Javascript et DOM

Code: js-dom

Auteurs et version

- Daniel K. Schneider

- Version: 0.8 (modifié le 3/4/08)

Prérequis

Module technique précédent: **js-intro** (Javascript à l'ancienne)

Voir aussi:

Module technique suppl.: **php-dom** (XML DOM avec PHP 5)

Abstract

Introduction à JavaScript et DOM

Objectifs

- Survol des différents DOM
- HTML DOM de base (simples lectures et manipulations)
- A faire: XML-DOM, Ajax

Source & Remerciements

- Une grande partie de ces slides est inspirée par la Référence du DOM Gecko
- Je vous conseille également de travailler avec ce manuel en ligne ...
  url: **http://developer.mozilla.org/en/docs/Gecko_DOM_Reference**
  url: **Traduction française: Référence du DOM Gecko**
- Une autre source était les spécifications W3C et ses traductions
  url: **http://www.w3.org/TR/DOM-Level-2-Core/**
  url: **http://www.yoyodesign.org/doc/w3c/dom2-core/Overview.html (traduction)**
  url: **http://www.w3.org/TR/DOM-Level-2-HTML/**
  url: **http://www.yoyodesign.org/doc/w3c/dom2-html/Overview.html.html (traduction)**
  url: **http://www.w3.org/TR/DOM-Level-2-Events/**
  url: **http://www.yoyodesign.org/doc/w3c/dom2-events/Overview.html.html (traduction)**
  url: **http://www.w3.org/TR/DOM-Level-2-Style/**
  url: **http://www.yoyodesign.org/doc/w3c/dom2-style/Overview.html (traduction)**
- Flanagan, David, JavaScript, 5th edition (!), O'Reilly (la version 4 va plus ou moins)

1. Table des matières détaillée

2. Introduction

Les navigateurs récents (IE6, Moz1x, Firefox) implémentent un modèle plus puissant et surtout plus propre de ce qu'on avait appelé "DHTML" et qui reposaient sur des technologies non-standardisées et donc incompatibles. Voici les composantes

1. Le langage "JavaScript", un langage de scripting "objets" et qui actuellement est formalisé sous l'appellation "ECMAScript"
2. Les "Document Object Models" (DOMs), qui définissent une série de méthodes et propriétés avec lesquelles on peut interroger et modifier n'importe quel élément d'un document HTML, XML ou encore langages XML comme SVG, X3D, etc.
3. Et bien sûr: HTML + CSS

2.1 Les DOMs du W3C

Principe

1. Un DOM est une description structurée d'un document HTML ou XML.
2. Un DOM fournit une interface à cette structure et qui permet de modifier structure, contenu et style d'un document XML ou HTML

En termes techniques:

1. Un DOM est un arbre avec des noeuds
2. Chaque noed est un objet pour lequel il existe des méthodes et des propriétés (qu'on peut "lire" ou modifier).

- Ces interfaces (classes, méthodes et propriétés) sont implémentés pour plusieurs langages comme JavaScript, PhP 5, Java, Python, Perl, ActiveX, ... Dans ce contexte, on parle de "Language bindings".
- Les noms des interfaces, classes, méthodes et propriétés sont en principes les mêmes dans toutes ces implémentations (ce qui facilite la vie).

URL des spécifications:

url: **http://www.w3.org/DOM/DOMTR (liste des spécifications)**

url: **http://www.w3.org/2005/11/Translations/Lists/ListLang-fr.html (liste des traductions)**

2.2 Niveaux et familles DOM

- Les spécifications W3C DOM sont organisés en "niveaux" (Angl. levels)

Level 0

- Par "level 0" on entend des implémentations variés non standardisés, comme le modèle d'objets DHTML de Microsoft, le DOM de Netscape 4.x, etc.
- Certaines fonctions sont toujours indispensables comme par exemple la méthode **`alert()`**

Level 1 (octobre 1998)

- DOM Level 1 Core: définit la navigation et la manipulation de documents HTML et XML
- DOM Level 1 HTML: extensions spécifiques à HTML

Level 2 (2001)

- 6 spécifications, ajoute XML namespace support, filtered views and events.
- DOM Level 2 Core Specification (étend DOM Level 1 Core)
- DOM Level 2 Views Specification:
- DOM Level 2 Events Specification: gestion d'événements (mais pas du clavier)
- DOM Level 2 Style Specification: lecture et modification du CSS
- DOM Level 2 Traversal and Range Specification
- DOM Level 2 HTML Specification: extensions pour HTML

Level 3: contient 6 spécifications

- DOM Level 3 Core
- DOM Level 3 Load and Save
- DOM Level 3 XPath
- DOM Level 3 Views and Formatting
- DOM Level 3 Requirements

- DOM Level 3 Validation

Autre langages XML du W3C

- MathML
- SVG
- SMIL

Autres langages

- X3D, ...

2.3 Implémentations

- La plupart des navigateurs modernes (2005 +) offrent des implémentations assez bonnes du DOM Level 1 (mais jamais complètes) et des implémentations lacunaires de DOM Level 2 (IE 7 par exemple est plutôt mauvais !).
- La plupart des navigateurs implémente des extensions propriétaires aux DOM 1 et 2 et qu'il faut éviter si possible, par exemple le "fameux" innerHTML.
- Par contre, on est obligé à utiliser les implémentations propriétaires de l'objet "navigator", car il n'existe pas encore de standards (mais les implémentations se ressemblent).
- J'ai testé la plupart des exemples avec Firefox 1.5 ou 2.x, et IE 7. Certains avec Nokia N73

Tableaux de comparaison:

