

get

Web Avancé

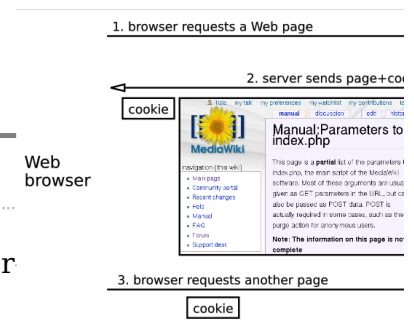
Luc Brun

Plan

- Les cookies
- Les sessions
- Ajax
- RSS

Les cookies

- Initialement utilisé sous unix dans les communications inter
- Permettent de stocker une information **chez le client**
 1. Le client demande une page
 2. Le serveur renvoi la page+ un (ou plusieurs)cookies
 3. Le client revoie automatiquement les cookies au serveur lors de connexions futures



Utilisation des cookies

- A quoi cela sert il ?
 1. Identifier une personne
 2. Implémenter la notion de panier
 3. personnaliser des applications (igoogole, Wikipedia,...)
 4. Calculer des profils d'utilisateurs (pub ciblés) ou calculer des statistiques sur des pages

Positionnement des cookies

- Demande du client au serveur

```
GET /~luc/WEB_AVANCE/s5-intro.html HTTP/1.1
Host: www.greyc.ensicaen.fr
```

- Réponse "classique" du serveur :

```
HTTP/1.1 200 OK
Content-type: text/html
```

```
<html>...</html>
```



Notez le saut de ligne ! Il marque la fin de l'en tête et le début du contenu du message.

Notez également que text/html peut être remplacé par n'importe quel type-mime.

Positionnement des cookies

- Positionner d'un cookie
- Demande du client au serveur

```
GET /~luc/WEB_AVANCE/s5-intro.html HTTP/1.1
Host: www.greyc.ensicaen.fr
```

- Réponse du serveur avec positionnement d'un cookie :

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name=value; expires=date; path=/; domain=www.greyc.ensicaen.fr
<html>...</html>
```

Options des cookies

```
Set-Cookie: name=value; expires=date; path=/; domain=www.greyc.ensicaen.fr
```

- Un cookie possède :
 1. Un **nom** (name), une **valeur** (value)
 2. Une date **d'expiration**(date) au delà de laquelle il sera effacé chez le client.
Une date dans le passé efface un cookie.
 3. Un **domaine**(domain) auquel il sera transmis
 4. Un **chemin** indiquant à partir d'où il pourra être lue.

Un cookie est identifié par le triplé nom/chemin/domaine => différents cookies de même nom.

Options des cookies (détails)

Set-Cookie: name=value; expires=date; path=/; domain=www.greyc.ensicaen.fr

- Par défaut la date d'expiration correspond à la fin de la session. Un cookie avec une date d'expiration est appelé **persistant**. Le format de la date est :

Wdy, DD-Mon-YYYY HH:MM:SS GMT

Exemple, Mercredi 20 août 2008 à 21h05:

Wed, 20-08-2008 21:05:00 GMT

- Par défaut le nom de domaine & chemin sont tirés de l'url. Par sécurité un cookie est refusé si le nom du serveur n'apparaît pas dans l'url.

Cookies et Shell : Un exemple

```
#!/bin/sh
expires=`date -d '+1 hour' +%a, %d-%b-%Y %T GMT`\
echo "Content-type: text-html"
echo "Set-Cookie: seen=yes;expires=$expires"
echo ""
echo "<html><head><title>Hello</title></head><body><h1>"
if echo $HTTP_COOKIE | grep -q seen
then
    echo "Je vous ai déjà vu quelque part"
else
    echo "Hello Etranger"
fi
echo "</h1></body></html>"
```

Lecture d'un cookie en schell

```
#!/bin/sh
get_val()
{
    name=$1
    if [ "$HTTP_COOKIE" = "" ]
    then
        echo
        return -1
    fi
    ifs=$IFS
    IFS=";"
    set $HTTP_COOKIE
    IFS=$ifs
    for egal in $*
    do
        var=`echo $egal | cut -d"=" -f1`
        if [ $var = $name ]
        then
            echo $egal | cut -d"=" -f2
            return 0
        fi
    done
    echo ""
    return -2
}
```

Cookies en PHP

- Positionné par la fonction **setcookie**
Syntaxe:

setcookie(name, value, expire,path,domain);

- Le positionnement du cookie doit apparaître dans l'entête donc **avant** toute balise html.

Cookies en PHP: Exemple

