input type=button="cmdROUGE" value="ROUGE" onClick="setROUGE()">
<input type=button="cmdVERT" value="VERT" onClick="setVERT()">
<input type=button="cmdBLEU" value="BLEU" onClick="setBLEU()">
</form></body></html>
```

LiveConnect : *Communication Java vers JavaScript (i)*

- **netscape.javascript.JSObject, netscape.javascript .JSException**
- package Java permettant à une APPLET chargée dans une page d'accèder aux objets JavaScript définis dans la page.

```
<SCRIPT>
function whatIuse() {
  alert("Vous surfez avec " + navigator.appName
          + " " +  navigator.appVersion)
}
</SCRIPT>

<APPLET CODE="myapplet.class" MAYSCRIPT>
```

LiveConnect : *Communication Java vers JavaScript (ii)*

```
import netscape.javascript.* (JSObject et JSException)
public class myApplet extends Applet {
  public void init() {
    JSObject win        = JSObject.getWindow(this);
    JSObject doc        = win.eval("document");
    JSObject myForm     = win.eval("document.testForm");
    JSObject doc2       = (JSObject) win.getMember("document");
    JSObject myForm2            = (JSObject) doc2.getMember("testForm");
    JSObject check2             = (JSObject) myForm2.getMember("testChk");
    Boolean isChecked           = (Boolean) check2.getMember("checked");
  }
 public boolean mouseUp(Event e, int x, int y) {
    win.eval("alert(\"Hello world!\");");      // par .eval
    String args[] = {"Hello"," World Bis!"};
    win.call("alert", args);                   // méthode de window
    String args2[] = {""};                     // pas d'arguments
    win.call("whatIuse", args2);               // fonction utilisateur
    return true;
} }
```

Netscape et Proxy

- Fichier proxy.pac
  - configure dynamiquement l 'accès aux proxys pour un navigateur

```
// fichier proxy.pac (d 'après Mac Farlane Chap 2)
function FindProxyForURL(url, host) {
 if (isPlainHostName(host) || isInNet(host, "192.123.234.0") )
  return "DIRECT";
 else if ( url.substring(0, 6) == "https:" || url.substring(0, 6) == "snews:")
  return "SOCKS sockshost:1081";
 else {
  if ( Math.random() < 0.5 )
     return "PROXY proxy1:1080 ; proxy2:1080";
  else
     return "PROXY proxy2:1080 ; proxy1:1080";
 }
}
```

JavaScript Hors Browser (i)

- JavaScript peut servir de langage de script en dehors des Navigateurs Webs
  - Builder (Ant), Rule engines (Drools), Component (SCA), …
- Monde Java
  - Rhino (http://www.mozilla.org/rhino/)
    - Un interpréteur
      - org.mozilla.javascript.tools.shell.Main
    - Un compilateur JS vers Java
      - org.mozilla.javascript.tools.jsc.Main
    - Utilisable depuis la tâche <script> de ANT
  - Resin JavaScript Engine
    - Peut compiler le JavaScript en bytecode Java !
