

AJAX



P.Mathieu

LP DA2I Lille
<http://www.iut-a.univ-lille.fr>
prenom.nom@univ-lille.fr

17 décembre 2019

1 Javascript

2 XML

3 AJAX

- JavaScript, langage interprété à l'intérieur de HTML, permettant de créer des objets, des fonctions et réagir aux événements.
- Il s'exécute sur le navigateur client
- Langage non typé, case sensitive (`onclick` \neq `onClick`)
- Le code JavaScript peut être mis n'importe où dans la page, entre balises `<script>` et `</script>`
- Tout ce qui est dans le `head` du code HTML est exécuté au chargement de la page (avant son affichage).
On y place en général les fonctions
- Une page HTML est lue linéairement par le navigateur, le code JavaScript est exécuté quand le navigateur accèdera à cette partie.

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Titre de la page</title>
<script language="javascript" type="text/javascript">
    aff1();
    function aff1(){alert("Alerte 1 : avant l'affichage");}
    function aff2(){alert("Alerte 2 : durant l'affichage");}
    function aff3(){alert("Alerte 3 : lors du click");}
</script>
</head>
<body>
<h1 id="message">Hello world</h1>
<script>aff2()</script>
<form>
    <input type="button" value="cliquer" onclick="aff3()" />
</form>
</body>
</html>
```

Différents évènements :

onchange . on modifie un champ (si rien n'est changé, il ne se déclenche pas)

onfocus . on rentre dans un champ

onblur .on sort du champs

onclick . on appuie sur un bouton

onload . la page est chargée

onkeypress . une touche est frappée au clavier

onsubmit . à la soumission du formulaire, juste avant envoi

setTimeout(fonc,delai) . Déclenche une minuterie et appelle le code javascript `fonc` dans `delai` millisecondes.

onsubmit sur un formulaire autorise l'envoi si la fonction renvoie true

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Titre de la page</title>
<script language="javascript" type="text/javascript">
function verif(val)
{
    if (val.length == 0 || isNaN(val))
    {
        alert("un entier est obligatoire !");
        return false;
    }
    else return true;
}
</script>
</head>
<body>
<form method="get" action="" onsubmit="return verif(nombre.value)" >
Saisir une chaine :
<input type="text" name="chaine" size="10" onchange="chaine.value=chaine.value.toUpperCase()" />
Saisir un nombre :
<input type="text" name="nombre" size="10" onchange="verif(nombre.value)" />
<input type="submit" value="Envoyer" />
</form>
</body>
```

Nombreuses bibliothèques prêtes à l'emploi

- JQuery
- prototype
- ExtJ (ex YUI)
- Dojo
- Mootool
- ...

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Titre de la page</title>
<script type="text/javascript" src="../../scripts/jquery-1.4.2.js" />
</head>
<body>
....
</body>
```

1 Javascript

2 XML

3 AJAX

- eXtensible Markup Language
- Méta-langage de description à balises, standardisé en 1996 par le W3C
- Permet d'organiser des données sous forme hiérarchique dans un format texte (donc ouvert)
- Chaque élément débute par une balise de début `<nom>` et se termine par une balise de fin `</nom>`
- les noms de balises sont entièrement libres
- un élément racine unique est obligatoire
- Un élément peut contenir une suite d'attributs
- un élément vide peut se noter `<nom/>`

```
<?xml version="1.0" encoding="UTF-8"?>
<agenda>
  <personne saisie="17/02/2003" dernieremodif="24/12/2010">
    <nom>Durand</nom>
    <prenom>paul</prenom>
    <age>12</age>
  </personne>
  <!-- ***** commentaire ***** -->
  <personne saisie="17/05/2009" dernieremodif="23/11/2010">
    <nom>Dupond</nom>
    <prenom>louis</prenom>
    <adresse>
      <numero>17</numero>
      <rue>rue Faidherbe</rue>
      <codepostal>59000</codepostal>
      <ville>Lille</ville>
    </adresse>
  </personne>
</agenda>
```

Attribut ou élément ?

- l'ordre des attributs dans une balise n'est pas imposé contrairement aux éléments
- la description d'un attribut est plus concise qu'un élément
- un élément peut décrire une donnée sous une forme hiérarchique
- un élément peut être répété plusieurs fois

Un CSV peut être facilement codé en XML

par exemple dump d'une table `personne (nom, prenom, age)`