url: **Comparison of layout engines (DOM) (wikipedia)**

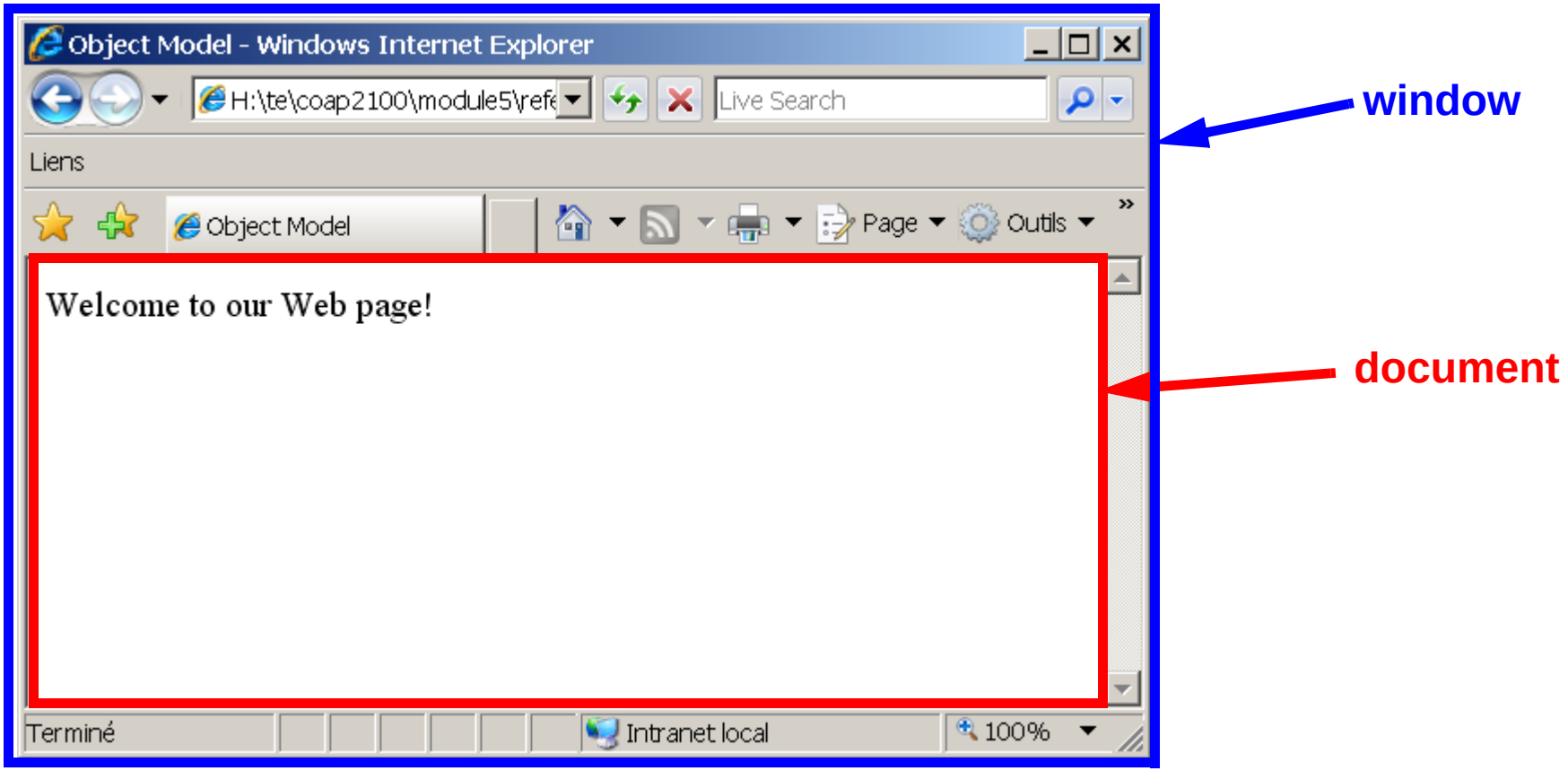
url: **http://www.quirksmode.org/dom/w3c_html.html (Quirksmode)**

url: **http://www.quirksmode.org/dom/w3c_core.html**

2.4 Survol des objets DOM

Il existe quelques variables système qui contiennent les objets les plus importants:

- **document**: accéder et manipuler les contenus et styles d'une page
- **window**: informations sur les fenêtre et création de nouvelles fenêtres
- ***navigator***: informations sur le navigateur (browser).



2.5 Le rapport entre DOM et Javascript

Voici un exemple de sensibilisation

url: **http://developer.mozilla.org/en/docs/The_DOM_and_JavaScript (source de cet exemple)**

- ECMAScript (ou "core Javascript") est un langage de programmation
- Les liaisons (bindings) pour ECMAScript ajoutent l'interface au DOM

Ce qui est en rouge est DOM

```
var liste_balises_a = document.getElementsByTagName("a");
for (var i = 0; i < liste_balises_a.length ; i++)
{
  alert("Href of " + i + "-th element is : " + liste_balises_a[i].href + "\n");
}
```

Quelques explications

`document.getElementsByTagName("a")`

- **`document`** est un objet qui contient tout ce qui se trouve dans un document (page)
- getElementsByTagName est une méthode qui retourne une liste de noeuds (objets) de tous les éléments <a>..

`liste_balises_a.length`

- `length` est une propriété de la classe NodeList et indique la longeur de ce type de liste

`liste_balises_a[i].href`

- `href` est une propriété de l'interface HTMLAnchorElement et retourne le contenu de l'attribut "href"
- Alternativement on aurait pu utiliser liste_balises_a.item(1).href

`alert()`

- est une méthode DOM niveau 0 pour créer un petit popup (non standarde, mais implémentée partout)

3. Eléments du DOM Core et DOM HTML et ECMAScript bindings

- Ce chapitre documente une partie du DOM core level 2 et DOM-HTML ainsi que leurs liaisons avec ECMAScript.

  url: **http://www.w3.org/TR/DOM-Level-2-Core/ (W3C Recommendation 13 November, 2000)**

  url: **http://www.w3.org/TR/DOM-Level-2-HTML/ (W3C Recommendation 09 January 2003)**

Fonction du DOM Core

- La spécification DOM niveau 2 Core se compose d'un jeu d'interfaces de base pour créer et manipuler la structure et le contenu d'un document.
- Ce module Core contient également des interfaces spécialisées réservées à XML.

Fonction du DOM HTML

- La spécification DOM niveau 2 HTML définit une interface qui permet aux programmes l'accès et la mise à jour dynamique du contenu et de la structure des documents HTML 4 et XHTML 1.x

Rapport entre DOM Core et DOM HTML

- Le modèle objet de document niveau 2 HTML repose sur le modèle objet de document niveau 2 Core.
- Autrement dit DOM HTML ajoute juste des fonctionalités spécifiques à HTML (il s'agit simplement de simplifier la vie pour le programmeur avec des "shortcuts"...)

3.1 Apperçu du modèle objet de document Core

Le modèle DOM niveau 2 code définit un document XML ou HTML comme hierarchie de noeuds (node) de certains types.

Anatomie du contenu d'un arbre DOM défini par le DOM 2 Core Interface:

Type de noeuds	Noeuds enfants
Document	• Element (un, au maximum), ProcessingInstruction, Comment, DocumentType (un, au maximum)
DocumentFragment	• Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
DocumentType	• aucun enfant
EntityReference	• Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Element	• Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
Attr	• Text, EntityReference
ProcessingInstruction	• aucun enfant
Comment	• aucun enfant
Text	• aucun enfant
CDATASection	• aucun enfant
Entity	• Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Notation	• aucun enfant

- Voir aussi 3.2 "Types de données" [15]

Pour chacun de ces types de noeuds il existe des méthodes et propriétés pour accéder aux données ou pour les modifier.