- Monde .NET
  - Compilateur JScript vers CLI

JavaScript Hors Browser (ii)
Interpréteur Rhino

```
G:\devtools\rhino1_5R5pre>java -jar js.jar
Rhino 1.5 release 5 0000 00 00
js> new java.util.Date()
Sat Mar 06 23:08:22 CET 2004
js> java.lang.Math.PI
3.141592653589793
js> f = new java.io.File("test.txt")
test.txt
js> f.exists()
false
js> f.getName()
test.txt
js> f.listFiles
function listFiles() {/*
java.io.File[] listFiles(java.io.FileFilter)
java.io.File[] listFiles(java.io.FilenameFilter)
java.io.File[] listFiles()
*/}
js> for (i in f) { print(i) }
exists
parentFile
mkdir
…
```

JavaScript Hors Browser (iii)
Interpréteur Rhino

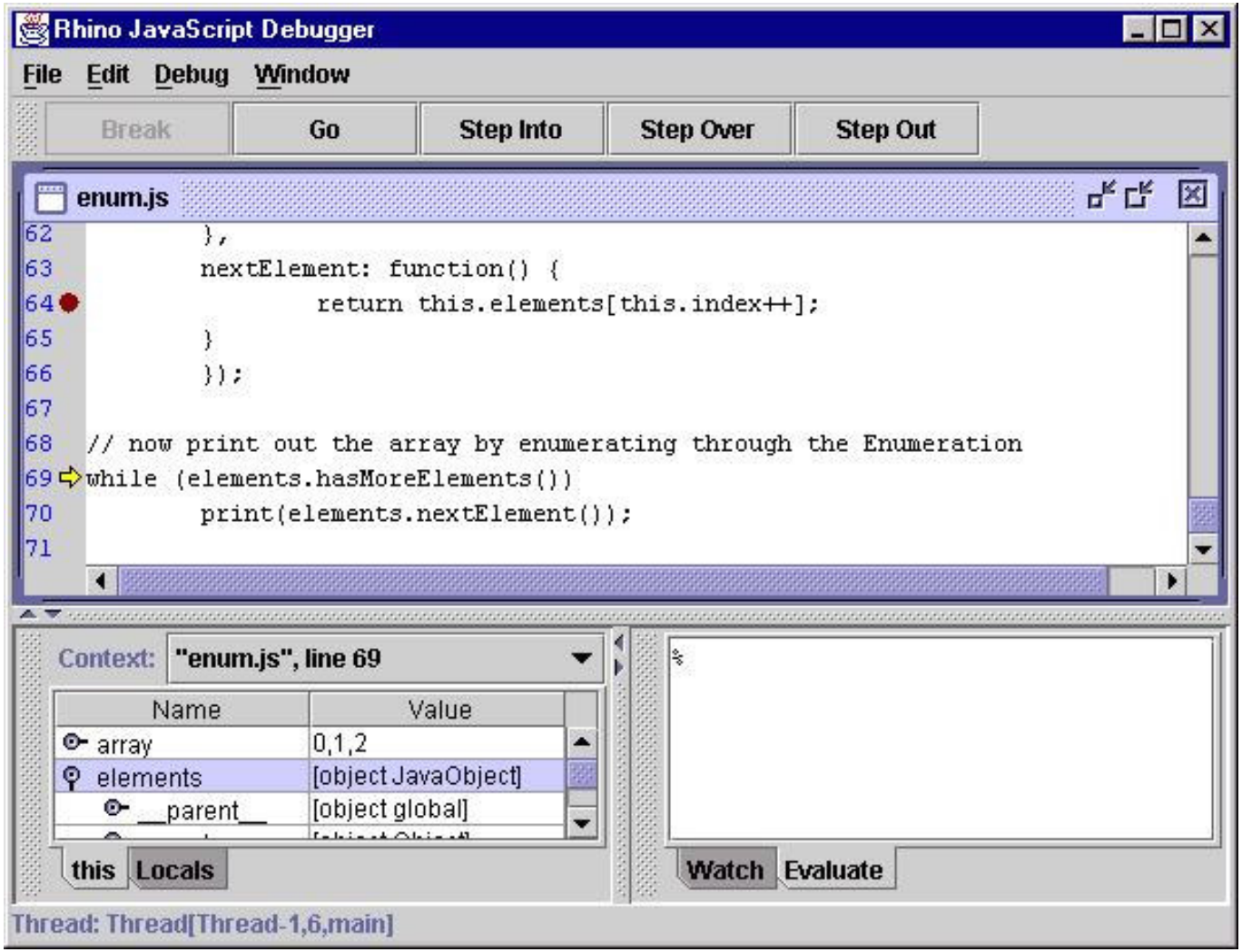
```
js> obj = { run: function () { print("\nrunning"); } }
[object Object]
js> obj.run()
running
js> r = new java.lang.Runnable(obj);
adapter18@1dd46f7
js> t = new java.lang.Thread(r)
Thread[Thread-0,5,main]
js> t.start()
js>
running
js> java.lang.System.exit(0)
G:\devtools\rhino1_5R5pre>
```

JavaScript Hors Browser (iv)
Interpréteur Rhino

```
G:\devtools\rhino1_5R5pre\examples>java -jar ..\js.jar
Rhino 1.5 release 5 0000 00 00
js> importPackage(java.awt);
js> frame = new Frame("JavaScript")
java.awt.Frame[frame0,0,0,0x0,invalid,hidden,layout=java.awt.BorderLayout,title=