```
<?php
setcookie("seen", "yes", time()+3600);
?>

<html>
<head><title>Hello</title></head>
<body><h1>
<?
if(isset($_COOKIE["seen"]))
    echo "Je vous ai déjà vu quelque part";
else
    echo "Hello Etranger";
?>
</h1>
</body>
</html>
```


Cookies en Javascript

- Positionnement de la variable **document.cookie**

document.cookie = "name=value; expires=date; path=path; domain=domain";


- En lecture document.cookie contient une suite d'affectations séparées par des ;

var1=val1; var2=val2;...

-  Fonctionnement différent qu'en shell ou en PHP: Javascript est exécuté sur le navigateur !

Cookies en Javascript: Exemple

```
<script language="Javascript">
var seen=true;
if((document.cookie.length==0)|| (document.cookie.indexOf("seen")==-1))
{
    document.write("Hello Etranger!");
    seen=false;
}
date=new Date();
date.setTime(date.getTime()+60*60*1000);
document.cookie="seen=yes; expires="+date.toGMTString()+" "; path="/";
if(seen)
document.write("Je vous ai déjà vu quelque part..");
</script>
```

-  document.cookie est immédiatement modifié.

Lecture, écriture de Cookies en Javascript

```
function createCookie(name,value,days,hours) {
    if (days||hours) {
        var date = new Date();
        date.setTime(date.getTime()+(days*24*60*60*1000)+(hours*60*60*1000));
        var expires = "; expires="+date.toGMTString();
    }
    else var expires = "";
    document.cookie = name+"="+value+expires+"; path=/";
}

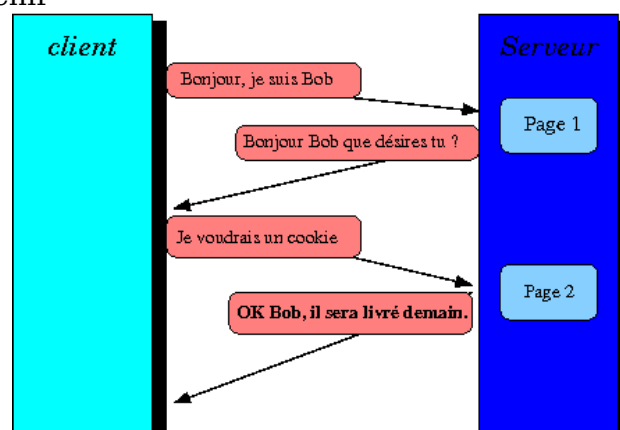
function readCookie(name) {
    var nameEQ = name + "=";
    var ca = document.cookie.split(';');
    for(var i=0;i < ca.length;i++) {
        var c = ca[i];
        while (c.charAt(0)==' ') c = c.substring(1,c.length);
        if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length,c.length);
    }
    return null;
}

function eraseCookie(name) {
    createCookie(name,"",-1);
}
```

- Source: <http://www.quirksmode.org/js/cookies.html>

Définition de données persistantes

- Ce que l'on veut obtenir



Méthode par champs caché

- Une façon de faire est d'utiliser les champs cachés des formulaires.
- L'utilisateur s'identifie par un formulaire
- L'application renvoie sa réponse avec un formulaire contenant un champ caché qui identifie l'utilisateur.

Les sessions PHP

- Plutôt que d'utiliser des formulaires avec champs cachés qui contiennent le nom, on utilise une variable codant une clé unique pour chaque client.
- Cette variable est transmise de page en page par un cookie ou en la plaçant dans l'url
- Chaque client qui se connecte sur une page du site envoie sa clé
- Le serveur PHP, retrouve les informations relatives à l'utilisateur à partir de sa clé.

Démarrage d'une session

```
bool session_start ( void )
```

- Crée une nouvelle session ou récupère les données d'une session précédente.
- On peut également donner un nom à la session par la fonction `session_name` (utile pour les personnes qui acceptent manuellement les cookies)
- La fonction renvoie vrai en cas de succès et faux sinon.



Si la clé est transmise par cookie, le `start_session` doit être positionné avant toute balise HTML

Positionnement des variables : \$_SESSION.

- Un tableau associatif contenant les variables du script courant.
- Exemple:

```
<?php
// page1.php
session_start();

echo 'Welcome to page #1';
$_SESSION['favcolor'] = 'green';
$_SESSION['time'] = time();
// Works if session cookie was accepted
echo '<br /><a href="page2.php">page 2</a>';
// Or maybe pass along the session id, if needed
echo '<br /><a href="page2.php?" . SID . ">page 2</a>';
?>
```

- SID est une constante correspondant au numéro de la session. Elle est passée dans l'url si les cookies ne sont pas acceptés.

Les fonctions isset, unset

isset et unset permettent respectivement de déterminer si une variable est positionné et de la supprimer. Dans le cadre des sessions :

isset(\$_SESSION['pizza']) :	permet de savoir si la variable pizza est positionné dans la session.
unset(\$_SESSION['pizza']) :	permet de supprimer pizza de la liste des variables de la session.
void session_unset (void) :	supprime toutes les variables de la session.

Fin de sessions

bool session_destroy(void)

- Détruit toutes les variables de la session mais pas la session même qui peut être récupéré par un nouveau session_start().
- Pour tuer également la session, il faut détruire l'id de celle-ci. Si l'id est passé à l'aide de cookies, cela peut être effectué en détruisant le cookie associé

Fin de sessions : Example

```
<?php
// Initialize the session.
// If you are using session_name("something"), don't
session_start();

// Unset all of the session variables.
$_SESSION = array();

// If it's desired to kill the session, also delete
// Note: This will destroy the session, and not just
if (isset($_COOKIE[session_name()]))
    setcookie(session_name(), '', time()-42000, '/')

// Finally, destroy the session.
session_destroy();
?>
```

AJAX

AJAX(Asynchronous Javascript and Xml) est une méthode permettant d'interroger un serveur http a partir d'un navigateur et du langage Javascript. Son avantage est de permettre une plus grande réactivité de l'interface par rapport au web classique.

Cette partie du cours est inspirée des cours de :

- Olivier Lezoray (cours XML)
- Cours du CERTA de Pierre Loisel et al.

un ciment de technologies existantes ?

AJAX est constitué d'une combinaison de technologies existantes. Un schéma classique d'utilisation est le suivant:

1. Un utilisateur exécute une action
2. l'action appelle une fonction Javascript qui envoie une requête à un serveur
3. le serveur traite la requête et renvoie une réponse au format texte ou XML
4. la réponse est récupérée par une fonction Javascript qui va modifier le document HTML en jouant sur les propriétés CSS ou sur le contenu HTML du site en utilisant DOM (voir + loin).

XmlHttpRequest

AJAX est basé sur la classe javascript *XMLHttpRequest*. A l'origine un objet ActiveX introduit par Microsoft dans IE5 puis repris par les autres navigateurs en tant qu'objet Javascript. Microsoft a suivi depuis IE7. Un objet XMLHttpRequest:

- ouvre une connexion HTTP
- permet de communiquer via HTTP en GET ou POST
- gère la réception des réponses du serveur en XML ou texte.

Création d'un objet XmlHttpRequest

- Avec un navigateur "moderne" il suffit de taper :

```
requeteHTTP=new XMLHttpRequest();
```

- Mais cela ne marche ni avec IE5 ni avec IE6. Pour ceux là il faut utiliser :

```
◦ requetteHttp=new ActiveXObject("Microsoft.XMLHTTP");
```

ou

```
◦ requetteHttp=new ActiveXObject("Msxml2.XMLHTTP");
```

D'autre part certains navigateurs exigent que le type de données utilisé par le serveur soit du text/xml. On peut forcer ce type par la fonction `overrideMimeType('text/xml')`. Une fonction résumant l'ensemble de ces contraintes est disponible [ici](#) (source P. Loisel).

Propriétés de XMLHttpRequest

readyState : voir les valeurs
onreadystatechange : nom de la fonction callback à exécuter lorsqu'il y a un changement d'état de l'objet
status : le code HTTP renvoyé par la requête
statusText : la chaîne correspondant au code HTTP
responseText : réponse au format texte
responseXML : réponse au format XML

Méthodes de XMLHttpRequest

open(String method , String url, asynFlag) : initialisation de la requête
method : GET ou POST
url : url à déclencher
asynFlag : synchrone ou asynchrone
overrideMimeType(String mimeType) : force le type MIME de la réponse
send(Variant body) : Envoi de la requête (null en get ou données en post)
setRequestHeader(String header, String value) : en-tête HTTP à envoyer

Un premier exemple (simple)

On utilise une méthode GET, en mode synchrone, avec une réponse de type text.

