Technologies Web

JavaScript

Marco Winckler

ICS-IRIT

Université Paul Sabatier (Toulouse 3)

http://www.irit.fr/~Marco.Winckler

winckler@irit.fr

Références : sites Web

- Spécifications html:
  - http://www.w3.org/TR/html401/interact/scripts.html
- Spécifications ECMA-262:
  - http://www.ecmainternational.org/publications/standards/Ecma-262.htm
- Spécifications JavaScript1.5:
  - http://devedge.netscape.com/central/javascript/
- DOM Javascript:
  - http://devedge.netscape.com/library/xref/2002/clientdata/index.html
- DOM w3c:
  - http://www.w3.org/DOM/DOMTR
- Fonctions et objets:
  - http://fr.selfhtml.org/javascript/objets/index.htm

Partie 1 : Généralités

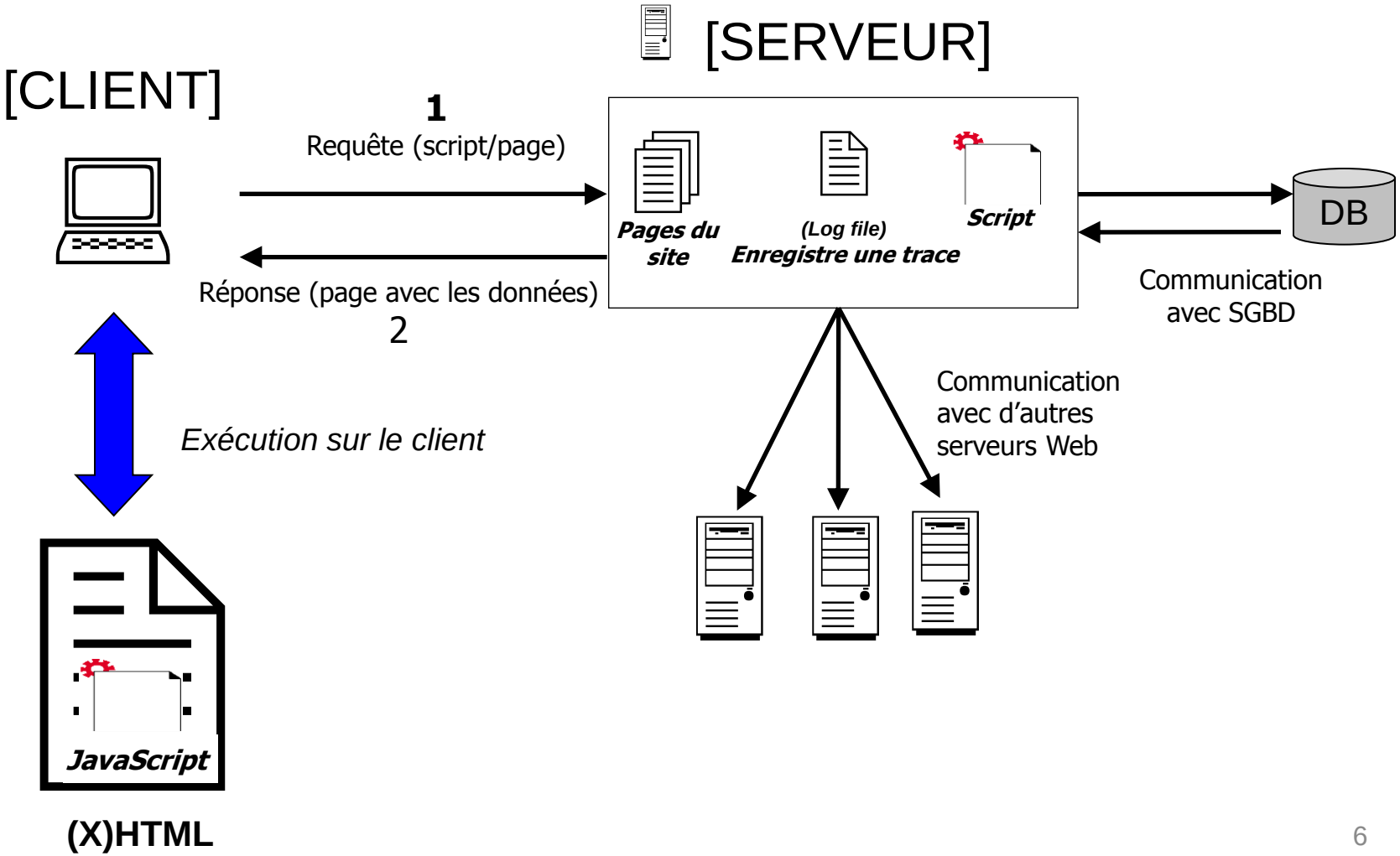
Origines

- LiveScript, Netscape
- 1995, JavaScript (joint venture Netscape & Sun Microsystems)
- Fin des années 90 : ECMA-262 (*European Computer Manufactors Association)* $\rightarrow$ ***ISO-16262***
  - *ECMA-262* $\rightarrow$ EMACScript, supporté par Netscape et Microsoft
  - Netscape : JavaScript
  - Microsoft : JScript

Le langage

- Trois parties:
  - **Le noyau** : structure du langage (opérateurs, expressions, commandes, sub-programmes)
  - **Le côté client** : contrôle du browser + interaction avec l'utilisateur
  - Le côté serveur : objets qui permet l'exécutions des programmes sur le serveurs (très peu utilisé)

L'architecture Web



Pour quoi des scripts client ?

- Limiter les échanges entre client et serveur
- Permettre des vérifications localement (date, nombre, etc)
- Ajouter de l'interactivité aux pages Web (bouton, textes, etc)
- Java ou JavaScript ?
  - Applet Java
    - Langage de programmation,
    - Indépendant de HTML
    - Très Puissant
  - JavaScript
    - Ancré dans HTML
    - Langage interprété,
    - Variables non typées

Utilisations de JavaScript