JavaScript,resizable,normal]
js> frame = new Frame("JavaScript")
java.awt.Frame[frame1,0,0,0x0,invalid,hidden,layout=java.awt.BorderLayout,title=
JavaScript,resizable,normal]
js> frame = new Frame("JavaScript")
java.awt.Frame[frame2,0,0,0x0,invalid,hidden,layout=java.awt.BorderLayout,title=
JavaScript,resizable,normal]
js> frame.show()
js> frame.setSize(new Dimension(200,100))
js> button = new Button("OK")
java.awt.Button[button0,0,0,0x0,invalid,label=OK]
js> frame.add(button)
java.awt.Button[button0,0,0,0x0,invalid,label=OK]
js> frame.show()
js> quit()
```

JavaScript Hors Browser (iv)
Debuggeur Rhino

java org.mozilla.javascript.tools.debugger.Main [options] [filename.js] [script-arguments]



JavaScript Hors Browser (v)

Exemple avec la tâche <script> ANT (java)

```
<project name="squares" default="main" basedir=".">
  <target name="setup">
    <script language="javascript"> <![CDATA[
      for (i=1; i<=10; i++) {
        echo = squares.createTask("echo");
        main.addTask(echo);
        echo.setMessage(i*i);
      }
    ]]> </script>
  </target>
  <target name="main" depends="setup"/>
</project>
```

JSR 223
Scripting for the Java Platform

- API pour exécuter des scripts depuis Java
  - Indépendance du langage (JavaScript, Groovy, Jython, PHP, …)