```
nom;prenom;age  
durand;paul;22  
dupond;pierre;25  
lefebvre;jean;24
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<personne>  
  <ligne  nom="durand"  prenom="paul"  age="22"  />  
  <ligne  nom="dupond"  prenom="pierre" age="25"  />  
  <ligne  nom="lefebvre" prenom="jean"  age="24"  />  
</personne>
```

... mais pas l'inverse !

- CSV et XML sont tous deux en format texte (donc ouverts)
- CSV ne nécessite pas de balise, donc bien plus léger
- XML contient une référence à l'encodage et à la grammaire
- XML peut coder des structures complexes (éléments imbriqués, éléments parfois manquants, éléments répétés, ...)
- XML peut contenir des commentaires ()
- en XML les données textuelles peuvent tenir sur plusieurs lignes

On utilise CSV pour de grandes quantités de petites données non imbriquées (en râteau) ; on utilise XML pour l'échange inter-applications de structures complexes.

La DTD décrit la structure (grammaire) du document XML. Permet au receveur d'un fichier XML d'en vérifier la validité

La DTD n'est pas obligatoire. Si elle existe, elle peut :

- être intégrée directement dans le document xml
- être liée via un chemin relatif (balise SYSTEM)
- être liée via une URL absolue (balise PUBLIC)

De nombreuses DTD existent : MathML (formules mathématiques), SVG (géométrie 2D), OFX (infos financières), XSLT, XHTML

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Titre de la page</title>
</head>
<body>
<h1 id="message">Hello world</h1>
</body>
</html>
```

Exemple de DTD

```
<!ELEMENT agenda (personne)*>

<!ELEMENT personne (nom,prenom,age?, adresse?) >
<!ATTLIST personne saisie CDATA #REQUIRED >
<!ATTLIST personne dernieremodif CDATA #REQUIRED >

<!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT age (#PCDATA) >

<!ELEMENT adresse (numero|rue|codepostal|ville) >
<!ELEMENT numero (#PCDATA) >
<!ELEMENT rue (#PCDATA) >
<!ELEMENT codepostal (#PCDATA) >
<!ELEMENT ville (#PCDATA) >
```


- Comment vérifier qu'un fichier est bien formé ?
... en le chargeant dans un navigateur, ou par programme
- Comment vérifier qu'un fichier est valide ?
Comment parcourir les données ?
Le JDK fournit un package JAXP (Java API for XML Processing),
réunion de deux API
 - SAX** Simple API for XML. Programmation événementielle
avec procédure déclenchées au fur et à mesure de la
lecture des balises.
 - DOM** Document Object Model. Programmation
navigationnelle après le chargement complet du
document.

```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class TestXML
{
    public static void main (String args[]) throws Exception
    {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setValidating(true);
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.parse(args[0]);
    }
}
```

..... suite du précédent

```
Element root = doc.getDocumentElement();

Element pers = (Element) root.getElementsByTagName("personnes").item(0);
NodeList lperss = pers.getChildNodes();
for (int i=0; i<lperss.getLength(); i++) {
    if (lperss.item(i).getNodeType() == Node.ELEMENT_NODE) {
        Element pers = (Element) lperss.item(i);
        System.out.print(pers.getAttributeNode("nom").getNodeValue()+";");
        System.out.print(pers.getAttributeNode("prenom").getNodeValue()+";");
        System.out.println(pers.getAttributeNode("age").getNodeValue());
    }
}
}
```

XML

Nombreux outils

JAXP bibliothèque standard java intégrée dans le JDK pour analyser ou valider les documents XML

SimpleXML analyseur fourni avec PHP 5

XERCES API pour l'analyse de documents XML

JDOM //

DOM4J //

CASTOR //

XStream //

XALAN API Apache pour le traitement XSLT

1 Javascript

2 XML

3 **AJAX**

Asynchronous Javascript and Xml

Terme inventé en 2005 pour désigner une techno ancienne de 1999.