- Pour quoi faire ?
  - Traitement local (client) d'évènements déclenchés par l'utilisateur
- Applications simples
  - Ex. : calculettes, convertisseurs, devis automatique
- Aspects graphiques
  - Ex. : modifications d'images liées à la position de la souris)
- Tests de validité
  - Ex. : saisie de chiffres exclusivement)
- Chargement de plug-in

Les ajouts requis dans HTML

- Balise <script>:
  - Permet l'insertion de code exécuté par le browser
  - Attribut type="text/javascript"
  - Attribut src="filename.js" : charger du code
  - stocké dans un autre fichier
  - Attribut defer: peut continuer la lecture de la page.
- Balise <noscript>:
  - Contenu affiché si le langage de scripts n'est pas reconnus.

Script joint ou externe

- Script joint au code (X)HTML

```
<script type="text/javascript">
        <!--
        -- JavaScript script --
        // -->
</script>
```

- Fichier externe, ex. :

```
<script type="text/javascript" src="filename.js" />
```

Exécution des scripts

- Trois méthodes d'exécution:
  - Exécution à la lecture de la page: balise <script>
  - En lien de balise (href): avec le pseudo-protocole javascript
  - En réaction à un évènement: action de l'utilisateur

Les ajouts requis dans HTML

- Attributs des évènements :
  - contient le code à exécuter en réponse à l'évènement:
  - onload: pour <body> et <frameset>
  - onclick
  - onmouseover
  - …

Partie 2 : Programmation

Liste de mots réservés

break, case, catch, continue, default,delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, void, while, with

- Commentaires

// pour une ligne

```
/*
    Plusieurs lignes
*/
```

Expression de contrôles

== égal
! = différent
< inférieur
> supérieur
<=inférieur ou égal
>=supérieur ou égal
=== exactement égal // pour la conversion de types
! == exactement différent

