Contenu

Flux RSS !

Je ne sais pas si cet acronyme vous est familier, la version qui nous intéresse signifie
Really Simple Syndication

Je vais essayer de vous traduire cela en français, Syndication, mot anglais mais ça ne vous a pas échappé, désigne la pratique consistant à « vendre » le droit de reproduire un contenu ou de diffuser un programme à plusieurs diffuseurs. Le système de syndication a été créé par la presse écrite américaine, les « syndicates » vendant leur production (bandes dessinées, chroniques etc) à plusieurs journaux locaux.
Par extension la syndication de contenu Web, car c'est de cela dont nous allons parler consiste à rendre une partie d'un site accessible depuis d'autres sites. La météo, l'horoscope, les news ce ne sont pas les sujets qui manquent et qui par essence même sont différents à chaque fois que nous allons nous connecter pourtant sur la même page. Comprenez bien cela (et vous aurez tout compris au RSS) pour un même lien de connexion, je ramène des informations variables mais pour avoir un semblant d'universalité, de compatibilité entre tous les systèmes susceptibles de lire du flux RSS il va falloir établir des règles, mettre un peu d'ordre. Vous vous souvenez de ma commode et de ses tiroirs ? On va retrouvez le même principe sauf que les noms vont changer
La commode s'appellera « balise ». Les tiroirs s'appelleront « métadonnées »
Et comme toute ces données sont de longueur variable on les bornera il y aura donc une balise au début de l'information qu'elle représente et une balise de fin
Même chose pour les métadonnées.

Prenons de suite un exemple cela sera tout de suite plus clair (au passage vous allez voir que nous retrouvons le contenu de nos fichiers xml (descriptor et screen) qui sont sur le même principe.

Un flux RSS commencera toujours par cette balise (appairée avec la balise de fin)

```
<rss>
        Ici du contenu
</rss>
```

Le / signifie fin
La balise suivante est obligatoirement une balise « channel »
Donc

```
<rss>
    <channel>
    </channel>
</rss>
```

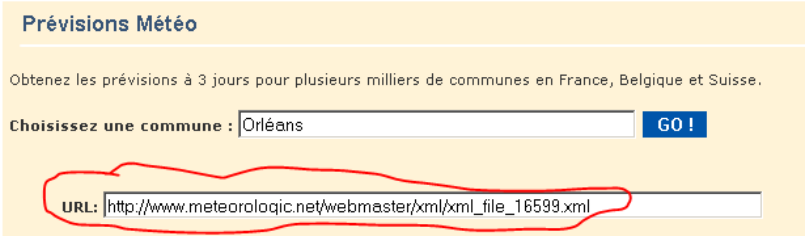
Etc nous n'allons pas passer tous les balises en revues, je vous invite à lire tout cela sur ce lien Wikipédia par exemple
http://fr.wikipedia.org/wiki/RSS_(format)
Ce que nous retiendrons c'est qu'un flux RSS peut contenir plusieurs « informations » sa balise est <item>
Le lien lorsqu'il existe <link>
La date de publication, ça peut servir pour ne pas lire l'horoscope d'hier ☺

Mais pourquoi faire compliqué lorsqu'on peut faire simple ? Restons fidèle à notre devise (enfin j'espère que vous l'avez fait votre aussi)

Imaginons qu'on veuille lire du flux meteo, je sais il existe déjà quelques applications mais on veut plus je ne sais pas moi, pour les 3 jours qui viennent, pour la semaine
Nous allons donc sur notre moteur de recherche favori et nous tapons « meteo flux rss »
Et nous regardons les résultats
J'ai flashé sur celui-ci

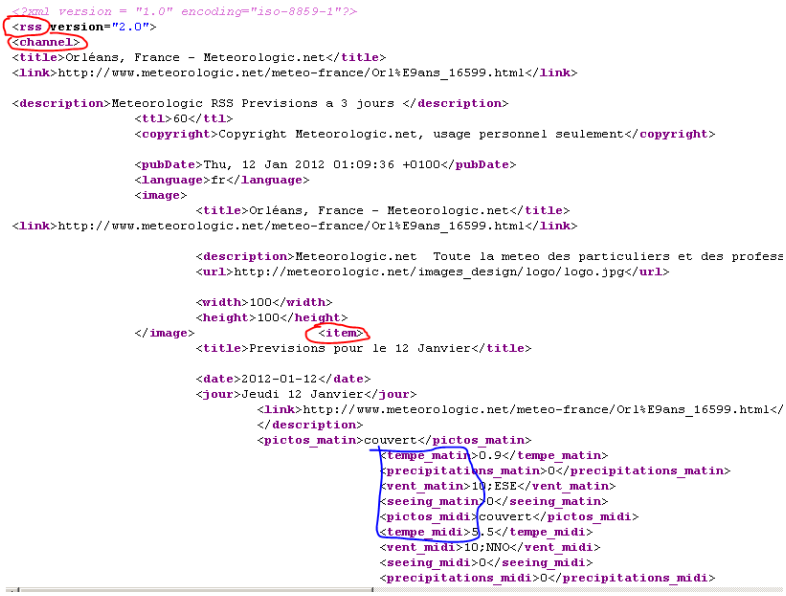
Flux météo RSS/XML - Meteorologic **RSS**
www.**meteo**rologic.net/zone-**rss**.php
Meteorologic RSS, **flux RSS** et XML de **météo**. Bienvenue sur Meteorologic RSS, vous trouverez ci-dessous des flux d'observations pour plusieurs milliers de ...