```
<script type="text/javascript" src="ajax.js">
</script>
<script type="text/javascript">
    function sendRequest(url)
    {
        var requeteHttp=getRequestHttp();
        if(requeteHttp==null)
        {
            alert("Impossible d'utiliser Ajax sur ce navigateur");
        }
        else
        {
            requeteHttp.open('GET',url,false);
            requeteHttp.send(null);
            if(requeteHttp.readyState==4) // requete achevé. Résultat transmis
            {
                if(requeteHttp.status==200)// fin correcte de la requete
                alert(requeteHttp.responseText); //affichage du message
            }
            else
            alert("Error :"+requeteHttp.status+",La requete ne s'est pas correctement execu
            }
        }
        return true;
    }
</script>
</head>

<body>
    <h1>Page 1 Ajax</h1>

    <a href="#" onclick="return sendRequest('page2.php?messg=coucou');"> requete</a>
```

Remarques et démonstrations

- La méthode GET est indiquée par le premier paramètre de `requeteHttp.open('GET',url,false);`
- En mode GET les paramètres sont passés dans l'URL
- Enfin le dernier champs à `false` indique que l'on est en mode synchrone (le client attend la réponse du serveur avant de continuer)
- L'utilisation du mode texte apparaît dans `alert(requeteHttp.responseText);`

Démonstration

Modification du contenu d'une balise.

- On peut modifier simplement le contenu d'une balise en utilisant sa propriété `innerHTML`. Pour cela on modifie le `alert` en un code comme :

```
document.getElementById("toto").innerHTML=response;
```

- Exemple pour l'heure sur le serveur:

Mise à jour de l'heure

Heure du serveur :

- Le code PHP se résume à :

```
header("Cache-Control: no-cache, must-revalidate");  
echo date('h:i:s');
```

Notez l'utilisation du header pour forcer le rechargement de la page par le navigateur.

Appels Asynchrones(1/2)

- Problème des requêtes synchrones: Le navigateur est figé en attendant la réponse.
- En mode asynchrone, le navigateur continue ses tâches et appelle une fonction de callback lorsque le résultat arrive.

```
var requeteHttp=getRequestHttp();  
if(requeteHttp==null)  
{  
    alert("Impossible d'utiliser Ajax sur ce navigateur")  
}  
else  
{  
    requeteHttp.open('GET',url,true);  
    requeteHttp.onreadystatechange=process_request();  
    requeteHttp.send(null);  
    ...  
}
```

Appels Asynchrones (2/2)

- Pb: Avec ce système la fonction `process_request()` est forcément sans paramètre: Pas moyen d'y passer `requeteHttp` et donc pas moyen de traiter la réponse.
- On utilise les fonction anonymes

```
function sendRequest(url)
{
    var requeteHttp=getRequestHttp();
    if(requeteHttp==null)
    {
        alert("Impossible d'utiliser Ajax sur ce navigateur");
    }
    else
    {
        requeteHttp.open('GET',url,true);
        requeteHttp.onreadystatechange=function (){ process_request(requeteHttp);
        requeteHttp.send(null);
    }
    return true;
}
```

Passage d'arguments en mode POST

- Il est nécessaire de paramétrer l'entête utilisé par la requête

```
requeteHttp.open('POST',url,true);
requeteHttp.setRequestHeader('Content-Type','application/x-www
```

- et de transmettre les valeurs:

```
requeteHttp.send(var1=val1&var2=val2&...)
```

- Un exemple de code

```
function postRequestWithParam(url,param)
{
    var requeteHttp=getRequestHttp();
    if(requeteHttp==null)
    {
        alert("Impossible d'utiliser Ajax sur ce navigateur");
    }
    else
    {
        requeteHttp.open('POST',url,true);
        requeteHttp.onreadystatechange=function (){ process_request(requeteHttp);
        requeteHttp.setRequestHeader('Content-Type','application/x-www-form-urlencoded');
        requeteHttp.send(param);
    }
    return true;
}
```

XML et DOM

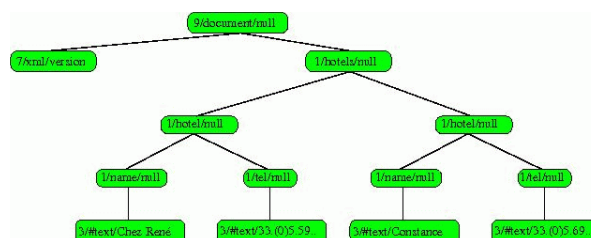
- On a jusqu'à présent utilisé des requêtes non structurées en mode texte (requeteHttp.responseText)
- Si on veut (par ex.) transmettre une liste avec des sous-listes. Il faudra convenir d'un protocole entre le serveur et le client (ex. item1:sousitem11,...sousitem1n;item2:sousitem21...).
- Le serveur codera sa réponse et
- le client devra développer une procédure de décodage pour mettre à jour la page.
- Mieux: Le serveur envoie sa requête au format XML et le client la décode grâce à DOM

XML(eXtensible Markup Language)


- Un langage composé de balises et sous-balises

```
<?xml version="1.0" encoding="utf-8"?>
<hotels>
<hotel prix="48">
<name>Chez René **</name>
<tel>33.(0)5.59.69.46.28.</tel>
</hotel>
<hotel prix="56">
<name>Hotel Constance </name>
<tel> 33.(0)5.69.29.19.36.</tel>
</hotel>
</hotels>
```

- Un document XML correspond à un Arbre (affichage : Type du noeud/nom/valeur)



DOM (Document Object Model)

- Le DOM permet de manipuler/explorer l'arbre d'un document XML.
-  Un fichier XHTML n'est qu'un document XML particulier. AJAX va donc:
 - Explorer l'arbre XML correspondant au résultat de la requête pour
 - modifier l'arbre XML codant la page HTML
- Les principaux types de noeuds

DOM: Les noeuds (propriétés)

Ceci n'est qu'un résumé de DOM d'avantage d'information peut être trouvé [ici](#)

Code	Type
1	Element
2	attribut (d'un noeud)
3	texte
9	Document

nom	Type	Modifiable	Description
nodeName	Chaîne	Non	Le nom de ce noeud, qui dépend de son type; voir la table ci-dessus.
nodeValue	Chaîne	Oui	La valeur de ce noeud, qui dépend de son type; voir la table ci-dessus.
nodeType	Entier	Non	Un code représentant le type d'objet sous-jacent, tel que défini ci-dessus.
parentNode	Node	Non	Le parent de ce noeud.
childNodes	NodeList	Non	La liste de noeuds constituée des enfants du noeud.
firstChild	Node	Non	Le premier enfant de ce noeud Si un tel noeud n'existe pas, la valeur null est retournée.
lastChild	Node	Non	Le dernier enfant de ce noeud Si un tel noeud n'existe pas, la valeur null est retournée.
previousSibling	Node	Non	Le noeud qui précède immédiatement le noeud courant. Si un tel noeud n'existe pas, la valeur null est retournée.
nextSibling	Node	Non	Le noeud qui suit immédiatement le noeud courant. Si un tel noeud n'existe pas, la valeur null est retournée.

DOM: Les noeuds (Méthodes)

Node insertBefore(in Node newChild, in Node refChild)

provoque(DOMException); : Insère le noeud newChild avant le noeud enfant refChild (si ce dernier existe). Si refChild est null, alors newChild est inséré à la fin de la liste des enfants. Retourne le noeud inséré.

Node replaceChild(in Node newChild, in Node oldChild)

provoque(DOMException); : Remplace le noeud enfant oldChild par newChild dans la liste des enfants, et retourne le noeud oldChild. Si le newChild est déjà dans l'arbre, il en est d'abord retiré. Retourne le noeud remplacé.

Node removeChild(in Node oldChild) provoque(DOMException) :

Retire le noeud enfant oldChild de la liste des enfants, et le retourne.

Node appendChild(in Node newChild) provoque(DOMException); :

Ajoute le noeud newChild à la fin de la liste des noeuds enfants du noeud courant. Si newChild est déjà dans l'arbre, il est d'abord retiré. Retourne le noeud ajouté.

boolean hasChildNodes(); : C'est une méthode pratique qui permet de savoir si un noeud a des enfants. Retourne true si le noeud a un enfant, false sinon.

Node cloneNode(in boolean deep); : Retourne un clone de ce noeud, c'est à dire, qu'elle est utilisée comme constructeur générique pour copier des noeuds. Le noeud cloné n'a pas de parent (c'est à dire que si on lui applique la méthode parentNode la valeur retournée est null.). Si le paramètre deep est true, la méthode clone récursivement le sous-arbre du noeud spécifié. Sinon elle ne clone que le noeud lui-même (et ses attributs, si c'est un noeud de type Element). Retourne le noeud dupliqué.

DOM: Les NodeList

- L'attribut *length* code nombre de noeuds dans la liste. Le domaine de validité du paramètre index va de 0 à length-1 inclus.
- Méthode: *item* Retourne l'item de la collection se trouvant à la position spécifiée par le paramètre index. Si index est supérieure ou égale au nombre de noeuds de la collection, la méthode retourne la valeur null.
- Une utilisation typique d'une NodeList est donc:

```
for(i=0;i<list.length;i++)  
{  
    n=list.item(i);  
    ...  
}
```


DOM: Les attributs

nom	Type	Modifiable	Description
name	Chaîne	Non	Retourne le nom de l'attribut courant.
specified	booléen	Non	Prend la valeur true si une valeur a été explicitement donnée à l'attribut courant dans le document original ; false sinon.
value	Chaîne	Oui	A l'utilisation, la valeur de l'attribut est retournée comme une chaîne de caractères. Les entités caractères et les entités générales de

DOM : Les éléments

- **Attribut:** tagName (non modifiable), correspond au nom de l'élément

DOMString getAttribute(in DOMString name); : Permet d'obtenir une valeur d'attribut à partir du nom de l'attribut. Retourne la valeur de l'attribut

void setAttribute(in DOMString name, in DOMString value) provoque (DOMException); : Ajoute un nouvel attribut. Si un attribut de même nom existe déjà au niveau de l'élément, sa valeur est changée par celle passée en paramètre. Ne retourne rien.

void removeAttribute(in DOMString name) provoque (DOMException); : Supprime un attribut spécifié par son nom. Si l'attribut supprimé a une valeur par défaut, il est alors immédiatement remplacé.

Attribut getNode(in DOMString name); : Permet d'obtenir un noeud de type Attr à partir de son nom. Retourne le noeud de type Attr dont le nom est celui spécifié ou null si il n'y a pas d'attribut de ce nom.

Attribut setAttributeNode(in nouvelAttribut) provoque (DOMException); : Rajoute un nouvel attribut. Si un attribut de ce nom est déjà présent dans l'élément, il est remplacé par le nouveau. Si le nouvel attribut newAttr remplace un attribut existant de même nom, le noeud attribut Attr précédemment existant est retourné, sinon la valeur null est retournée.

Attribut removeAttributeNode(in oldAttribut) provoque (DOMException); : Supprime l'attribut spécifié. Retourne le noeud qui a été supprimé.

NodeList getElementsByTagName(in DOMString name); :

Retourne une liste de noeuds (NodeList) de tous les éléments descendant ayant un nom donné, organisés selon l'ordre dans lequel ils pourraient apparaître dans un parcours préfixé de l'arbre des objets

⚠ fonction très utile.

DOM: Document

- **Attribut:** `documentElement` (non modifiable), correspond à l'élément racine du document

Element **createElement(in DOMString tagName)** **provoque (DOMException); :** Crée un élément du type spécifié. Notez que l'instance retournée implémente l'interface `Element`, donc les attributs peuvent être spécifiés directement sur l'objet retourné. Retourne le nouvel objet.

Text **createTextNode(in DOMString data); :** Crée un noeud `Text` contenant la chaîne de caractères spécifiée. Retourne le nouvel objet `Text`.

Attr **createAttribute(in DOMString Name)** **provoque (DOMException); :** Crée un objet `Attr` d'un nom donné. Notez que l'instance de l'objet `Attr` peut ensuite être rattachée à un `Element` en utilisant la méthode `setAttribute`.

NodeList **getElementsByTagName(in DOMString tagName); :** Retourne un objet `NodeList` (une liste de noeuds) de tous les objets `Elements` ayant un nom de balise donné, laquelle est ordonnée selon l'ordre de lecture prédéfini de l'arbre du Document.

DOM: NameNodeMap

Représente une collection de noeuds non ordonnés. Son unique attribut `length` représente le nombre d'éléments.

Node **getNamedItem(in DOMString name); :** Permet d'obtenir un noeud à partir de son nom.

Node **setNamedItem(in noeud arg)** **provoque(DOMException); :** Ajoute un noeud en utilisant son attribut `nodeName`. Notez que plusieurs noeuds d'un même type ne peuvent pas être stockés puisque leurs noms rentreraient alors en collision

Node **removeNamedItem(in DOMString name)** **provoque (DOMException); :** Retire un noeud spécifié par son nom. Si le noeud retiré est de type `Attr` ayant une valeur par défaut, il est immédiatement remplacé. Retourne le noeud retiré de la table des noeuds nommés ou la valeur `null` si aucun noeud de ce nom n'existe dans cette table.

Node **item(in unsigned long index); :** Retourne l'élément à la position `index` du `NameNodeMap`. Si la valeur de `index` est supérieure ou égale au nombre de noeuds de la table, la valeur retournée est `null`.

Un premier exemple (simple)