- Opérateur booléen ET : &&
- Opérateur booléen OU : ||

Instructions Conditionnelles

```
if (a > b)
   a = b;
else {
   b = b + 1;
   a = b + 2;
}
```

```
swich(variable ou expression) {
   case valeur_1:
         // instruction ;
         break;
   …
   [default :
         // instruction;]
}
```

Les Types Primitives

- **Nombres** :
  - Pas de distinction entre réel et entier
  - Ex. : var taille =1.75; var taille = new Number(1.75);
  - Formats possibles : 72 7.2 .72 72. 7E2 7e2 .7e2 7.e2 7.2E-2
- **Booléens** :
  - Deux valeurs possibles uniquement : true ou false
- **Chaînes de caractères** :
  - String, écrite entre ‘ ’ ou " "
  - Ex.1: ‘cas d\’école’ Ex.2: ‘D:\\filename’
- **Nulle**
  - Mot réservé : **null**
  - false ou zero
- **Non défini**
  - Mot réservé : **undefined**
  - (false or NaN)

Expressions Itératives

```
while (expression) {
   instructions;
}

for (initial; condition; increment ) {
    instructions;
}
```

Variables

- Non typées
- Pas d'initialisation par défaut
- Mot clé: var
- Noms de variables
  - Commencer par : lettres, souligné (_), dollar ($)
  - Pas de limite de taille
  - Attention à la casse : a ≠ A

Déclaration des variables

```
var conteur,
    index,
    pi = 3.14159265,
    texte = "chaîne de caractères",
    stop_flag = true;
```

- index vaut **undefined**
- Toujours déclarer les variables!

Déclaration des Tableaux

- Exemples :

  var tab=new Array();

  - crée un tableau vide.

  var tab=new Array(5);

  - crée un tableau de 5 éléments vides.

  var tab=new Array(1,2,'a');

  - crée un tableau de 3 éléments. Equiv. à var tab=[1,2,'a'];.

Méthode utilisés avec les tableaux

join(sep):

- joint les éléments dans une chaîne en utilisant le séparateur sep

var names = new Array[“Mary”, “John”, “Max”];

var names_string = names.join(“ : ”); // “Mary : John : Max”

reverse(), sort()

- pour inverser l’ordre et trier.

names.sort(); // [“John”, “Mary”, “Max”];

concat(...)

- concatène des éléments ou des tableaux;

slice(debut,fin):

- extraction de sous-tableaux (les indices peuvent être négatifs)

push(val); pop(), unshift(val), shift()

- manipulations type piles.

Les fonctions

- Exemple *:*

```
…
fun();
function fun() {
    var doc = "c pas drôle! <br/>";
    return doc;
}
```

- ***return*** est le mot clé utilisé pour faire retourner une valeur à la fonction

Création et modification d'objet

```
// création d'un objet
var my_car = new Object();
// création et initialisation de la propriété « make »
my_car.make = "Ford";
// création et initialisation d'objet ajouté
my_car.engine = new Object();
my_car.engine.config = "v6";
my_car.engine.hp = "200";
```

Access au valeur d’un objet

```
// deux façon de faire
var prop1 = my_car.make;
var prop2 = my_car["make"];

// effacer une proprieté
delete my_car.make;
```

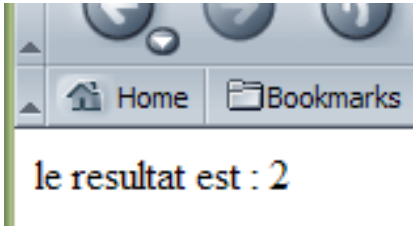
Expressions régulières

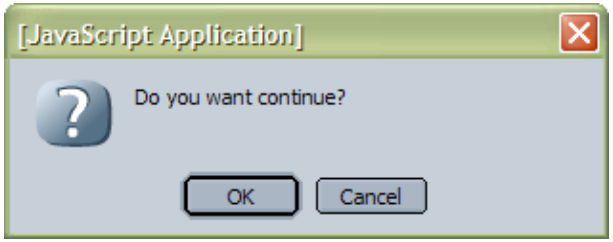
- re=/exp/ ou re=new RegExp("exp")
- Méthodes : RegExp.exec, RegExp.test, String.match, String.search, String.replace, String.split
- Ex. :

```
<script type="text/javascript">
   re = /(\w+)\s(\w+)/;
   str = "John Smith";
   newstr = str.replace(re, "$2, $1");
   document.write(newstr);
</script>
```