Je m'y rends de ce pas
Je renseigne la ville qui m'intéresse je fais go, valide dans la liste en dessous et j'obtiens une url



C'est cette url que je traiterais dans mon application pour l'instant je la copie dans ma barre d'adresse et j'arrive sur cette page

C’est bien joli tout ça mais je ne vois toujours pas mes balises !
Clic droit sur cette page et « code source de la page » (sur firefox)
Et la oui je suis heureux, je vois enfin mes balises

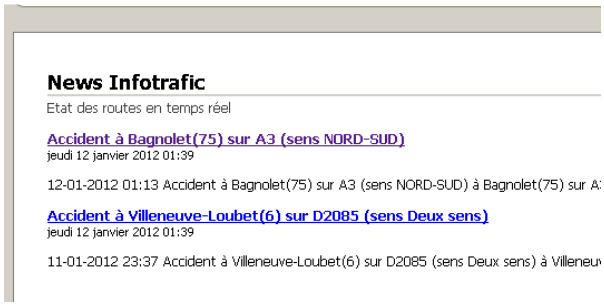


J’ai RSS, Channel, Item (3 fois pour les 3 jours présents sur la page)
Plus des métadonnées concernant la météo elle-même pour la couverture nuageuse, la température etc.

Vous voulez lire le flux RSS sur le trafic routier, moteur de recherche traffic flux rss

RSS 2.0 - Infotrafic®
www.info**trafic**.com/plus.php?link=**fluxrss**.php&region=
Site d'information sur la circulation routière, info **trafic**, bouchons, travaux, **... Flux RSS**. Qu'est ce qu'un fil RSS ? Les fils RSS (Really Simple Syndication) sont **...**

Et nous arrivons au final ici
http://www.infotrafic.fr/rss_infotrafic.php



News Infotrafic
Etat des routes en temps réel

Accident à Bagnolet(75) sur A3 (sens NORD-SUD)
jeudi 12 janvier 2012 01:39

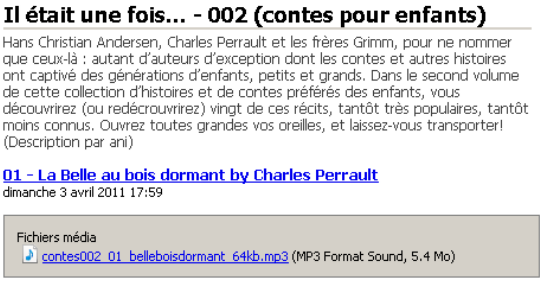
12-01-2012 01:13 Accident à Bagnolet(75) sur A3 (sens NORD-SUD) à Bagnolet(75) sur A

Accident à Villeneuve-Loubet(6) sur D2085 (sens Deux sens)
jeudi 12 janvier 2012 01:39

11-01-2012 23:37 Accident à Villeneuve-Loubet(6) sur D2085 (sens Deux sens) à Villeneu

```
<rss version="0.91">
    <channel>
        <title>News Infot
        <description>Etat
        <link>http://www.
        <lastBuildDate>Th
        <generator></gene
        <image>
            <url>http://w
            <title>Logo I
            <link>http://
            <description>
        </image>
        <item>
            <title>Accide
            <link>http://
            <description>
            <author>conta
            <category>Tra
            <pubDate>Thu,
        </item>
```

Etc. Vous l'avez compris la seule limite sera votre imagination.
Ce qui va nous importer à présent c'est de savoir comment nous allons traiter toutes ces informations avec notre adorable lapin.
Et pour illustrer notre code j'ai personnellement retenu ce site
http://librivox.org/rss/3998