Application classique : Chaque click sur un bouton entraine un rechargement complet de la page.

Principe AJAX : Accéder au serveur web en Javascript pour mettre à jour uniquement de petites parties de la page WEB en cours.

AJAX

Avantages de l'approche AJAX

- Optimiser le trafic en ne ramenant que les données nécessaires au rafraichissement de la page
(20 à 80% d'une page web ne sert qu'à la présentation !)
- Permet d'apporter une aide à l'utilisateur en complétant pas par pas les champs des formulaires de saisie (listes, tris etc ..)
- Permet une approche "une seule page pour l'application"
(GoogleMaps, NetVibes, FlickrR, GMail, ...)

The screenshot shows a web browser window with the address bar displaying `www.netvibes.com/privatepage/1#General`. The page title is "Général". The dashboard contains several widgets:

- BFM TV**: A news widget with categories like "Politique", "Société", "International", and "Divertissement". It features a "BFM TV NEWS 24/7" logo and several news items with photos and headlines.
- Lille, France**: A weather widget showing the date "9 Décembre Mardi 4°C" and a 3-day forecast for Wednesday (6°C), Thursday (7°C), and Friday (3°C).
- Ebay**: A widget with a message about authorization and a "Widget Lien" section for adding a website.
- E-mail Wizard**: A widget titled "Vérifiez votre email depuis votre dashboard" with a grid of email providers (GMail, Outlook, Yahoo! Mail, AOL, Mac, Orange, Free, SFR, Numéricable) and an "Autres (POP)" option.
- Frenchweb.fr**: A widget with a logo and a section titled "A la une" containing news about CNIL and Twitter.
- Courrier International**: A news widget with a "Courrier International" logo and sections for "Monde", "France", "Cartoons", and "Insolites". It includes headlines about the 2014 election and corruption in Algeria.
- Twitter**: A widget with a Twitter logo and a message to connect to a Twitter account.
- Facebook**: A widget with a Facebook logo and a message that the service is temporarily unavailable.
- L'Obs - Actualités**: A widget with a logo and a section titled "Le nouvel Observateur" with headlines about Platin and Guantanamo.
- Les journaux de la rédaction**: A widget listing recent publications from the editorial team.
- To Do List**: A simple task list widget.

- Les requêtes sont envoyées via un objet Javascript (et pas un formulaire !) au serveur sur événements JavaScript
- Envoi et réception sont dissociés : Asynchrone

Quand JavaScript envoie une requête au serveur, on peut continuer à remplir des formulaires ou à cliquer sur des boutons :
On n'attend jamais !

- A la réception de la réponse, le code JavaScript de mise à jour (callback) est automatiquement appelé par le navigateur.
- L'objet clé : `XMLHttpRequest`

Création d'une application AJAX

3 étapes fondamentales :

- 1 Créer l'objet Javascript permettant de communiquer avec le serveur web :

```
req = new XMLHttpRequest();
```

- 2 Sur évènement, exécuter la requete qui demande au serveur web les infos à réactualiser :

```
function appel() {  
    req.open("GET", url, true);  
    req.onreadystatechange = callback;  
    req.send(null);  
}
```

- 3 Actualiser la partie de la page web nécessaire :

```
function callback() { ... }
```

- l'objet `XMLHttpRequest` a un code d'état (1 à 4). L'état 4 indique que le serveur a terminé son travail. La fonction est appelée 4 fois !
- le **status** indique si tout s'est bien passé. 200 :ok 404 :pb d'url
- la reponse est fournie dans `req.responseText`

```
function callback()  
{  
    if (req.readyState ==4)  
        if (req.status == 200)  
        {  
            var rep = req.responseText;  
            /* mettre à jour la page ici ! */  
        }  
        else alert("erreur : "+req.status);  
}
```