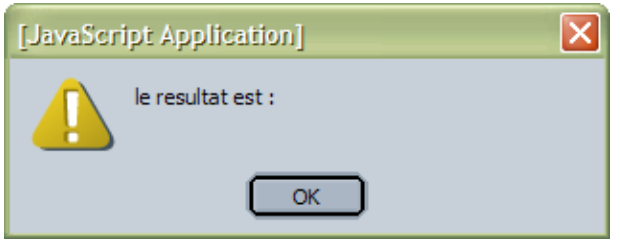
Lecture et Affichage de donné

```
var result = 1;
document.write("le resultat est : ", result+ 1);
```



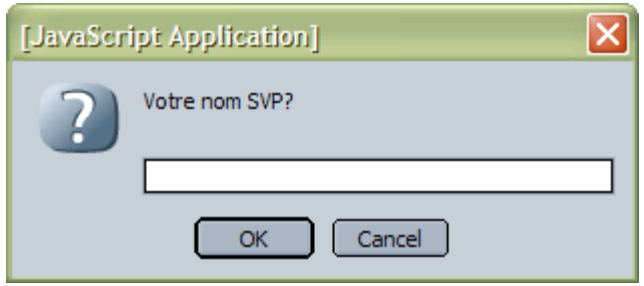



```
var question = confirm("Do you want continue?");
// ok = true
// cancel = false
```




```
var result = 1;
alert("le resultat est : ", result+ 1);
```

```
var name = prompt("Votre nom SVP?");
```



Partie 3 : Modèle Objet

Objets Définis

- Utilitaires du langage: Math, String, Array, etc.
- Classes du navigateur : instanciées au démarrage (ECMA)
- Window : zones du navigateur
- Navigator : informations propres au navigateur utilisé
- Screen : capacités d'affichage de la machine du client
- document DOM: modèle de document HTML (w3c)

Exemples d'objets définies

- Objet *MATH*

  var x = 10;

  var y = Math.sin(x);

- Objet *String*

```
var str = "George";
var len = str.lenght;          // 6
str.chartAt(2);                // 'o'
str.indexOf(r);                // 3
str.substring(2,4);            // 'org'
str.toLowerCase();             // 'george'
```

Objet "utilitaire"

- Objet global
  - Conversion de types
    - Ex.

```
var str = String(value);
var num = 6;
var str_value = num.toString();          // '6'
var str_value_binay = num.toString(2);   // '110'
var number = Number(str_value) – 1;      // 5
```

  - Gestion des URI

Objet "utilitaire"

- Classes du langage
  - Date:
    - var maDate = new Date (annee, mois, jour, h, min, s);
    - getDate() : retourne le jour du mois (entre 1 et 31)
    - getDay(), getHours(), getMinutes(), getMonth(),
    - getSeconds(),getFullYear()
  - Image:
    - var monImg = new Image();

Classe Window

- Instance créée par zone d'affichage pouvant contenir du code HTML
- Attributs
  - history : liste des URL visitées. Méthode : back(), forward(), go(entier)
  - location : caractéristiques de l'Url contenue dans la zone. Attributs : href, hostname, pathname, port, protocol