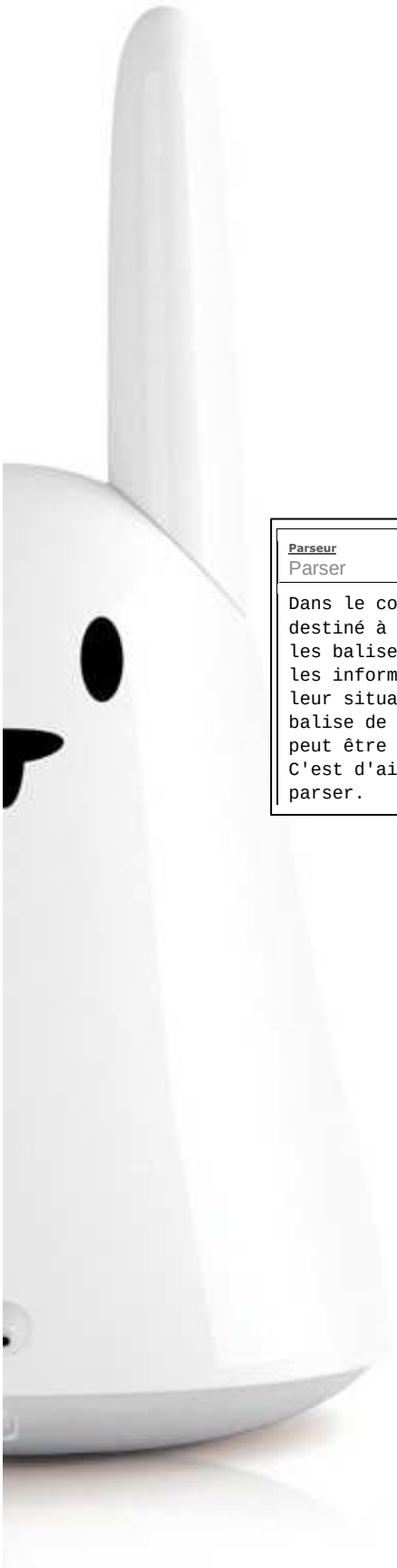


Il était une fois... - 002 (contes pour enfants)
Hans Christian Andersen, Charles Perrault et les frères Grimm, pour ne nommer que ceux-là : autant d'auteurs d'exception dont les contes et autres histoires ont captivé des générations d'enfants, petits et grands. Dans le second volume de cette collection d'histoires et de contes préférés des enfants, vous découvrirez (ou redécouvrirez) vingt de ces récits, tantôt très populaires, tantôt moins connus. Ouvrez toutes grandes vos oreilles, et laissez-vous transporter!
(Description par ani)

01 - La Belle au bois dormant by Charles Perrault
dimanche 3 avril 2011 17:59

Fichiers média
contes002_01_belleboisdormant_64kb.mp3 (MP3 Format Sound, 5.4 Mo)

Petites (nombreuses en fait) recherche sur le net à la recherche d'un « convertisseur » de fichier XML.
Le plus intéressant que j'ai trouvé c'est XML DOM.

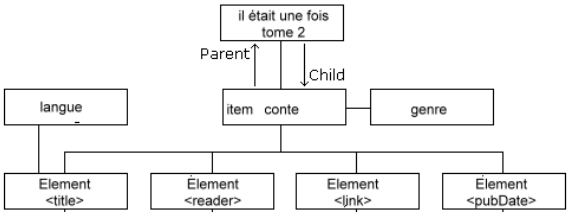


XMLDOM

DOM pour Document Object Model

Le XML DOM définit un standard qui permet d'accéder et de manipuler les données d'un fichier XML.

Votre fichier XML sera vu comme un arbre et il ne vous restera qu'à cueillir les fruits.



Nous n'allons surtout pas réinventer le monde et nous utiliserons donc bien entendu un « parser » (parseur en français), c'est comme cela que ça s'appelle, existant.

Parseur
Parser

Dans le contexte de l'Internet, analyseur syntaxique destiné à récupérer les informations contenues dans les balises d'un document XML. Cet outil distinguera les informations en fonction de leur contenu et de leur situation dans le document : balise de début, balise de fin, etc. Plus généralement, un parseur peut être assimilé à un outil d'analyse syntaxique. C'est d'ailleurs le sens premier du terme anglais parser.

Celui que j'ai retenu s'appelle xmldom.js, il faudra le mettre dans le répertoire de votre application et ajouter cette ligne au début de votre main.js

```
include("util.js");
include("xmldom.js");
```

Le fichier est trouvable sur le net mais je l'ai également mis sur mon serveur ici

A présent ce que nous devons faire c'est charger le fichier xml dans dans Karotz pour traitement, nous allons utiliser l'instruction http.get que l'on retrouve sur notre site préféré http://dev.karotz.com/sdk/#General-methods

On aura donc :

```
var mon_rss = http.get("http://librivox.org/rss/3998");
```

L'instruction qui suit va alimenter XMLDOM.

Un (rapide) parcours de xmldom.js nous amène ici

```
function XMLDoc(source, errFn) {
    /**********************************************************
    function:       XMLDoc

    Author: mike@idle.org

    Description:
        a constructor for an XML document
        source: the string containing the document
        errFn: the (optional) function used to log errors
    **********************************************************
```

Nous allons faire l'impasse de errFn puisque c'est optionnel et ne retenir que XMLDoc(source)

On écrira donc :

```
var mon_xml = new XMLDoc(mon_rss);
```

Ce qui donne donc pour l'instant :

```
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    var mon_rss = http.get("http://librivox.org/rss/3998");
    var mon_xml = new XMLDoc(mon_rss);
```

On peut d'ailleurs faire l'économie d'une ligne en écrivant

```
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    var mon_xml = new XMLDoc(http.get("http://librivox.org/rss/3998"));
```

Et maintenant comment allons chercher nos informations ?