Trois moyens de mise à jour

- 1 Valeur des champs **input**
utiliser directement la propriété **value** facile !

```
document.getElementById("nom").value="nouveau texte"
```

- 2 Valeur des zones identifiées par `span` avec `id`
utiliser la propriété `innerHTML`, propriété de tout élément HTML
qui désigne le contenu qui se trouve entre sa balise entrante et sa
balise fermante.

```
document.getElementById("x").innerHTML = "nouveau"
```

- 3 Autre mise à jour
Parcourir l'arbre **DOM** ... beaucoup plus dur ;-)

Exemple de modification d'un élément `input`

```
<script language="javascript" type="text/javascript" >
function incr()
{
    var i = document.getElementById("num");
    i.value = parseInt(i.value)+1;
    // ou un accès à requete
}
</script>

<form>
<input type="text" name="num" id="num" value="1"/>
<p />
<input type="button" value="incrémenter" onclick="incr();" />
</form>
```

Exemple de modification d'un élément identifié

```
<html>
<script>
function modifier()
{
    var x = document.getElementById("x");
    x.innerHTML = "valeur modifiée";
}
</script>

<h1>Page de test</h1>
La phrase contenant <span id="x">l'élément à changer</span>

<input type="button" onclick=modifier() />
</html>
```

innerHTML permet d'affecter, la valeur
innerText permet de lire la valeur

Quelques conseils ...

- préférer XHTML ... sinon c'est le navigateur qui devine !
- utiliser des DIV et des SPAN pour nommer les zones
- id permet d'identifier une zone unique
- class permet d'identifier un ensemble.
- Bien nommer les zones à modifier évite le parcours de l'arbre DOM

Malheureusement `req = new XMLHttpRequest();` ne fonctionne pas chez microsoft

```
var req=null;
creerRequete();

function creerRequete() {
  try { req = new XMLHttpRequest(); }
  catch (essaiMicrosoft)
  {
    try { req = new ActiveXObject(`Msxml2.XMLHTTP`); }
    catch (autremicrosoft)
    {
      try { req = new ActiveXObject(`Microsoft.XMLHTTP`); }
      catch (echec) { req=null; }
    }
  }
  if (req==null)
    alert(`erreur de creation de XMLHttpRequest`);
}
```

à mettre dans le **head** ... pour un test immédiat !

Le passage de paramètres

Requête Get . Quand pas de params ou params peu sensibles.

```
var url="chercherClient.jsp?nom="+nom;  
req.open("GET",url,true);  
req.onreadystatechange = callback;  
req.send(null);
```

Requête Post . Permet d'envoyer des form, de l'Xml, une image..

```
var url="servlet/chercherClient";  
req.open("POST",url,true);  
req.onreadystatechange = callback;  
req.setRequestHeader("Content-Type","application/xml");  
req.send("nom="+nom);
```

Pas de parenthèses à la fonction ! Il est néanmoins possible de mettre une fonction anonyme qui appelle une fonction avec paramètres.

Tout englober dans un objet

```
function AJAXInteraction(url, callback) {

    var req = init();
    req.onreadystatechange = processRequest;

    function init() {
        if (window.XMLHttpRequest) {
            return new XMLHttpRequest();
        } else if (window.ActiveXObject) {
            return new ActiveXObject("Microsoft.XMLHTTP");
        }
    }

    function processRequest () {
        if (req.readyState == 4) {
            if (req.status == 200) {
                if (callback) callback(req.responseXML);
            }
        }
    }

    this.doGet = function() {
        req.open("GET", url, true);
        req.send(null);
    }

    this.doPost = function(body) {
        req.open("POST", url, true);
        req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        req.send(body);
    }
}
```

AJAX

... et pour l'appel

```
function makeRequest() {  
    var ai = new AJAXInteraction("url",  
                                function() { alert("affichage reponse"); });  
    ai.doGet();  
}
```

```
<input type="button" onclick="makeRequest()" value="Go!"/>
```

C'est exactement comme cela que fonctionnent les librairies Javascript genre JQuery, Prototype, Mojo etc ...