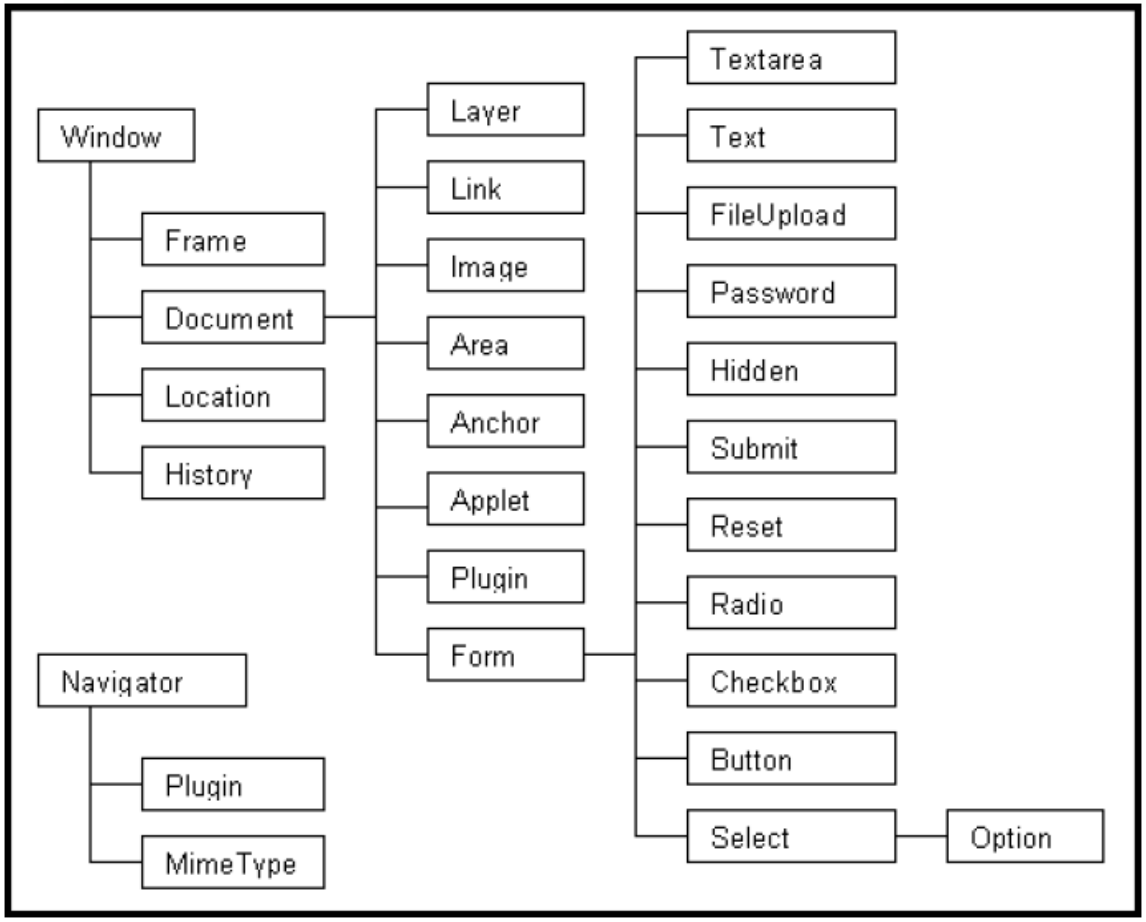
Classe Window

- documents : caractéristiques du document HTML et de ses composants
- Attributs : forms, images, links
- Méthodes : write(), writeln(), open(), close()
- name : nom de la zone
- parent, top, opener : objet Window contenant cette
- zone, ou ayant ouvert la fenêtre.
- status : texte a écrire dans la zone status de la fenêtre