Vous vous souvenez de mon jeu de cartes des 7 familles (tutoriel 3)

Dans ma famille mon_xml je veux avoir accès aux Node (les nœuds) du document. On écrira donc

```
mon_xml.docNode
```

Puis je veux sélectionner un node en particuliers :

```
mon_xml.docNode.selectNode(ici le node qui m'intéresse)
```

A présent que je suis positionné sur un node en particuliers je veux son contenu par exemple (texte)

```
mon_xml.docNode.selectNode(ici le node qui m'intéresse).getText() ;
```

Aussi simple que cela ;) Enfin presque parce qu'il faut avoir une bonne vision de notre arborescence malgré tout.

Je veux que mon lapin me dise le titre du premier conte (title) ainsi que le nom du narrateur (reader).

Pour le titre on aura :

```
var titre = mon_xml.docNode.selectNode("/channel/item[0]/title").getText();
```

Pour le narrateur on aura :

```
var titre = mon_xml.docNode.selectNode("/channel/item[0]/reader").getText();
```

Voici notre main.js que je vous laisse essayer.

```
include("util.js");
include("xmldom.js");

var karotz_ip="192.168.1.46";//ici votre adresse IP mais ai-je encore besoin de vous le dire ;)
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        exit();
    }
    return true;
}
var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }
    return true;
}
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    var mon_rss = http.get("http://librivox.org/rss/3998");
    var mon_xml = new XMLDoc(mon_rss);
    var titre = mon_xml.docNode.selectNode("/channel/item[0]/title").getText();
    var lecteur = mon_xml.docNode.selectNode("/channel/item[0]/reader").getText();

    karotz.tts.start("Bonjour, je vais vous lire : " + titre + ". lu par : " + lecteur, "fr", exitFunction);
}

karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

On essaye de suite histoire de voir si on n'a pas perdu la main. Au passage j'espère que vous avez de vous-même pensé à mettre http dans le descriptor.xml sinon il n'est pas trop tard ;)

```
<access>http</access>
<access>tts</access>
<access>button</access>
<access>multimedia</access>
```

Pas grand-chose à dire, ça fait ce qu'on lui demande de faire, personnellement je ne suis pas satisfait du « by » qui est mal lu mais en même temps on se demande bien ce qu'il fait la.

Qu'à cela ne tienne voilà l'occasion d'apprendre encore quelque chose, nous allons remplacer « by » par « par » en utilisant l'instruction replace donc sous la ligne var titre = etc on met :

```
titre = titre.replace("by", "par") ;
```

Pour le numéro au début du titre faut voir, on peut imaginer le garder, faire lire au lapin tous les titres en donnant justement leur numéro et ensuite choisir notre conte en énnonçant son numéro. Pourquoi pas.

Mais si jamais ça vous gêne voici comment raccourcir une chaîne de caractères :

Substring

```
ma_chaine.substring(valeur de départ,valeur de fin)
```

ici ma_chaine c'est titre et je veux enlever les 4 premiers caractères donc valeur de départ vaudra 5.

Pour la valeur de fin comme la chaîne est de longueur variable je vais juste demander à avoir la longueur de la chaîne, on écrit ma_chaine.length.

Donc on écrira : `titre = titre.substring(5, titre.length);`

On note au passage que cela vaut aussi pour extraire une sous chaîne d'une chaîne.

Je vous laisse essayer. Pendant que je prépare la suite, la lecture du conte ;)

On obtient le lien par le nœud « link »

```
Var lien = mon_xml.docNode.selectNode("/channel/item[0]/link").getText();
```

Et pour le lire :

```
karotz.multimedia.play(lien, exitFunction);
```

Enfin presque parce que 2 callback en exitFunction pas top, souvenez vous du tutoriel 5 sur le chaînage des instructions et on se souvient des variables globales et locales, ici var lien est local si je veux l'utiliser ailleurs va falloir la déclarer en global.

Voici le code qui va bien

```
include("util.js");
include("xmldom.js");
var lien;
var karotz_ip="192.168.1.46";
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        exit();
    }
    return true;
}

var lire = function(event) {
    if((event == "CANCELLED") || (event == "TERMINATED")) {
        karotz.multimedia.play(lien, exitFunction);
    }
    return true;
}
var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }
    return true;
}
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    var mon_rss = http.get("http://librivox.org/rss/3998");
    var mon_xml = new XMLDoc(mon_rss);
    var titre = mon_xml.docNode.selectNode("/channel/item[0]/title").getText();
    titre = titre.replace("by", "par");
    titre = titre.substring(5, titre.length);
    var lecteur = mon_xml.docNode.selectNode("/channel/item[0]/reader").getText();
    lien = mon_xml.docNode.selectNode("/channel/item[0]/link").getText();
    karotz.tts.start("Bonjour, je vais vous lire : " + titre + ". lu par : " + lecteur, "fr", lire);
}

karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Que se passe-t-il si on remplace les item[0] par item[4] par exemple ? on peut essayer jusqu'à 19 et comme nous sommes un peu faineant, euh pas vous ah bon, moi si alors je remplace index[0] par index[numero] et au

début de mon programme je mets

Var numero = 0 ; //et je n'aurai à changer que ce 0 par le numéro que je veux (entre 0 et 19).