Eléments supplémentaires:

- une interface ***NodeList*** pour la gestion de listes ordonnées d'objets Node (typiquement on a cela après avoir fait une requête XPath etc.)
- une interface ***NamedNodeMap*** pour la gestion de noeuds référencé par leur nom d'attribut,

DOM HTML spécialise cet arbre, c.a.d. il existe des types d'objets pour toutes sortes d'éléments spécifiques à HTML, par exemple:

- 'Document' devient 'HTMLDocument'

3.2 Types de données

Tableau des types des données DOM Core les plus importants:

Classes (Interfaces du DOM)	variable ECMAScript	fonction
Node		Implémente les noeuds d'un arbre DOM (par exemple document, element, attributes, etc. et qui héritent ses méthodes et propriétés)
Document	document	L'objet document représente un document HTML, XHTML et XML (implémente l'interface "Document".
Element		element se réfère à un élément (représentation d'une balise et de son contenu). Element hérite de Node (interface pour les nœuds).
NodeList		Une nodeList est un tableau d'éléments, comme celui qui est renvoyé par la méthode document.getElementsByTagName(). Les éléments d'une nodeList sont accessibles par un index de deux manières différentes (mais identiques quand au résultat): • list.item(1) • list[1]
Attribute		Une référence à un objet qui représente un attribut. Un attribut est extension de Node.
NamedNodeMap		Une namedNodeMap est comme un tableau, mais où l'on peut accéder aux éléments à la fois par nom ou par index

Tableau des types de qqs. données du monde HTML-DOM:

- Le modèle objet de document (DOM) niveau 2 HTML de plusieurs autres spécifications, notamment DOM Core

Classes	variable ECMAScript	fonction
Window AbstractView	window	L'objet window représente la fenêtre elle-même. Window contient le document et le navigator parmi ses enfants Note: 'Window' ne fait pas partie du standard W3C !
HTMLDocument Document	document window.document	Un objet HTMLDocument représente la racine de la hiérarchie HTML et porte la totalité du contenu. Hormis un accès à la hiérarchie, il offre aussi des méthodes commodes pour obtenir certains ensembles d'informations du document.
HTMLElement Element		Représente un élément HTML
HTML...Element		Il existe des interfaces spécialisées pour la plupart des éléments HTML (environ 50), par exemple HTMLHeadElement, HTMLButtonElement,...

Suite ....

- Dans la suite nous allons plutot présenter DOM & Javascript par types d'objets (donc différents niveaux et éléments de spécifications confondus).

4. HTML Window

- Cette interface non-standardisée permet de manipuler le conteneur (fenêtre) d'un document HTML, ou encore de créer des fenêtres, popups, etc.
- Window repose sur l'interface AbstractView du DOM Level 2 Views Specification
- Un standard est en discussion depuis un bon moment

  url: **http://www.w3.org/TR/Window/**

  Note: Il existe pleins d'autres propriétés et méthodes que celles présentés ci-après

4.1 Contenu de la fenêtre

window.document

- retourne l'objet document associé à une fenêtre

```
doc= window.document;
// affiche le titre du document courant dans la console.
window.dump(doc.title);
```

window.navigator

- retourne une référence vers l'objet navigateur

```
nav = window.navigator;
```

- Note: l'objet navigateur vous permet ensuite de tester l'environnement de l'utilisateur

```
alert ("Langage de votre navigateur: " + nav.language);
```

Voir 5. "Navigator" [21]

window.history

```
histoire = window.history;
alert ("Vous avez " + histoire.items + " dans l'historique de votre nav");
```

window.status

- lit ou modifie la "status bar" (en bas de la fenêtre)
  `window.status = "le site le plus cool";`
- Le code suivant montre comment changer la status bar lorsque l'utilisateur glisse la souris sur un lien
- marche uniquement si votre Navigateur l'autorise ...

```
<a href="edutechwiki.unige.ch/en"> onmouseover="status='Consultez
EduTechWiki'; return true;">EduTech Wiki</a>
```

window.print()

- envoie le contenu de la fenêtre sur l'imprimante
  `window.print();`

Exemple 4-1: Quelques informations sur la fenêtre

url: **http://tecfa.unige.ch/guides/js/ex/dom-intro/window.html**

4.2 Popups

- DOM Level 0 (pas standardisé)

window.alert(message)

```
window.alert("Hello tout le monde");
```

result = window.confirm(message);

```
result = window.confirm("Voulez-vous écraser votre disque dur ?");
if (result) alert ("ok c'est fait");
```

window.prompt(text, value)

text est le message affiché
value est la valeur par défaut (à option)

```
result = window.prompt("Quelle est votre couleur préférée ?");
window.alert("Ah vous aimez le " + result);
```

Note:

- Ces methodes sont aussi disponibles sous forme courte, c.a.d. des fonctions:
- alert() - confirm() - prompt()

4.3 Fenêtres

window.open(strUrl, strWindowName [, strWindowFeatures]);

- crée une nouvelle fenêtre

  url: **http://developer.mozilla.org/en/docs/DOM:window.open**
- strUrl indique l'URL de la page à ouvrir. Pour obtenir une fenêtre vide utiliser un string vide:
  `window.open("", "fenetre vide");`
- StrWindowName: vous devez donner une nom à la fenêtre (un string)
- StrWindowFeatures vous permet d'indiquer une liste de propriétés de la fenêtre

Exemple:

```
win = window.open("http://tecfa.unige.ch/guides/tie/tie.html",
       "transparents",
       "resizable=yes,scrollbars=yes,status=yes,toolbar=false");
```

- Note: réfléchissez bien avant d'enlever la toolbar ... les gens détestent cela.
- Attention: les features se trouven dans un seul string !!

5. Navigator

- l'objet navigateur se trouve référencé dans l'objet fenêtre
- Il s'agit de DOM level 0 .... donc pas standardisé !

nav = window.navigator

- Selon les implémentations, il existe des propriétés pour identifier précisément type de navigateur, plugins installés etc.

Exemple 5-1: Détection du navigateur et redirection

url: **http://tecfa.unige.ch/guides/js/ex/dom-intro/navigator.html**

Exemple 5-2: Afficher quelques propriétés du navigateur

url: **http://tecfa.unige.ch/guides/js/ex/dom-intro/navigator-props.html**

6. Noeuds et éléments

6.1 DOM Node et DOM Element

- La plupart des objets DOM HTML héritent du DOM Core.

Pour rappel, un aperçu des types d'objets DOM universels les plus importants:

Node

- implémente le noeud de base dans un arbre XML ou HTML
- implémente toutes les propriétés et méthodes pour examiner et pour traverser un arbre

Element et HTMLElement

- (hérite de Node)
- implémente les éléments (donc ce qui correspond à une balise et son contenu)
- implémente surtout les propriétés et méthodes nécessaires pour manipuler des attributs
- Par rapport à Element, HTMLElement implémente des fonctionalités supplémentaires

Text

- (hérite de Node et de CharacterData)
- implémente les CDATA (texte à l'intérieur d'une balise).