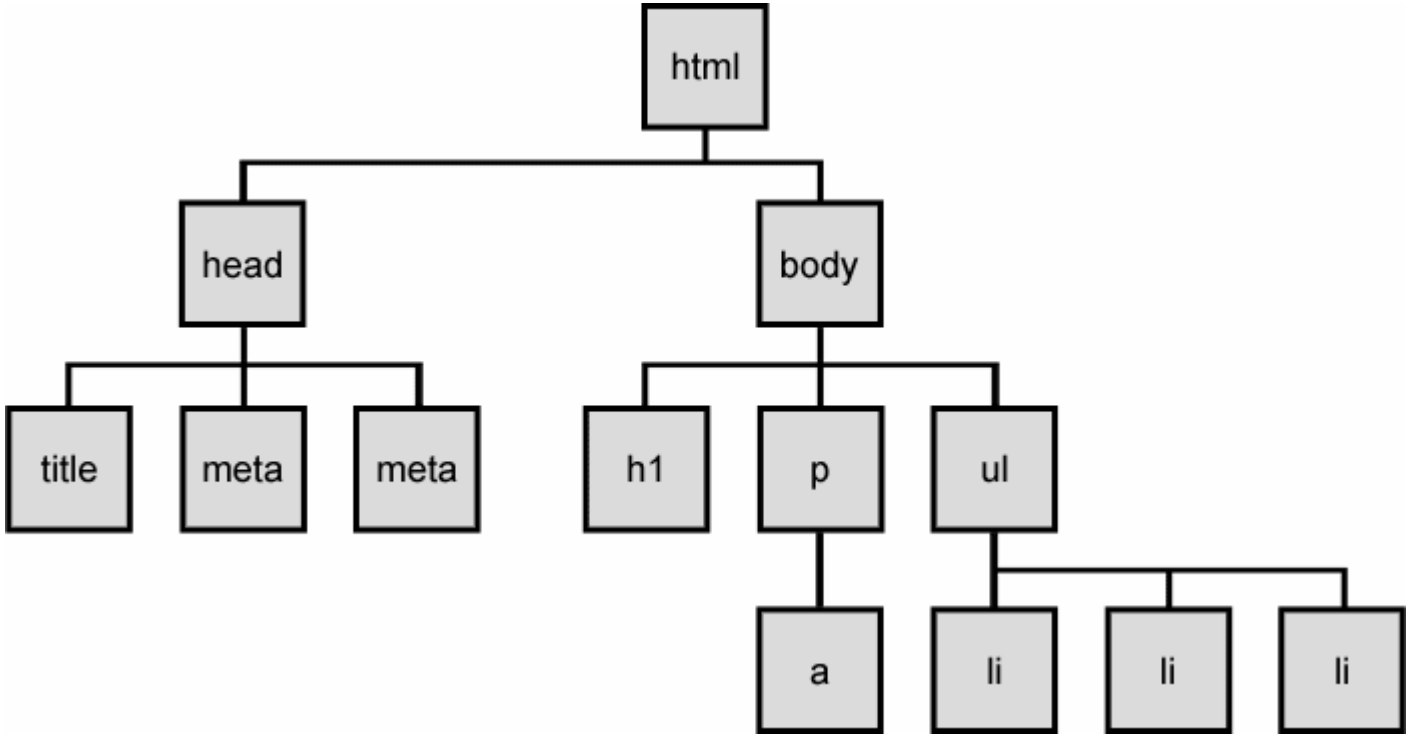
Classe Window

- Instance créée par zone d'affichage pouvant contenir du code HTML
- Méthodes :
  - open(url,nom,options): ouvre une fenêtre
  - options: toolbar=yes/no, location=yes/no,
  - Status=yes/no, menubar=yes/no, titleBar=yes/no,
  - alwaysRaised=yes/no, resizeable=yes/no,
  - screenX=N, screenY=N, outerWidth=N,
  - outerHeigth=N
  - close(), focus(), blur()
  - alert(msg), prompt(msg,def.),
  - confirm(msg): fenêtres de dialogue

Gestion du document



Exemple



Access des éléments DOM (1/3)