```
...requeteHttp.onreadystatechange=function(){afficher_alert(requeteHttp)};...
et
function afficher_alert(requeteHttp)
{
    var hotels=requeteHttp.responseXML.getElementsByTagName("hotel");
    for(i=0;i<hotels.length;i++)
    {
        var hotel=hotels.item(i);
        var prix =hotel.getAttribute("prix");
        //      var name=hotel.getElementsByTagName("name").item(0).firstChild.nc
        var name=hotel.childNodes.item(1).firstChild.nodeValue;
        var tel=hotel.getElementsByTagName("tel").item(0).firstChild.nodeValue;

        alert("prix="+prix+"name="+name+"tel="+tel);
    }
}
```

- [exemple d'exécution](#)

Un second exemple (pas très propre)

```
function afficher_alert2(requeteHttp)
{
    var hotels=requeteHttp.responseXML.getElementsByTagName("hotel");
    var obj=document.getElementById("answer1");
    obj.innerHTML="

    ";
    for(i=0;i<hotels.length;i++)
    {
        var hotel=hotels.item(i);
        var prix =hotel.getAttribute("prix");
        var name=hotel.getElementsByTagName("name").item(0).firstChild.nodeV
        var tel=hotel.getElementsByTagName("tel").item(0).firstChild.nodeVal
        obj.innerHTML+="
```

- [exemple d'exécution](#)

Un dernier exemple

```
function afficher_table(requeteHttp)
{
    var titles=Array("Prix","Nom","Tel");
    var values=Array();
    var hotels=requeteHttp.responseXML.getElementsByTagName("hotel");
    var obj=document.getElementById("answer");
    var table=document.createElement("table");
    var oldTables=obj.getElementsByTagName("table");
    if(oldTables.length==1)
        obj.replaceChild(oldTables.item(0),table);

    table.setAttribute("border","1");
    var tr=document.createElement("tr");
    var td;
    for(i=0;i<3;i++)
    {
        td=document.createElement("td");
        td.appendChild(document.createTextNode(titles[i]));
        tr.appendChild(td);
    }
    table.appendChild(tr);
    for(i=0;i<hotels.length;i++)
    {
        var hotel=hotels.item(i);
        values[0] = hotel.getAttributeNode("prix").value;
        values[1] = hotel.getElementsByTagName("name").item(0).firstChild.nodeValue;
        values[2] = hotel.getElementsByTagName("tel").item(0).firstChild.nodeValue;
        tr=document.createElement("tr");

        for(j=0;j<3;j++)
        {
            td=document.createElement("td");
            td.appendChild(document.createTextNode(values[j]));
            tr.appendChild(td);
        }
        table.appendChild(tr);
    }
    obj.appendChild(table);
}
```

Un dernier exemple (Exemple)

- Soit le code HTML

```
<li>    <a href="#" onclick="javascript:post_sendRequest('hotels.xml',afficher_t
</ul>
    <div id="answer"></div>
```

- DOM

Les flux RSS

• RSS: Rich Site Syndication.

(ou Really Simple Syndication)

- Un flux RSS est un document XML présentant une série de nouvelles (généralement brèves) datées.
- Le principal avantage des flux RSS est qu'ils peuvent être combinés avec des lecteurs de flux présentant les dernières nouvelles de plusieurs sites simultanément.
- Les principaux lecteurs sont
 - Bloglines [www.bloglines.com]
 - Google Reader [www.google.com/reader]
 - Feedbucket [www.feedbucket.com]
- On peut également utiliser netvibes ou igoogole (installés en home page)

Définition d'un flux RSS

- Un flux rss débute obligatoirement par :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
```

afin de préciser la version de rss utilisée.