  - Présent dans Mustang (Java 6.0)
- ScriptEngine
  - JavaScript (Rhino), PHP, Groovy, Ruby …
- Outils JRunScript (Java 6.0)

  jrunscript -e "cat('http://java.sun.com')"

  jrunscript -l js -f test.js

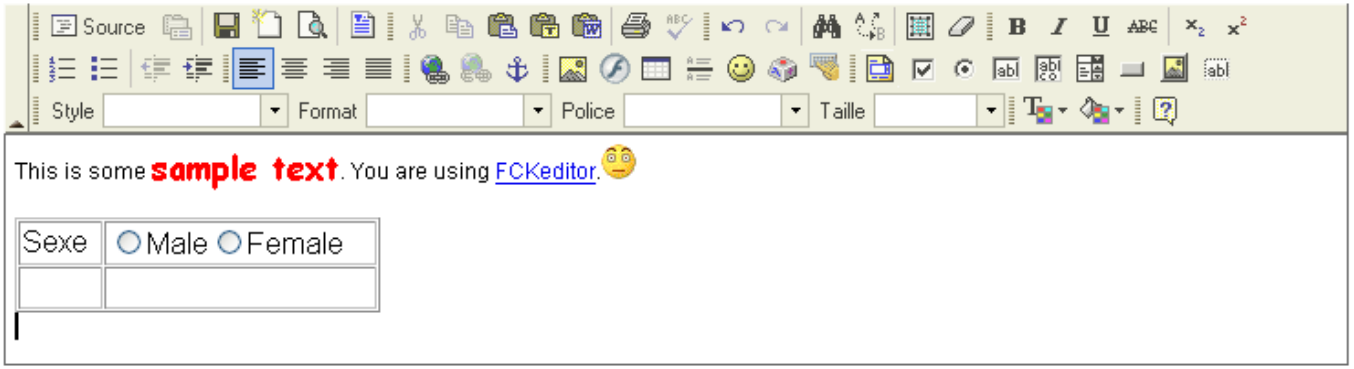
  Jrunscript

  js>print('hello world');

  hello world
- Alternative
  - BSF (bien plus ancien)

Bibliothèques

- Multitude de scripts et des « composants »
  - Intégrables à vos applications Web
- Exemples
  - Prototype http://www.prototypejs.org/api
  - DOJO http://dojotoolkit.org/
  - FCKEditor http://www.fckeditor.net/



  - …

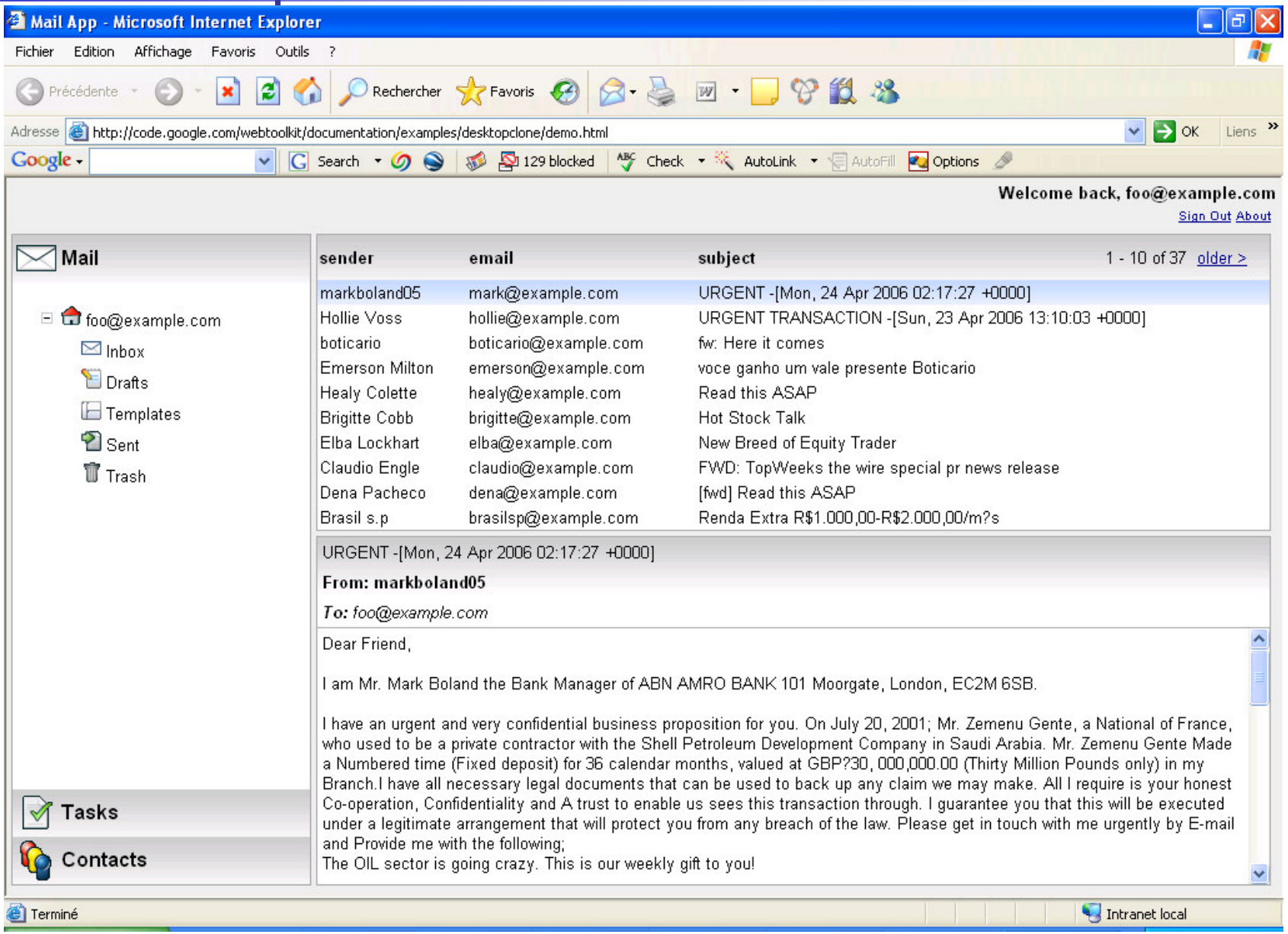
Google Web Toolkit
http://code.google.com/webtoolkit/

- Compilateur Java vers JavaScript/AJAX pour la réalisation de clients riches Web 2.0

- APIs
  - Widgets de base
  - Communications push/pull
    entre les clients AJAX et la servlet (coté serveur)

- Alternative : Echo2

Google Web Toolkit Exemple



Google Toolkit

- **JavaScript Native Interface (JSNI)**
- GWT compiler
  - translates Java source into JavaScript. Sometimes it's very useful to mix handwritten JavaScript into your Java source code. For example, the lowest-level functionality of certain core GWT classes are handwritten in JavaScript. GWT borrows from the Java Native Interface (JNI) concept to implement Java*Script* Native Interface (JSNI).