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head <title>form </title> </head>
  <body>
     <form action=" " name="myForm" >
          <input type="button" name="turnItOn" />
      </form>
    </body>
</html>
```

document.forms[0].elements[0];

document.myForm.turnItOn;

document.getElementById("turnItOn");

Access des éléments DOM (2/3)

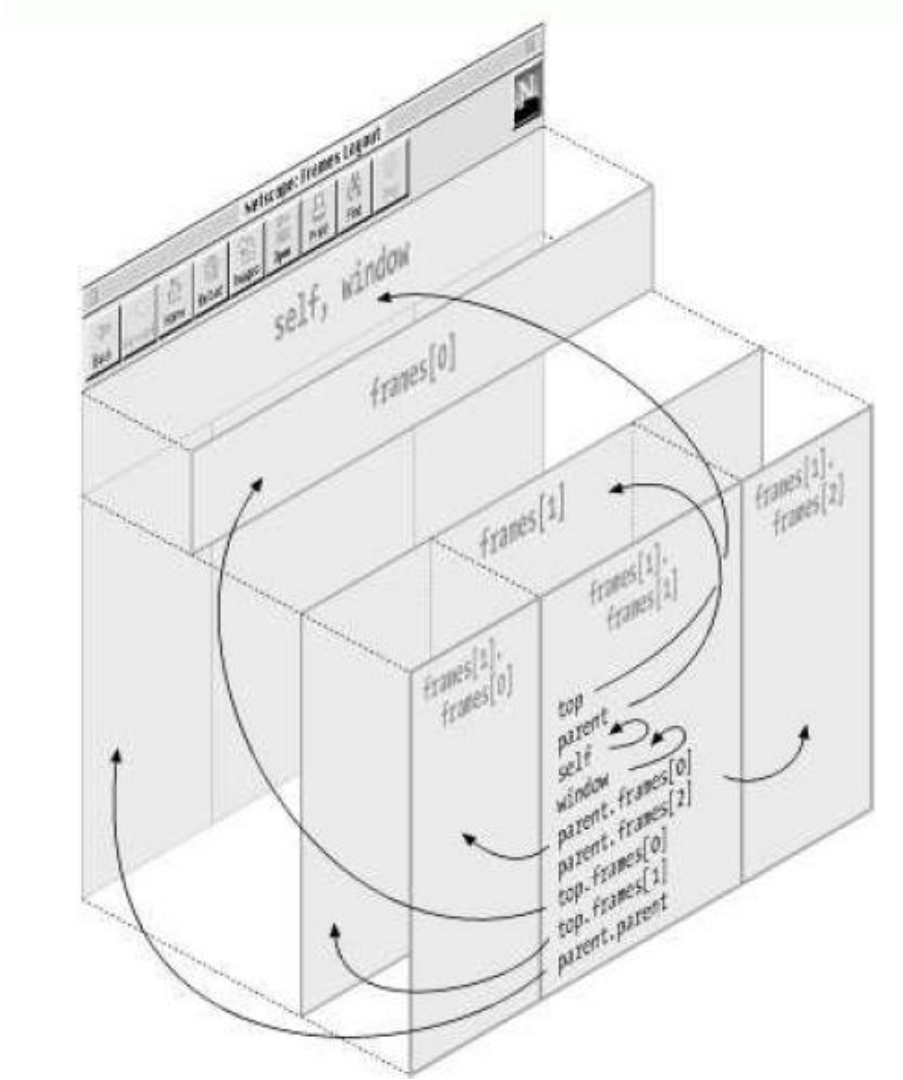
```
<form action=" " id="vehicleGroup" >
    <input type="checkbox" name="vehicles" value="car" /> car
    <input type="checkbox" name="vehicles" value="truck" /> truck
    <input type="checkbox" name="vehicles" value="bike" /> bike
</form>
```