C'est bien gentil tout ça me direz vous mais d'abord comment sait-on qu'il y a 20 items puisque nous allons lire des sites dynamiques et que forcément il n'y aura pas toujours le même nombre d'articles (enfin ça dépend, un site qui annonce les nouvelles du jour ou le trafic a forcément une longueur variable, un site qui donne la meteo sur 3 jours aura toujours 3 jours, il faudra donc s'adapter, enfin ce qui vaut pour un fixe vaut pour un dynamique et réciproquement.

Vous n'avez pas tout compris, c'est ça ?

Je recommence en fait on aimerait savoir, par programme combien il y a d'articles dans mon flux rss à l'instant où je l'ai lu.

Je vais donc demander à mon XMLDOM de me lister les éléments « item » qu'il y a dans le nœud « channel ». XLMDOM va me retourner un tableau contenant autant d'item qu'il en aura trouvé.

```
var tableau_item = mon_xml.docNode.selectNode('/channel').getElements("item");
```

Il ne me restera plus qu'à prendre la longueur de ce tableau qui me donnera le nombre d'item.

```
var nbre_item = tableau_item.length;
```

Ce que nous allons faire dans un premier temps c'est de demander au lapin de nous lire tous les titres.
Ensuite on verra ce qu'on fait ☺
Je ne sais pas si vous avez déjà réfléchi à la chose, éventuellement je vous laisse faire d'ailleurs car je pense qu'arrivé au 6ème tutoriel vous devez commencer à être presqu'aussi bon que moi !
Quoi ? Meilleur que moi ? Mais j'espère bien ! Quelle plus belle récompense que de voir l'élève surpasser le maître.

Ici vos notes

karotz - tutoriel pour débutant sous Windows -

Voici ce que j'ai fait :

Définition de mes variables en « Global » donc au début.

```
include("util.js");
include("xmldom.js");

var lien, titre, auteur, mon_xml, nbre_item,i;
var fin_lecture = 0;
var karotz_ip="192.168.1.46";
```

Fin_lecture est une bascule (cf tutoriel 5) donc 2 valeurs possibles (0/1) ça pourrait être (vrai/faux ou oui/non)

Gestion du bouton

```
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        exit();
    }
    return true;
}
```

Ici on pourrait imaginer que l'appui « SIMPLE » arrête la lecture des titres pour lancer la lecture du conte.

Lecture d'UN titre, le numéro à lire est passé en argument.

```
var lecture = function(numero) {
  fin_lecture = 0;
  titre = mon_xml.docNode.selectNode("/channel/item["+ numero+"]/title").getText();
  titre = titre.replace("by", "par");
  titre = titre.substring(5, titre.length);
  lecteur = mon_xml.docNode.selectNode("/channel/item["+ numero+"]/reader").getText();
  lien = mon_xml.docNode.selectNode("/channel/item["+ numero+"]/link").getText();
  karotz.tts.start("Je peux vous lire : " + titre + ". lu par : " + lecteur, "fr", fin_titre);
}
```

On note « numero » qui est le numéro de l'item passé en argument. Donc ce qui était écrit ("/channel/item[0]/title") devient ("/channel/item["+ numero+"]/title")
Le drapeau fin_lecture est mis à zéro car on va commencer la lecture

Je teste la fin de la lecture d'un titre.

```
var fin_titre = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        fin_lecture = 1;
    }
    return true;
}
```

Si pas terminé je continue ma vie (enfin le programme continue sa vie)
Si terminé je mets mon drapeau fin_lecture à 1 et le programme reprend son cours.

Ici pas besoin d'explication ;)

```
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    mon_xml = new XMLDoc(http.get("http://librivox.org/rss/3998"));
    nbre_item = mon_xml.docNode.selectNode('/channel').getElements("item").length;
    log("nbre item " + nbre_item);
```

Ici c'est plus intéressant nous apprenons à faire une boucle !
Non que dis-je, pas une mais deux boucles (tout ça pour le même prix ;)

```
    for (i = 0; i < nbre_item; i++) {
        log("I vaut : " +  i);
        lecture(i);

        while (fin_lecture == 0) { };
    } //fin du for i
    exit();
}
```

Traduction : tu démarres avec i à 0, tu regardes si i est inférieur à mon nombre d'item, si oui alors tu fais ce qu'il y a en dessous (jusqu'au « } » qui ferme la boucle for) quand tu as finis un premier tour tu fais +1 à i (i++) et tu recommences jusqu'à ce que i soit supérieur ou égal à nbre_item), auquel cas tu sors de la boucle donc après le « } » et tu fais ce qui suit (ici on sort du programme par exit()).
On note que le i passé en argument => lecture(i) se transforme en « numero » dans la fonction lire function(numero). Ça n'est pas anormal !