- Writing JSNI methods is a powerful technique, but should be used sparingly. JSNI code is less portable across browsers, more likely to leak memory, less amenable to Java tools, and hard for the compiler to optimize.
- We think of JSNI as the web equivalent of inline assembly code. You can:
  - Implement a Java method directly in JavaScript
  - Wrap type-safe Java method signatures around existing JavaScript
  - Call from JavaScript into Java code and vice-versa
  - Throw exceptions across Java/JavaScript boundaries
  - Read and write Java fields from JavaScript
  - Use hosted mode to debug both Java source (with a Java debugger) and JavaScript (with a script debugger, only in Windows right now)
- Utilisation avec les RPC
  - JavaScript (browser side) – Java (Server side)
    - com.google.gwt.user.server.rpc.RemoteServiceServlet

OpenLaszlo *http://www.openlaszlo.org/*

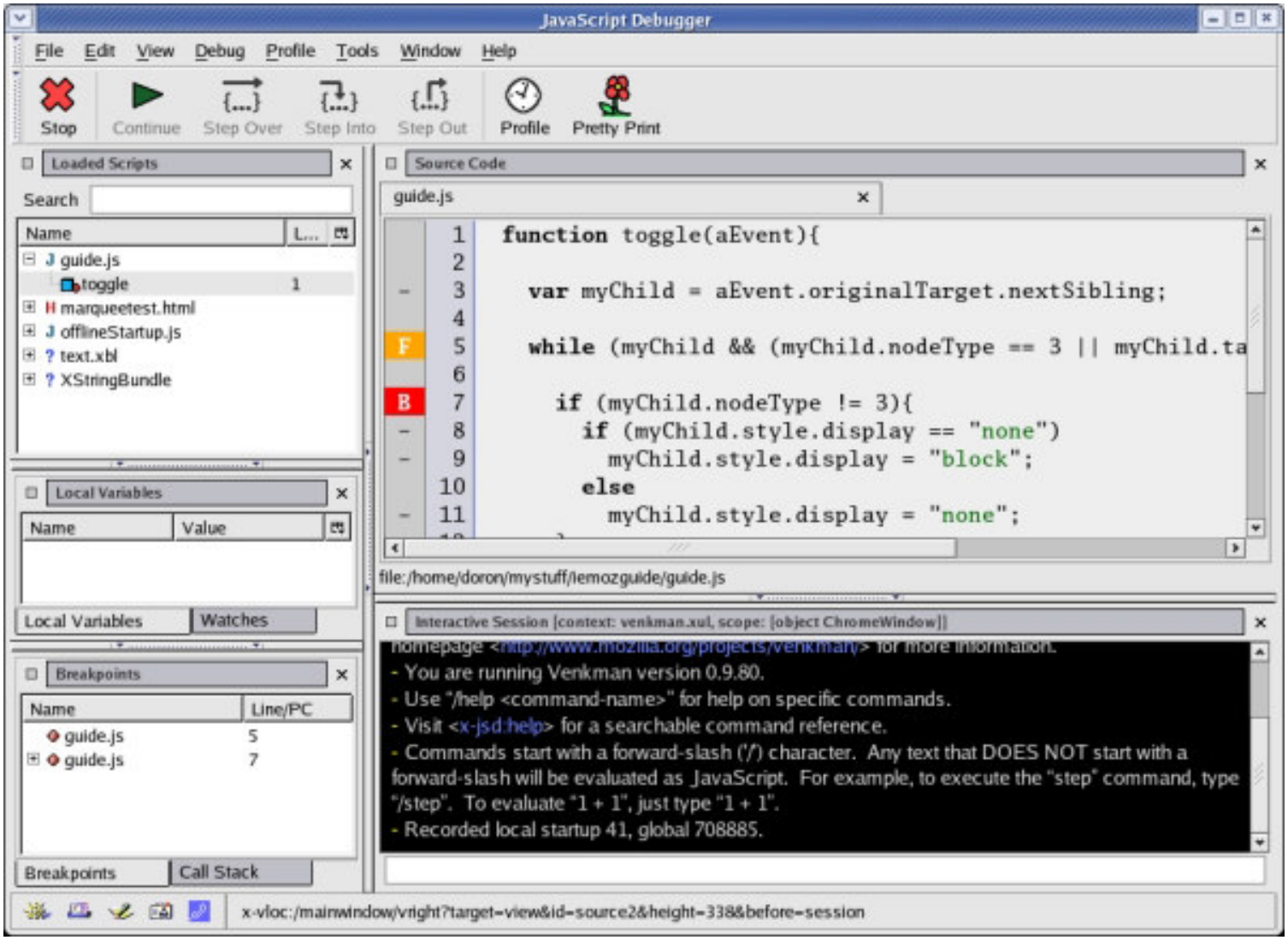
- Compile LZX-written programs to Flash and DHTML/AJAX
  - The LZX language is a mix of XML and **JavaScript**
- APIs for
  - animation, layout, data binding, server communication, and declarative UI, audio and video streaming.
- License: CPL (similar to EPL)
- Misc