6.2 Tests

hasAttribute

- Returne une valeur booléenne qui indique si l'attribut existe ou pas.

hasAttributeNS

- pareil, mais pour un namespace

hasAttributes

- Returne une valeur booléenne qui indique si l'élément possède des attributs.

hasChildNodes

- Retourne une valeur booléenne qui indique si l'élément courant a des enfants ou non.

6.3 Informations sur l'élément

nodeName

- Retourne le nom du noeud.

nodeType

- Retourne les type d'un noeud sous forme d'un nombre entier
- Ces informations sont très importants pour savoir ce qu'un programme doit faire avec un noeud.

Résumé des types de noeuds et leur nom sous forme de constante ou de valeur

Interface (type de noeud)	Nom de la constante	Valeur retournée par nodeType
Element	Node.ELEMENT_NODE	1
Attribut	Node.ATTRIBUTE_NODE	2
Noeud texte	Node.TEXT_NODE	3
	Node.CDATA_SECTION_NODE	4
	Node.ENTITY_REFERENCE_NODE	5
	Node.ENTITY_NODE	6
Instruction de traitement	Node.PROCESSING_INSTRUCTION_NODE	7
Commentaire	Node.COMMENT_NODE	8
	Node.DOCUMENT_NODE	9
	Node.DOCUMENT_TYPE_NODE	10
Fragment XML	Node.DOCUMENT_FRAGMENT_NODE	11
	Node.NOTATION_NODE	12

Exemple 6-1: Tester si le premier noeud d'un document est un commentaire

```
var node = document.documentElement.firstChild;
if(node.nodeType != Node.COMMENT_NODE)
  alert("You should comment your code well!");
```

6.4 Contenu d'un élément

childNodes

- Renvoie une liste des nœuds enfants de l'élément.
  ```
  var collNoeuds = elementDeReference.childNodes;
  ```

className

- Définit ou obtient la classe de l'élément.

firstChild

- Renvoie le premier enfant direct du nœud.

id

- Définit ou obtient l'identifiant (id) de l'élément courant.

innerHTML (pas compatible W3C DOM !)

- Renvoie ou définit l'ensemble du balisage et du texte contenu au sein d'un élément donné.
- Balisage est une chaîne contenant le contenu de l'élément (y compris ses sous-éléments) au format HTML brut. Par exemple, " Du texte ".
  ```
  var balisage = element.innerHTML;
  element.innerHTML = balisage;
  ```

lastChild

- Retourne le dernier enfant d'un noeud.

NodeList.length

- Returne le nombre d'items dans une liste (e.g. childNodes).

nodeValue

- Returne la valeur d'un noeud.

textContent

- Retourne/définit le contenu texte d'un élément.

```
text = element.textContent
element.textContent = "this is some sample text"
```

appendChild(noeud)

- Insère le nœud spécifié à la fin de la liste de nœuds de l'élément courant.

```
// Create a new paragraph element
var p = document.createElement("p");
// Append it to the end of the document body
document.body.appendChild(p);
```

insertBefore (nouvel_élément, élément_cible)

- La méthode insertBefore permet d'insérer un nœud juste avant un élément de référence parmi les enfants du nœud courant.

```
parentDiv = document.getElementById("div-parent");
span2 = document.getElementById("span-enfant");
span1 = document.createElement("span");
parentDiv.insertBefore(span1, span2);
```

6.5 Voisinage d'un élément

parentNode

- Retourne le noeud parent.

nextSibling

- Renvoie le nœud suivant immédiatement dans l'arbre (petit frère)

```
// dans un tableau, les cellules sont des enfants
cell1 = document.getElementById("td1");
cell2 = cell1.nextSibling;
```

previousSibling

- Retourne le noeud précédant (grande soeur).

6.6 Style

style

- retourne l'objet qui contient les définitions du style.
- cf. chapitre 11. "HTML Style" [39]

6.7 Attributs

attributes

- Renvoie une collection des attributs de l'élément.

```
// récupère le premier élément <p> du document
var para = document.getElementsByTagName("p")[0];
// récupère ses attributs
var attr = para.attributes;
```

getAttribute(nom_attribut)

- Returns the value of the named attribute on the current node.

```
var div1 = document.getElementById("div1");
var align = div1.getAttribute("align");
alert(align); // shows the value of align for the element with id="div1"
```

getAttributeNode()

- Returne l’attribut du noeud en tant que noeud, c.f. 8. “HTML Attribut” [34]

setAttribute(nom, valeur)

- Ajoute un attribut avec une valeur ou change la valeur d’un attribut qui existe.

setAttributeNode()

- Ajoute un noeud attribute à l’élément courant.

setAttributeNodeNS()

- Pareil, mais avec un namespace.

7. HTML Document

- Dans le DOM Core, l'objet document représente tout le contenu d'un document
- DOM:HTMLDocument est une interface spécialisée pour le traitement des documents HTML (par exemple avec document.cookie ou document.alinkColor).

7.1 Ouvrir, écrire (dans) et fermer un document:

document.open()

- Ouvre un flux pour l'écriture dans le document et qui écrase tout

document.write(balisage)

- Écrit du texte dans le document.

document.close()

- Un document ouvert peut être fermé à l'aide de document.close().

```
document.open(); // ouvre un document
document.write("<h1>Hello !</h1>");
document.close();
```

7.2 Extraction d'éléments et d'attributs

document.getElementById(id_élément)

- Renvoie une référence à l'objet correspondant à l'identifiant donné.
- Il s'agit ici d'une méthode très populaire !
  `document.getElementById("p1")`

document.getElementsByName(nom_élément)

- Renvoie une liste des éléments ayant le nom donné.

```
//<div name="up">200</div>
//<div name="up">145</div>
//<div name="other">178</div>
up_divs = document.getElementsByName("up"); // retourne une liste de nodes
dump(up_divs.item(0).tagName); // returns "div"
```

document.getElementsByTagName(nom_balises)

- Renvoie une liste des éléments avec le nom de balise donné dans le document
  `elements = document.getElementsByTagName(h1); //retourne une liste des h1`

Exemple 7-1: Extraction d'éléments avec leur id

```
// pour <p id="p1" class="reggy" align="right">texte</p>
premierParagraphe = document.getElementById("p1");
listeAttributs = premierParagraphe.attributes;
```

Quelques propriétés:

- il existe plusieurs propriétés pour extraire qqs. éléments spécifiques, par ex.

document.forms

- forms renvoie une liste des éléments FORM du document courant

document.body

- body renvoie le nœud BODY du document.

document.links

- Renvoie un tableau de tous les liens du document.

document.referrer

- Renvoie l'URI de la page qui a amené à cette page.