Vous vous souvenez du tutoriel 5 sur le chaînage des instructions, le code Java qui continue d'exécuter les lignes les unes après les autres sans s'occuper de savoir s'il peut le faire ? Ou lorsque plus haut j'écris « le programme continue sa vie » c'est justement ici qu'il la continue, il lance la lecture d'un premier titre, revient ici pour faire le i suivant, revient aussitôt ici pour lancer la lecture du prochain sans s'occuper de savoir si la lecture du premier est terminée et ainsi de suite il a le temps de passer 20 fois dans cette boucle avant même que la lecture du premier titre ne soit terminée, ce qui ne fait pas notre affaire bien entendu. Il va donc falloir calmer les ardeurs de notre programme. On va donc lui demander d'attendre que le drapeau fin_lecture soit à 1 avant de continuer. En français on va lui dire : tant que fin_lecture vaut zéro alors attends boucle sur toi-même sinon passe à la suite. En java cela s'écrit : While (fin_lecture ==0){} ; On pourrait mettre des instructions dans la partie entre accolade comme un log("coucou je boucle ") ; par exemple ou tester si un évènement intervient comme l'appui sur la tête ou une oreille bougée.

```
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Lecture du conte pour lequel on est en train de lire le titre lorsqu'on appuie une fois sur la tête =>
On définit tout d'abord un drapeau

```
var lecture_conte = false;
```

J'avais envie de changer avec ce drapeau pour vous montrer une autre façon de tester si vrai ou faux.

En oui/non le test se fait par

```
If (lecture_conte == 0){alors fait ceci}
```

Ou

```
If (lecture_conte == 1){alors fait ceci}
```

Sur un vrai / faux on pourrait écrire la même chose

```
If (lecture_conte == false){alors fait ceci}
```

Ou

```
If (lecture_conte == true){alors fait ceci}
```

Mais la particularité du true/false est qu'il est facultatif
Si j'écris

`If ( !lecture_conte){alors fait ceci}` lire => if not lecture_conte,
comprendre si je n'ai pas lecture_conte (donc si lecture_conte est faux)
Ou

`If (lecture_conte){alors fait ceci}` lire => if lecture_conte,
comprendre si j'ai lecture_conte (donc si lecture conte est vrai)

Demande de lecture du conte dont le titre est en train d'être lu par l'appui simple sur la tête comme ceci

```
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        exit();
    }
    if (event == "SIMPLE") {
        lecture_conte = true;
        karotz.ears.moveRelative(360, 0);//visualise pris en compte du bouton
    }
    return true;
}
```

Et c'est dans ma fonction fin_titre que je vais tester en plus ce drapeau (en jaune les changements)

```
var fin_titre = function(event) {
    if ((!lecture_conte) && ((event == "CANCELLED") || (event == "TERMINATED"))) {
        fin_lecture = 1;
    }
    if ((lecture_conte) && ((event == "CANCELLED") || (event == "TERMINATED"))) {
        log("je pass en lecture conte ");
        karotz.multimedia.play(lien, fin_du_conte);
    }
    return true;
}
```

Et c'est dans ma fonction fin_conte que je réinitialise fin_conte à false mais également que je mets fin_lecture à 1 car il ne faut pas oublier ma boucle while qui attends justement que ce drapeau passe à un pour lire le titre suivant.

```
var fin_du_conte = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        fin_lecture = 1;
        lecture_conte = false;
    }
    return true;
}
```

J'appuie donc sur le bouton, dès que le titre est terminé je vais passer à la lecture du conte. Dès que le conte sera terminé je reprendrai la lecture des titres.
Maintenant je peux avoir envie d'interrompre la lecture de ce conte soit parce qu'il m'ennuie, soit parce que je me suis trompé soit tout simplement parce que je n'ai pas que ça à faire pendant mes tests ;)

Je modifie donc mon évènement appui simple de cette façon

```
    if (event == "SIMPLE") {
        if (!lecture_conte) {
                lecture_conte = true; ;
                log("lecture_conte :" + lecture_conte);
                karotz.ears.moveRelative(360, 0);}
        else {
                karotz.ears.moveRelative(0,360);
                karotz.multimedia.stop(fin_du_conte);

         }
```

Vous pouvez encore agrémenter ce programme en le faisant avancer dans ses « Item » en bougeant une oreille et reculer en bougeant une autre oreille
Vous vous souvenez, on a utilisé ces oreilles dans le tutoriel 5 donc ça n'a plus de secret pour vous.
D'abord le listener (on vérifie qu'on a bien ears dans descriptor.xml)

```
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    karotz.ears.addListener(earsListener);
    mon_xml = new XMLDoc(http.get("http://librivox.org/rss/3998"));
```

Puis la gestion des oreilles qui pourrait ressembler à ceci