- suit ensuite des méta données, dont 3 données obligatoires :

- **<title>** : Définit le titre du flux ;
- **<description>** : Décrit succinctement le flux ;
- **<link>** : Définit l'URL du site correspondant au flux.

et des données optionnelles comme :

- **<pubDate>** : Définit la date de publication du flux ;
- **<lastBuildDate>** : Définit la dernière date de modification du flux ;
- **<image>** : Permet d'insérer une image dans le flux ;
- **<language>** : Définit la langue du flux.

suit le contenu du flux sous forme d'items.

Contenu d'un flux RSS

- Défini par une liste de balises **<item>**
- Les principales balises sont
 - **<title>** : Définit le titre de l'actualité ;
 - **<link>** : Définit l'URL du flux correspondant à l'actualité ;
 - **<pubDate>** : Définit la date de l'actualité ;
 - **<description>** : Définit une description succincte de l'actualité ;
 - **<guid>** : Définit de manière unique l'actualité.

Selon la DTD RSS 2.0, il doit y avoir au moins un **<title>** ou une **<description>** dans un item et le reste des balises est optionnel.

Autres balises: **<author>** : mail de l'auteur, **<category>** : Associe l'item à une catégorie ; **<comments>** : URL d'une page de commentaire en rapport avec l'item.

DTD d'un RSS et Exemple

DTD : Voir le code [suivant](#)

Exemple : Voir le code [suivant](#)

Génération de Flux RSS: Un exemple (1/3)

```

header('Content-type: application/atom+xml');
$dbconn=connexion();
function display_news($query,$i)
{
    echo "<item>\n";

    echo "<title>".pg_fetch_result($query,$i,2)."</title>\n";
    echo "<description>".pg_fetch_result($query,$i,3)."</description>\n";
    echo "<pubDate>".pg_fetch_result($query,$i,1)."</pubDate>\n";
    echo "<link>".pg_fetch_result($query,$i,4)."</link>\n";

    echo "</item>\n";
}
echo '<?xml version="1.0" encoding="utf-8"?>';
?>

<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
<channel>
<title>IAPR TC 15 RSS</title>
<description>IAPR TC15 news </description>
<image>
    <url>http://www.greyc.ensicaen.fr/iapr-tc15/images/gbr_logo.jpg</url>
    <link>http://www.greyc.ensicaen.fr/iapr-tc15/</link>
</image>
<?
    $requete="select date of last update"; $query=pg_query($dbconn,$requete);

echo "<lastBuildDate>".pg_fetch_result($query,0,0)."</lastBuildDate>\n";
pg_free_result($query);
?>

<link>http://www.greyc.ensicaen.fr/iapr-tc15/rss.php</link>
<?
$requete="select features of news order by date desc"; $query=pg_query($dbconn)
if(!$query){ echo "Query error :".$requete; exit;}
for($i=0;$i<pg_num_rows($query);$i++)
    display_news($query,$i);
pg_free_result($query);pg_close($dbconn);
</channel></rss>

```

Génération de Flux RSS: Un exemple (2/3)

```
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
<channel>
<title>IAPR TC 15 RSS</title>
<description>IAPR TC15 news </description>
<image>
    <url>http://www.greyc.ensicaen.fr/iapr-tc15/images/gbr_logo.jpg</url>
    <link>http://www.greyc.ensicaen.fr/iapr-tc15/</link>
</image>
<?
$requete="select date of last update";
$query=pg_query($dbconn,$requete);

echo "<lastBuildDate>".pg_fetch_result($query,0,0)."</lastBuildDate>";
pg_free_result($query);
?>

<link>http://www.greyc.ensicaen.fr/iapr-tc15/rss.php</link>
```

Génération de Flux RSS: Un exemple (3/3)

```
<?
$requete="select features of news order by date desc";
$query=pg_query($dbconn,$requete);
if(!$query)
{
    echo "Query error :".$requete;
    exit;
}
for($i=0;$i<pg_num_rows($query);$i++)
    display_news($query,$i);
pg_free_result($query);
pg_close($dbconn);
?>
</channel>
</rss>
```

That's all folks.