document.images

- images renvoie une liste des images du document.

```
var listeimg = document.images;
for(var i = 0; i < listeimg.length; i++) {
    if(listeimg[i] == "xxx.gif") {
      alert("Attention aux noms d'images"!)
    }
}
```

Exemple 7-2: Tree walking: Collectionner les noms d'éléments

url: **http://tecfa.unige.ch/guides/js/ex/tree-walking/tree-walking.html**

```
    var node_list = "";

    function do_document () {
      mark_tags(document.body);
      alert ("This text has the following elements: " + node_list);
    }

function mark_tags(node) {
    // Check if n is an Element Node
    if (node.nodeType == 1 /*Node.ELEMENT_NODE*/)  {
     // Append the node name to the list
     node_list += node.nodeName + " ";
     // Let's see if there are children
     if (node.hasChildNodes()) {
         // Now get all children of n
         var children = node.childNodes;
         // Loop through the children
         for(var i=0; i < children.length; i++) {
                 mark_tags(children[i]);      // Recurse on each one
         }}   } }
</script>
</head><body>
   <h1>Tree walking</h1>
    <input type=button onClick="do_document();" value="Show body node names">
```

7.3 Création et modification d'éléments

document.createElement (nom_du_tag)

- Crée un nouvel élément.

```
var div = document.createElement("div");
```

document.createTextNode (texte)

- Crée un nœud de texte.

parent_node.insertBefore(new_node, existing_node)

- insère un (nouveau) élément avant un autre élément

```
preface.insertBefore(new_div_node, titre)
```

Exemple 7-3: Création d'un élément et insertion dans le DOM

url: **http://tecfa.unige.ch/guides/js/ex/dom-intro/document-trans1.html**

```
function change_me() {
      var doc       = window.document;
      var preface   = doc.getElementById("preface");
      var titre     = doc.getElementById("preface_title");

      // on cree un élément div
      var new_div_node = doc.createElement("div");
      // on cree un nouveau TextNode
      var contenu  = doc.createTextNode ("Auteur: DKS et collaborateurs");
      // on le met dans le div
      new_div_node.appendChild(contenu);

      // et on insere ce div devant l'élément qui a l'ID "preface"
      preface.insertBefore(new_div_node, titre);
```

```
}
.....
```

Voici le fragment HTML:

```
<div id="preface">
   <h2 id="preface_title">La préface</h2>
   blabla (qui a écrit cela ?)
 </div>
 <p>  <button onclick="change_me();">Transformez-moi !</button>  <p>
```

Exemple 7-4: Création d'un élément et insertion dans le DOM 2

- Voici 2 exemples similaires

  url: **http://tecfa.unige.ch/guides/js/ex/dom-intro/insert3.html**

  url: **http://tecfa.unige.ch/guides/js/ex/dom-intro/insert4.html**

8. HTML Attribut

- (à faire)

9. HTML Table

- (à faire)

10.HTML Form et HTML Elements

url: **http://developer.mozilla.org/en/docs/DOM:form**

- HTMLFormElement hérite de Element (donc de ses méthodes et propriétés)
- Ci-dessous on utilise "form" et "element"

form.elements

```
nodeList = form.elements
```

- retourne un tableau (array) de toutes les "contrôles" (balises pour formulaire) d'un formulaire

form.length

- indique le nombre des contrôles (utile pour programmer une boucle)
- nodeList = HTMLFormElement.elements

Accès aux éléments

- soit par leur indexe, soit par leur id ou leur name.

Accès par id

```
var inputs = document.getElementById("form1").elements;
```

Accès par indexe

```
var inputByIndex = inputs[2];
```

Accès par nom

```
var inputByName = inputs["login"];
```

Exemple 10-1: Interroger les valeurs d'un formulaire

url: **http://tecfa.unige.ch/guides/js/ex/forms/form2.html**

- Cette fonction traite le formulaire
- **`form = document.intoxication`** // assigne une référence du formulaire à form
- la boucle additionne **`form.elements[i].value`** à partir du 2ème élément !

```
function traiter() {
  var scale=0;
  var form = document.intoxication
    for (var i=1; i < (form.length-1); i++)  {
      if (form.elements[i].checked)  {
     scale = scale + parseInt(form.elements[i].value);
      }
    }
 // Ici une series de if's (c.f. le source)
  if (scale<8) {
    alert("Bonjour, " + form.nomdutoxicomane.value
       + ". Avec ton score de " + scale + ", tu es completement sain.")
      }
  else { .... }
```

```
<form name="intoxication">

      <h2>Interactive test</h2>
      <P>
    Quel est ton nom ?
    <input type="text" name="nomdutoxicomane">
      <P>
    Vous buvez de l'alcool :
    <input type="radio" name="choice1" value="1" checked>jamais
    <input type="radio" name="choice1" value="2">moins d'un litre par semaine
    <input type="radio" name="choice1" value="3">plus d'un litre par semaine
      <P>

    Quand vous buvez de l'alcool, vous finissez votre verre :
    <input type="radio" name="choice2" value="1" checked>jamais
    <input type="radio" name="choice2" value="2">parfois
    <input type="radio" name="choice2" value="3">toujours
      <P>

   [ ..... ]

    <input type="button" name="process" value="Voir les résultats"
onClick="traiter()">

   </form>
```

Exemple 10-2: Transformer le contenu d'un formulaire

url: **http://tecfa.unige.ch/guides/js/ex/forms/form1.html**

Cette fonction ramasse qqs. informations sur le formulaire et les affiche dans le contenu de la textarea.

```
function getFormInfo() {
      var f = document.forms["myform"];
      var info = "f.elements: " + f.elements + "\n"
      + "f.length: " + f.length + "\n"
      + "f.name: " + f.name + "\n"
      + "f.action: " + f.action + "\n"
      + "f.method: " + f.method + "\n"
      + "f.target: " + f.target;
      document.forms["myform"].elements['tex'].value = info;
      }
```

Cette fonction modifie le formulaire

```
function setFormInfo(f) {
      f.method = "GET";
      f.action = "Envoi_vers_python";
      f.name   = "Voici_un_nom_qui_jette";    }
.....
 <form name="myform" id="myform" action="Envoi_vers_php" method="POST">
    <input type="button" value="info"  onclick="getFormInfo();">
    <input type="button" value="set"   onclick="setFormInfo(this.form);">
    <textarea id="tex" style="height:15em; width:40em"></textarea>
 </form>
```

11.HTML Style

- Il est possible d'accéder et de modifier
  - la feuille de style du document
  - le résultat combiné des styles en cascade pour un élément
  - le style "inline" pour un élément défini

11.1Modifier le style CSS "inline"

- Dans cette section on montre comment changer le style "inline" d'un élément individuel.
- En règle générale, on utilise cette technique pour faire des pages interactives et/ou animés.
- Le principe est simple, pour accéer ou modifier une propriété CSS on passe par:

element.style.propriete_CSS

```
// supposons que para12 soit un élément précis de votre DOM
para12.style.color = "blue"
para12.style.fontSize = "18pt"
```