```
var earsListener = function(event) {
log("oreilles : " + event);
    if (event == "START_LEFT") {
          i -= i;
          if (i < 0){i = 0;}
          }
     if (event == "START_RIGHT") {
          i += i;
          if (i > nbre_item-1){i = nbre_item - 1;}
          }
        karotz.multimedia.stop(fin_du_conte);
}
```

Ça fonctionne mais ça n'est pas d'une précision professionnelle, le compteur avance (ou recule) souvent de plus d'une position ce qui n'aide pas vous en conviendrez. J'avais pensé tester l'évènement STOP de l'oreille mais mon log (event) montre qu'on l'a peu souvent lui aussi.
On pourrait donc ajouter un drapeau, un de plus mais en programmation c'est indispensable, qui prendrait en compte le fait que l'oreille a été bougée et tant que l'évènement demandé par ce bouger d'oreille n'aura pas été validé j'aurai beau encore bouger l'oreille il ne fera rien de plus.

Mais j'ai peut-être plus simple en fait (quoique la notion de simplicité est toute relative mais j'avais envie de m'amuser un peu en fait).
La faiblesse du code actuel c'est qu'on touche au pointeur i (en plus ou en moins) à chaque fois qu'on touche l'oreille donc on a vite fait de faire exploser sa valeur. L'astuce va donc consister à copier i (avec plus 1 si incrément et -2 si décrément car -1 ferait lire celui en cours) dans un autre

pointeur que j'appelle j tout simplement donc on pourra passer 10 fois dans la boucle le nouveau pointeur aura toujours la valeur i+1 ou i-2 .

Il ne reste plus qu'à récupérer la valeur de j pour notre pointeur i dans la boucle initiale et le tour est joué.

Voici le code

```
var lien, titre, auteur, mon_xml, nbre_item,i,j;
...
var earsListener = function(event) {
log("oreilles : " + event);
    if (event == "START_LEFT") {
          j = i-2;
          if (j < -1){j = -1;}//-1 car la boucle initiale commence par faire +1
        log("J left vaut : " + j);
        }
     if (event == "START_RIGHT") {
          J= i+ 2;
          if (j > nbre_item-1){j = nbre_item - 1;}
        log("j right vaut : " + j);
          }
        if (lecture_conte){karotz.multimedia.stop();}//si lecture conte
        else {karotz.tts.stop();} //si lecture du titre
}

...
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    karotz.ears.addListener(earsListener);
    mon_xml = new XMLDoc(http.get("http://librivox.org/rss/3998"));
    nbre_item =
mon_xml.docNode.selectNode('/channel').getElements("item").length;
    log("nbre item " + nbre_item);
    for (i = 0; i < nbre_item; i++) {
        j = i;
        lecture(i);
        while (fin_lecture == 0) {  };
        i = j;
        log("I vaut : " + i);
    } //fin du for i
    exit();
```

Et à présent ? On a terminé, bon vous l'avez remarqué les contes sont à peine audibles, je n'y suis pour rien, il n'y a rien pour le volume dans l'instruction multimédia mais les tutoriels sont faits pour apprendre avant tout et j'espère que vous aurez encore appris un peu avec celui-ci.

Et surtout même si nous lisons du flux rss c'est un flux plutôt statique on va donc voir ce qu'il faut modifier pour lire un site dynamique.

Imaginons que nous trouvions un site d'information qui diffuse à longueur de journée des infos nous n'aimerions pas relire tout à chaque fois que notre lapin va se connecter, l'idéal serait qu'il reprenne là où nous l'avons quitté. Nous allons également « tomber » sur des sites dont l'XML est noyé dans du HTML il fa donc falloir apprendre à isoler l'information utile.

Nous allons nous intéresser à ce site :

http://www.actu-auto-buzz.fr/component/option,com_ninjarsssyndicator/feed_id,1/format,raw/

UN bref coup d'œil sur le code nous donne qu'une envie, partir en courant en effet si on retrouve bien nos balises habituelles comme « channel », « item », « title » on voit une balise « description » qui en fait est

l'équivalent de notre conte mais notre conte était un fichier mp3 donc facile à lire avec l'instruction multimedia.

Ici la description est du texte donc pas de problème nous utiliserons la fonction tts de notre lapin.

Sauf que, aguerris comme nous le sommes à présent il y a des choses dans ce texte qui doivent nous interpeller et nous faire comprendre que notre lapin ne va pas tout comprendre. Exemple ces guillemets présents dans le texe, nous, nous utilisons les guillemets justement pour dire que c'est du texte. Il y a les images en .jpg, il y a des « balises » html pour mettre en gras par exemple que des choses indigestes pour notre lapin. Et ça n'est pas le moment qu'il nous fasse une indigestion, non ?