  - LZX reference manual : http://www.openlaszlo.org/lps/docs/reference/index.html
  - Laszlo Explorer : http://www.openlaszlo.org/lps/laszlo-explorer/index.jsp
  - OpenLaszlo server: Container servlet enabling JIT LZX compilation
  - Book
    - Norman Klein and Max Carlson with Glenn MacEwen , *Laszlo in Action,* Pub. Manning, November 2007, ISBN 1-932394-83-4

Des outils

- MicroSoft VisualStudio
- MacroMedia DreamWeaver
- Eclipse WST

- Venkman
- FireBug
- JSLint
- JSDoc
- JavaScript Shell

Venkman Debugger



From http://developer.mozilla.org/en/docs/Migrate_apps_from_Internet_Explorer_to_Mozilla

Webographie

- JS
  - http://www.javascript.com/
  - http://developer.mozilla.org/en/docs/JavaScript
  - http://www.mozilla.org/js/language/js20/

  - http://www.quirksmode.org/
  - http://fr.selfhtml.org/javascript/
- Rhino
  - http://www.mozilla.org/rhino/

Bibliographie

- Les livres *(Attention ! ils deviennent rapidement obsolètes)*
- *JavaScript*
  - Oreilly, http://www.ora.com/
  - Nigel McFarlane, JavaScript, Ed Eyrolles, 1998, ISBN 2-212-09034-X
  - Arman Danesh, JavaScript, (Simon & Schuster Macmillan) en français ISBN 2-7440-0167-8
- Ajax
  - David Crane and Eric Pascarello, Ajax in Action, Ed Manning, 2005, ISBN 1932394613
    - http://www.manning.com/crane