Liste des propriétés DOM CSS

- Attention (!!!) Les propriétés CSS du DOM ne correspondent pas aux noms "CSS"
- Pour une liste, cf:

  url: **http://developer.mozilla.org/en/docs/DOM:CSS**

  url: **http://www.w3.org/TR/DOM-Level-2-Style/css.html#CSS-CSS2Properties**

- En gros, on peut deviner le nom DOM:
  - il faut enlever le tiret "-"
  - il faut capitaliser les mots (sauf le premier)
  - par exemple: la propriété CSS "background-color" devenient en DOM CSS "backgroundColor".
- Voici juste quelques exemples:

DOM CSS	Propriété CSS
background	background
backgroundColor	background-color
borderColor	border-color
fontFamily	font-family
fontSize	font-size
marginBottom	margin-bottom
marginLeft	margin-left

Exemple 11-1: Changer dynamiquement le style d'un élément

url: **http://tecfa.unige.ch/guides/js/ex/dom-intro/change-style1.html**

- montre comment changer le style d'un lien spécifique

```
<script type="text/javascript">
    function changeText() {
    var p = document.getElementById("pid");
     p.style.color = "blue"
     p.style.fontSize = "18pt"
    }
    </script>
.....
<p id="pid"
    onclick="window.location.href = 'http://www.cnn.com/';">News Link</p>
    <form> <p>
    <input value="Change style" type="button" onclick="changeText();">
</p></form>
```

Exemple 11-2: Changer dynamiquement le style (2)

url: **http://tecfa.unige.ch/guides/js/ex/dom-intro/change_style2.html**

- montre comment changer entre 2 styles pour un lien spécifique

Exemple 11-3: Bouger un objet

url: **http://tecfa.unige.ch/guides/js/ex/dom-animate/move-object1.html**

```
function moveitBy(img, x, y){
      var obj          =  document.getElementById(img);
      obj.style.left   =  parseInt(obj.style.left)+x+"px"
      obj.style.top    =  parseInt(obj.style.top)+y+"px"
      }
      function moveitTo(img, x, y){
      var obj          = document.getElementById(img);
      obj.style.left   = x+"px"
      obj.style.top    = y+"px"
      }
```

- la fonction parseInt () va extraire un nombre d'un string à partir des premières chiffres (!)

```
parseInt("10px") == 10; parseInt("10bla20") == 10; pareInt("px10") == NaN;
```

Voici la partie HTML

```
<p><div id="image" style="position: relative; left: 0px; top: 0px;"><img src="tecfa-
affiche.gif"></div>

      <ul>
     <li><a href="javascript:moveitBy('image', 10, 0);">
        Move by 10px (right)</a></li>
 <li><a href="javascript:moveitBy('image', -10, 0);">Move by 10px (left)</a></li>
 <li><a href="javascript:moveitBy('image', 0, 10);">Move to 10px (down)</a></li>
 <li><a href="javascript:moveitBy('image', 0, -10);">Move to 10px (up)</a></li>
 <li><a href="javascript:moveitTo('image', 0, 0);">Move to original position</a></li>
      </ul>
```

Exemple 11-4: Animation en continu

url: **http://tecfa.unige.ch/guides/js/ex/dom-animate/dynamicposition2.html**

- Une fonction start va toutes les 100 millisecondes lancer la fonction "run".

```
function start() {
    pText = document.getElementById("pText");
    timer = window.setInterval( "run()", 100 );
  }
```

- La méthode window.clearInterval(timer) permet de tuer ce processus
- La fonction run() fait l'animation.

```
 ....
 pText.style.fontSize = size + "px";
 pText.style.left = count + "px";
```

- Une expression teste dans quel sens il faut aller (aggrandir ou diminuer)

```
if ( ( count % 200 ) == 0 ) {
    speed *= -1;
    pText.style.color = ( speed < 0 ) ? "red" : "blue" ;
    firstLine =         ( speed < 0 ) ? "Text shrinking" : "Text growing";
 }
```

Exemple 11-5: Tree walking: changer de style

url: **http://tecfa.unige.ch/guides/js/ex/tree-walking/tree-walking2.html**

```
<script>

 function do_document (css_class,color) {
      mark_tags(document.body,css_class,color);
    }

 function mark_tags(node,css_class,color) {
   // Check if n is an Element Node
   if (node.nodeType == 1 /*Node.ELEMENT_NODE*/)  {
    // Highlight if needed
    if (node.getAttribute("class") == css_class)
       node.style.color = color;
    // Let's see if there are children
    if (node.hasChildNodes()) {
        // Now get all children of node
        var children = node.childNodes;
        // Loop through the children
        for(var i=0; i < children.length; i++) {
                // Recurse on each child
                mark_tags(children[i],css_class,color);
        } }}
   }
</script>
</head>
```

```
<body>
    <h1>Tree walking</h1>
    <input type=button onClick="do_document('important','red');"
           value="Highlight important stuff">
    <input type=button onClick="do_document('boring','grey');"
    value="Highlight boring stuff">
.....
   <p class="boring">Let's hope you enjoy this</p>
   <p class="important">If you want to reuse this code:</p>
.....
```

Exemple 11-6: Tree walking: changer de style

url: **http://tecfa.unige.ch/guides/js/ex/tree-walking/tree-walking3.html**

- Code plus général que tree-walkin2.html et qui permet de transformer n'importe quelle propriétée de style

11.2Utilisation de setAttribute

element.setAttribute("style,"définition CSS .....")

- on peut aussi directement (re)définir la propriété "style" d'un élément et là on travaille directement avec CSS.
- Mais attention: cette opération écrase tous les autres styles inline associés à cet élément (à vérifier si cela comprend les styles hérités, sorry !). A utiliser avec prudence en tout cas.

```
var el = document.getElementById("some-element");
el.setAttribute("style", "background-color:darkblue;");
```

12.HTML Event

- HTML fournit une interface très puissante pour gérer les événements
- La différence entre "Event listener", "Event handler" et "Event".
  - Un "Event listener" observe si un événement a lieu
  - Un "Event handler" fait quelque chose avec l'événement
  - Un "Event" est un événement (l'utilisateur a fait qc. et cela se traduit en un objet de type Event)

12.1Rappel des événements prédéfinis dans Gecko

- Dom Event Level 2 en définit moins (les keys) !

  url: **http://www.w3.org/TR/DOM-Level-2-Events/events.html#Events-eventgroupings**
- Voici un tableau avec quelques événéments importants pour les objets de type HTML Element et Window.