AJAX

Le modèle DOM

```
<html>
  <head>
    <title>Titre de la page</title>
  </head>
  <body>
    <div class="logo">
      
    </div>
    <div class="corps">
      <h1>premier titre</h1>
      voici un texte avec un
      <span id="m">mot</span>
      mis en valeur
    </div>
  </body>
</html>
```

AJAX

Parcourir l'arbre DOM : L'objet document

Se placer dans l'arbre

```
document.documentElement  
document.getElementById("m");  
document.getElementsByTagName("div");
```

parcourir l'arbre

```
noeud.parentNode  
noeud.firstChild  
noeud.lastChild  
noeud.childNodes
```

Manipuler les noeuds

```
document.createElement("img");  
document.createTextNode("taratata");  
noeud.appendChild()  
noeud.replaceChild()
```

Accès aux valeurs

```
champ input : value  
noeud elmt : nodeName  
noeud texte : nodeValue
```

AJAX

Exemple de modification via un parcours de l'arbre DOM

```
<script language="javascript" type="text/javascript">

// accéder à l'elmt
var monElmt = document.getElementById("m");
var valeur = monElmt.childNodes[0].nodeValue;
// ou monElmt.firstChild.nodeValue;

// remplacer l'elmt
var nouv = document.createTextNode("lapin");
monElmt.removeChild(monElmt.childNodes[0]);
monElmt.appendChild(nouv);

</script>
```

AJAX

Fonctions utilitaires classiques

```
// supprime tous les noeuds enfants de l'el  
function effacerTexte(el)  
{  
  if (el!=null)  
    if (el.childNodes)  
      for (var i=0; i<el.childNodes.length;  
        {  
          var noeudFils= el.childNodes[i];  
          el.removeChild(noeudFils);  
        }  
    }  
}
```

```
// remplace tout le texte d'un elmt par le  
function remplacerTexte(el,texte)  
{  
  if (el!=null)  
  {  
    effacerTexte(el);  
    var nouveauNoeud=document.createTextNode(texte);  
    el.appendChild(nouveauNoeud);  
  }  
}
```

```
// renvoie de texte de l'elmt demandé  
function getTexte(el)  
{  
  var texte="";  
  if (el!=null)  
    if (el.childNodes)  
      for (var i=0; i<el.childNodes.length; i++)  
      {  
        var noeudFils = el.childNodes[i];  
        if (noeudFils.nodeValue != null)  
          texte = texte + noeudFils.nodeValue;  
      }  
  return texte;  
}
```

AJAX

DOM et XML

... et si le serveur renvoie plusieurs données ?

Structuration des données en XML

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<liste>
  <paris>12000</paris>
  <lyon>5000</lyon>
  <lens>9000</lens>
</liste>
```

Envoi du serveur

```
req.setRequestHeader("Content-Type", "text/xml");
```

Récupération en Javascript

```
var docXml = req.responseXML;
var elmts = docXML.getElementsByTagName("lyon");
var lyon = elmts[0].firstChild.nodeValue;
```


- AJAX permet de mettre à jour de manière asynchrone des parties de la page.
- AJAX a aussi des inconvénients : il augmente souvent le trafic, il pose des pb d'accessibilité et de portabilité. Ne l'utiliser qu'à bon escient !
- XML est assez lourd. Sauf pour des structures complexes, un format csv sera en général bien suffisant. D'autres alternatives comme JSON existent.
- De nombreuses bibliothèques AJAX existent pour faciliter la vie du développeur (ajaxpatterns.org). Toujours veiller à comprendre ce qui est fourni !
- Voir aussi .. JQuery, YUI, Goggle Web Toolkit