```
var numChecked=0;
var dom = document.getElementById(="vehicleGroup");
for (index =0; index < dom.vehicles.lenght; index++) {
        if (dom.vehicles[index].checked)
                numChecked++;
}
```

Access des éléments DOM (3/3)

- *getElementById()*
- *getElementsByName()*
- *getElementsByTagName()*
- **getElementsByClassName() -> HTML 5**

Gestion du document



Gestion des événements

- Html: <a onclick="action(param)">.
- Objet Javascript: monBouton.onclick=action;
- DOM: element.addEventListener('click', action, false)

Gestion du document: DOM

- Représentation homogène du document sous forme d'arbre
  - Noeuds (nodes) pour chaque élément html
  - Noeuds Texte pour le contenue des éléments
- Permet de le manipuler par les méthodes des noeuds ou du document:
  - Recherche dans les fils du noeud (ou tout le document):
    - getElementsByTagName(nom): recherche les éléments selon leur nom, retourne un tableau:
    - paragraphes=document.getElementsByTagName('p');
    - getElementById(id): recherche un élément selon son attribut id, retourne l'élément.

Gestion du document: DOM

- Modification du document:
  - document.createElement(nom): Crée un élément html
  - document.createTextNode(texte): Crée du texte
  - appendChild(noeud) ajoute un noeud fils à un noeud
- Ex. : Ajout d'un paragraphe après le paragraphe d'id="test":

```
nouveau_p=document.createElement('p');
nouveau_txt=document.createTextNode('bla bla bla');
nouveau_p.appendChild(nouveau_txt);
document.getElementById('test').appendChild('nouveau_p');
```

  - (voir les références pour une liste complète des méthodes)

Gestion des événements

- *Ergonomie*: Aider l'utilisateur, ne pas le géner.
- *Accessible*: Le site ne doit pas dépendre du script.
- *Facile à implémenter*: discret pour le développeurs web. Inclusion du script et ajout de classes ou id)
- Notion de feuille d'actions.

Gestion des événements

- Séparation du comportement et de la structure.
- Association des fonctions aux événements par le script lui-même.
- Fonction: objets comme les autres.

Partie 4 : Compléments

Parcours du DOM

- Récuperation d'éléments
  - Par l'id (attribut ID de la balise)
    - Document.getElementById(« id »)
  - Par le nom (attribut name de la balise)
    - Document.getElementByName(« name »)
  - Par le nom de balise (tag name)
    - Document.getElementsByTagName(«tagname»)

Parcours du DOM 2

- A partir d'un élement
  - Obtention du père
    - <Element>.**parentNode**
  - Obtention des fils
    - <Element>.childNodes()

Css et javascript

- Css
  - P {

Border : solor red color 10px

Background-color : #FF9933

Color:#fffff

Font-size:36pt

}

- Javascript

```
document.getElementsByTagName("p")[i].style.border = "solid red 10px";
 document.getElementsByTagName("p")[i].style.backgroundColor "#FF9933";
document.getElementsByTagName("p")[i].style.color = "#FFFFFF";
document.getElementsByTagName("p")[i].style.fontSize = "36pt";
```

JQuery

- Une librairies pour faciliter l’exploration du dom

- $("a") équivalent à document.getElementsByTagName(« a »);

AJAX

- Une classe JS: XMLHttpRequest
  - Créer une instance

```
try {
xhr = new ActiveXObject("Microsoft.XMLHTTP"); // IE  ?}
catch(e) // Echec
 { xhr = new XMLHttpRequest(): // Netscape,Safari}
```

AJAX

- Faire une fonction d'attente de réponse
  - Les codes status

    0: non initialisé.

    1: connexion établie.

    2: requête reçue.

    3: réponse en cours.

    4: terminé.
  - Les codes http
    - 200 OK
    - 404 page non trouvée

AJAX

- xhr.onreadystatechange = function(){ // traitement de la réponse };

```
Corps de la fonction
if(xhr.readyState == 4)
    {
    if(xhr.status == 200)  {
      document.ajax.dyn="Received:" + xhr.responseText;
      }
    else {
      document.ajax.dyn="Error code " + xhr.status; }
 }
```

AJAX

- Requete
  - Lire un page

```
xhr.open('GET', 'http://www.xul.fr/fichier.xml', true);
```

  - Envoyer une page

```
xhr.send(null);
```

Html pour exemple AJAX

```
<FORM method="POST" name="ajax" action="">
<INPUT type="BUTTON" value="Submit" ONCLICK="submitForm()">
<INPUT type="text" name="dyn" value="">
</FORM>
```