Event Handler	Event	element	window	Utilisateur
onblur	**blur**	x	x	on focalise sur un autre objet (bouger, cliquer, etc ailleurs)
onclick	**click**	x	x	on clique sur un objet
ondblclick	**dblclick**	x		
onfocus	**focus**	x	x	on focalise sur l'objet
onkeydown	**keydown**	x	x	une touche reste enfoncée
onkeypress	**keypress**	x	x	une touche est enfoncée
onkeyup	**keyup**	x	x	
onmousedown	**mousedown**	x	x	clique avec la source
onmousemove	**mousemove**	x	x	on bouche la souris

Event Handler	Event	element	window	Utilisateur
onmouseout	**mouseout**	x	x	on déplace la souris ailleurs
onmouseover	**mouseover**	x	x	on déplace la souris sur l'objet
onmouseup	**mouseup**	x	x	on lache un bout de souris
onresize	**resize**	x	x	on diminue une fenetre
onload	**load**		x	on charge une fenêtre dans les navi.
onunload	**unload**			contenu part (on charge un autre document)
onclose	**close**		x	on ferme une fenêtre
ondragdrop	**dragdrop**		x	
onscroll	**scroll**		x	on scroll le contenu d'une fenêtre

Principe

- Lorsqu'on a une instruction HMTL/JS suivante:

```
<div onclick="traitement();">click me!</div>
```

- Dans le DOM une propriété représente cela:

```
<div id="mon_div" onclick="traitement();">click me!</div>
mon_el = document.getElementById("mon_div");
mon_el.onclick // retourne le nom de la fonction qui gère le onclick
```

- par défaut, la valeur DOM de **`mon_el.onclick`** est une fonction "Event Listener" prédéfinie:

```
function onclick(event) {
   traitement (); }
```

- Les noms des Event listeners sont définis dans les propriétés de l'objet (onload, onclose, etc.)
- Lors d'une invocation, ces fonctions reçoivent comme argument un objet "Event" qu'on peut exploiter.

12.2Construire des Event Listeners

- Pour assigner des Event Listeners, il existe 3 façons de faire:

A. Comme nos parents: inline

- A éviter car on mélange HTML et Javascript
- A éviter car (en tout cas moi je ne sais passer l'objet "événément"), c.f. page 52

url: **http://tecfa.unige.ch/guides/js/ex/dom-event/simple-event-listener0.html**

```
<script type="text/JavaScript">
function showPopup() {
 alert("Attention on va vous empester avec une pub !!" );
 var popup = window.open("http://tecfa.unige.ch/maltt/", "popup",
                         "height=800,width=600");
       popup.focus();
       return false;
      }
</script>

<body onload="load();">
    <div onClick="Javascript:showPopup();">CLICK ME !!</div>
```

- On déclenche un évent "click" en cliquant sur le texte "CLICK ME"
- Le Event Listener "onClick" va faire appel à la fonction Event Handler "showPopup".

B. Informel (marche avec IE et Firefox/Mozilla)

Exemple 12-1: Event Listener façon rapide

url: **http://tecfa.unige.ch/guides/js/ex/dom-event/simple-event-listener.html**

```
function load() {
  var p = document.getElementById('mon_div');
  p.onclick = showPopup; // Il s'agit de donner le NOM de la fonction
  }

function showPopup() {
  alert("Attention on va vous empester avec une pub !!" );
   var popup = window.open("http://tecfa.unige.ch/maltt/", "popup",
                           "height=800,width=600");
       popup.focus(); return false;
     }
.....
<body onload="load();">
<div id="mon_div">CLICK ME !!</div>
```

- Lorsque le document charge, on assigne le nom de la fonction showPopup à la propriété "onclick" de l'élément en question (le "div" du CLICK ME).
- Ensuite, la fonction showPopup gérer l'événement vous voulez ...

C. Selon la spécification DOM addEventListener

- Cette méthode est plus puissante puisqu'elle permet d'enregistrer plusieurs Event Listeners du même type.
- Ne marche pas avec IE 6 (4/2006), MS a son truc propriétaire équivalent.

Exemple 12-2: L'interface addEventListener

url: **http://tecfa.unige.ch/guides/js/ex/dom-event/addEventListener2.html**

```
// Exemple 1 d'enregistrement d'évènement
 function l_func() {
   var t2 = document.getElementById("t2");
   t2.innerHTML = "trois c'est mieux que deux !";
 }

 // Exemple 2 d'enregistrement d'évènement
 function touche () {
   alert ("je t'ai vu toucher !!");
   document.getElementById("table2").style.border = "yellow";
 }

 function load() {
    var el = document.getElementById("table");
    el.addEventListener("click", l_func, false);
    var el2 = document.getElementById("table2");
    el2.addEventListener("mouseover", touche, false);
 }
```

12.3Gestion de l'événement

- Lorsqu'un évent handler fait appel à une fonction, le handler lui transmet un objet "Event". Cet objet possède des informations utiles qu'on peut exploiter

  url: **http://tecfa.unige.ch/guides/js/ex/dom-event/examine-events2.html**

- A documenter ...

```
var x_td;
      var y_td;

      // Exemple  d'enregistrement d'évènement
      function touche (event) {

        x = event.pageX;
        y = event.pageY;
        obj = event.target;
       // alert ("je t'ai vu toucher \n aux positions x=" + x + ",y=" + y + "\n à l\'objet
" + obj);
        x_td.innerHTML = x;
        y_td.innerHTML = y;
        obj_td.innerHTML = obj.innerHTML;
      }


      function load() {
         var el2 = document.getElementById("table2");
         if (el2.addEventListener) el2.addEventListener("mouseover", touche, false)
         // for those who can't use a real browser
           else alert ("Sorry you need a browser that understands DOM Level 2 Events");
         x_td = document.getElementById("x");
         y_td = document.getElementById("y");
```

```
            obj_td = document.getElementById("obj");
         }
      </script>


   </head>

   <body onload="load();">
      <h1>Gecko DOM addEventListener example</h1>

Ceci est une démo d'un EventHandler pour l'événement mouseover enregistré pour la table.
Bougez la souris sur le tableau SVP:


      <table align="center" id="table2">
   <tr><td>Ma tete</td></tr>
   <tr><td>Ma poitrine</td></tr>
   <tr><td>Mon ventre</td></tr>
   <tr><td>Mes jambes</td></tr>
      </table>

<p>
Tu a commencé à me toucher aux coordonnées:
      <table align="center" id="coordinates">
   <tr><td>X = </td>  <td id="x"> </td></tr>
   <tr><td>Y = </td>  <td id="y"> </td></tr>
   <tr><td>Objet = </td>  <td id="obj"> </td></tr>
      </table>
```

13.AJAX

- Explications à faire, pour le moment c.f. autres sites, par exemple:

  url: **http://developer.mozilla.org/en/docs/AJAX**

  Asynchronous JavaScript and XML (AJAX) is not a technology in itself, but is a term that describes a "new" approach to using a number of existing technologies together, including: HTML or XHTML, Cascading Style Sheets, JavaScript, The Document Object Model, XML, XSLT, and the XMLHttpRequest object.

Exemple 13-1: AJAX request

url: **http://tecfa.unige.ch/guides/js/ex/ajax/ajax0.html**

- Montre comment demander une page (GET) dont le contenu sera affiché dans un alert

Exemple 13-2: AJAX request et pseudo XML response

url: **http://tecfa.unige.ch/guides/js/ex/ajax/ajax1.html**

url: **http://tecfa.unige.ch/guides/js/ex/ajax/ajax1.phps**

- Montre comment envoyer une requête au serveur
- Cette requête est construite à partir d'un geste d'utilisateur
- On affiche la réponse XML "texte" du serveur dans une fenêtre popup (sans la traiter)

Exemple 13-3: AJAX request et XML response

url: **http://tecfa.unige.ch/guides/js/ex/ajax/ajax2.html**

On analyse les réponses du serveur en tant que XML DOM en utilisant l'API DOM Core

- Ensuite, on insère quelque chose dans la page HTML

- Documentation incluse dans le code (en Anglais)

Code HTML et JavaScript

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Simple Ajax example</title>
<script type="text/javascript" language="javascript">
var url;
var table;

function init () {
  url = "ajax1.php";
 // url = "ajax-debug.php";
  table = document.getElementById("table1");
}

function makeRequest(element) {
  // This function is called from the HTML code below
  // element is the DOM element (tag on which the user clicked)

  var http_request = false;

  // ---- Mozilla, Safari, etc. browsers
  if (window.XMLHttpRequest) {
    http_request = new XMLHttpRequest();
   // This will make sure that the server response claims to be XML (in case we retrieve
something else)
    if (http_request.overrideMimeType) {
      http_request.overrideMimeType('text/xml');
    }
```

```
  // ---- IE browsers
} else if (window.ActiveXObject) {
  try {
    http_request = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
    http_request = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (e) {}
  }
}

// ---- abort if there is no reply
if (!http_request) {
  alert('Giving up :( Cannot create an XMLHTTP instance');
  return false;
}

// We register the function that will deal with a reply
http_request.onreadystatechange = function() { processServerReply(http_request); };

// This lines starts building the request - don't think that a GET request would work (?)
http_request.open('POST', url, true);
// Contents WE send from here will be urlencoded
http_request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

// here we extract the contents of tag on which the user clicked
user_pref    = element.innerHTML;
// This is the content of the request
// alert(user_pref);
user_request = "user_pref_fruit=" + user_pref;
```

```
  // We send the data - data are query strings
  http_request.send(user_request);
}

function processServerReply(http_request) {

  if (http_request.readyState == 4) {
    if (http_request.status == 200) {
      // We tell the server that we want to deal with XML as a DOM Document !!
      replyXML = http_request.responseXML;
      treatResponse (replyXML);
      // displayResponse (replyXML);
    } else {
      alert('There was a problem with the request.');
    }
  }
}

// This will change the HTML contents of the page
function treatResponse (reply) {
  // reply is a XML DOM datastructure !!
  // extract some XML - we know that it is in an "answer" tag
  // DOM HTML will not work, it's XML here
  var answer = reply.getElementsByTagName('answer').item(0).firstChild.nodeValue;

  // new tr, td elements
  var element_tr = document.createElement("tr");
  var element_td = document.createElement("td");
  // contents for the td element
  var text    = document.createTextNode(answer);
  element_td.appendChild(text);
```

```
  element_tr.appendChild(element_td);
  table.appendChild(element_tr);
}

// just for debugging, will open up a popup window. Useful if you tell the php to send
debugging infos...
function displayResponse (reply) {

win=window.open("","Results","width=250,height=300,status=1,resizable=1,scrollbars=1")
;
  win.document.open();
  win.document.write(reply);
  win.document.close();
}
 </script>

</head>
  <body onload="init()">
    <h1>Simple Ajax example</h1>

    <strong>Please</strong> click on a fruit:

    <ul>
      <li>I like
 <span
   style="cursor: pointer; text-decoration: underline"
   onclick="makeRequest(this)">
   apples
 </span>
      </li>
      <li>I like
```

```
<span
  style="cursor: pointer; text-decoration: underline"
  onclick="makeRequest(this)">
  oranges
</span>
     </li>
     <li>I like
<span
  style="cursor: pointer; text-decoration: underline"
  onclick="makeRequest(this)">
  bananas

</span>
     </li>

   </ul>

   <hr>
Dialog history:

  <table border id="table1">
<tr>
  <!-- one of (TD TH) -->
  <th>Server replies</th>
</tr>

   </table>

   <hr>
   <strong>Explanations</strong>:
   <p>
```

```
    This example is inspired by <a href="http://developer.mozilla.org/en/docs/
AJAX:Getting_Started">http://developer.mozilla.org/en/docs/AJAX:Getting_Started</a>

It does the following:
    <ul>
      <li>When you click on a fruit it will call a function that grabs the element,
extracts its contents and sends it to a php file with a POST argument that goes like
"user_pref_fruit=appeles"</li>
      <li>The server will give some answer that will be put into a DOM document. It's
contents are.
<pre>
&lt;?xml version="1.0" encoding="ISO-8859-1" ?&gt;
   &lt;answer&gt;Oh you like  bananas!&lt;/answer&gt;
</pre>
      </li>
      <li>You then can use the XML DOM Code API to take the answer apart .... We just
extract the answer and append it to this page.</li>
     <li>The source code for the JS part is in this page (do "View Source") and the PHP
part is <a href="ajax1.phps">ajax1.phps</a></li>
      <li>Other examples: see the <a href="./">directory</a></li>
    </ul>


    <hr>
    <address><a href="mailto:Daniel.Schneider@tecfa.unige.ch">Daniel K. Schneider</a></
address>
<!-- Created: Fri Apr 28 09:57:16 CEST 2006 -->
<!-- hhmts start -->
Last modified: Tue Oct 02 20:08:02 Paris, Madrid 2007
<!-- hhmts end -->
  </body>
```

```
</html>
```

Le fichier PHP (ajax1.php)

```
<?php
error_reporting(E_ALL);
header ("Content-type: application/xml");
echo '<?xml version="1.0" encoding="ISO-8859-1" ?>';

if (array_key_exists('user_pref_fruit', $_POST))
  { $user_pref  = $_POST['user_pref_fruit']; }
 else
   $user_pref="nothing";

echo "<answer>";
echo "Oh you like " . $user_pref . " !";
// echo "Oh you like " . $user_pref . " !" . " - Query String=" . $_SERVER["QUERY_STRING"];
echo "</answer>";

?>
```

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com
www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com
www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com
www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com
www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com
www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com
www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com