```
<description>
  <![CDATA[<p><strong>Après les PV des radars de feux rouges ce sont les infractions pour conduite en état d'iv

annulé peut en cacher un autre</h2>
text-align: justify;"><img src="http://www.actu-auto-buzz.fr/images/images-articles/controle-alcootest.jpg" alt="cc

text-align: justify;">Alerte : dangers en liberté</h3>
text-align: justify;"><img src="http://www.actu-auto-buzz.fr/images/images-articles/ethylometre-SERES.jpg" alt="eth
  </description>
  <author> contact@actu-auto-buzz.fr (Actu auto buzz)</author>
```

Repartons de ce programme pour faire nos tests, en jaune les modifs pour lire notre nouveau site, en rouge les lignes à supprimer.

```
include("util.js");
include("xmldom.js");
var description;
var karotz_ip="192.168.1.29";
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        exit();
    }
    return true;
}

var lire = function(event) {
    if((event == "CANCELLED") || (event == "TERMINATED")) {
        karotz.tts.start(description,"fr", exitFunction);
    }
    return true;
}
var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }
    return true;
}
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    var mon_rss = http.get("http://www.actu-auto-buzz.fr/component/option,com_ninjarsssyndicator/feed_id,1/format,raw/");
    var mon_xml = new XMLDoc(mon_rss);
    var titre = mon_xml.docNode.selectNode("/channel/item[0]/title").getText();
    //titre = titre.replace("by", "par");
    //titre = titre.substring(5, titre.length);
    //var lecteur = mon_xml.docNode.selectNode("/channel/item[0]/reader").getText();
    description = mon_xml.docNode.selectNode("/channel/item[0]/description").getText();
    karotz.tts.start(titre + ".", "fr", lire);
}

karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Je vous laisse essayer et vous comprendrez ce que je veux dire lorsque je parle d'indigestion. ;)

Vous avez vu le sursaut qu'il a avec ses oreilles ? On le sent prêt à vomir ☺

Alors quelle est la solution ? Après le parser pour XML un parser pour HTML ?

PARSER HTML

Pourquoi pas ? Mais est-ce bien utile ? En cherchant un parser HTML sur la toile je suis tombé sur une petite routine qui finalement me va très bien aussi restons à notre devise, pourquoi faire compliqué quand on peut faire simple. Il est malgré tout intéressant de comprendre comment ça fonctionne, après tout nous sommes aussi là pour nous instruire.

Cette routine reprend l'instruction « replace » que nous avons déjà utilisée dans ce tutoriel pour remplacer « by » par « par », là on imagine une routine qui va remplacer des par rien par exemple mais également des « / », « \ », «[» etc.

L'idée est donc de mettre d'un seul coup tout ce que l'on veut supprimer dans l'instruction replace

Au lieu d'écrire

```
Ma_chaine.replace("toto", "") ; // remplace toto par rien dans ma chaine
```

Et faire autant de replace que de choses à supprimer on va utiliser une « expression » en l'occurrence RegExp qui est une « expression régulière ».

RegExp

Une **regex (ou RegExp)** s'apparente à une expression mathématique, car on y trouve des opérateurs, des valeurs et des variables. Les regex permettent de se lancer à la recherche de motifs décrits par la combinaison d'opérateurs et de valeurs. Une utilisation récurrente des regex consiste en la recherche de mots clés dans des

On l'utilse comme ceci

Var ce_que_je_veux_supprimer = new RegExp (ici tout ce que je veux enlever, drapeau indiquant si la recherche doit se faire sur toute la chaine et doit tenir compte de la casse (majuscule et/ou minuscule))

Quelques exemples pour mieux comprendre :

Pour le deuxième argument on peut avoir :
"": rien du tout, aucune option n'est appliquée.
"g": effectue un recherche globale sur toute la chaine de caractères.
"i": n'est pas sensible à la casse, c'est à dire qu'il n'y a pas de distinction majuscules/minuscules.
"g"i: cumul des options g et i c'est à dire que la recherche s'effectue sur toute la chaine, sans distinction majuscules/minuscules.

Dans les exemples qui suivent je ne m'occupe donc pas du 2ème argument.

Pour le premier argument on peut avoir la recherche de :
caractères spéciaux comme par exemple rechercher dans un document :
un saut de page(\f)
un saut de ligne (\n)
un retour chariot (\r)
groupe de caractères comme par exemple rechercher dans un document :
[abcde] => tous les caractères entre crochets
[^abcde] => tous les caractères sauf ceux entre crochets
[\d] => n'importe quel chiffre entre 0 et 9
[\D] => tout ce qui n'est pas chiffre
[\w] => tout ce qui est caractère alphanumérique plus tous les chiffres + _
…
Les occurrences comme par exemple:
+ => le caractère doit apparaître au moins une fois
? => le caractère doit apparaître entre zéro et une fois
{n} => n étant le nombre d'occurrence où le caractère doit apparaître
…
Si vous voulez en savoir plus voici ma source
Et voici la routine que nous allons utiliser

```
var   enleve_balise = function(mon_html)
    {
    if(arguments.length < 3) {
     mon_html=mon_html.replace(/<\/?(?!\!)[^>]*>/gi, '');
    } else {
     var allowed = arguments[1];
     var specified = eval("["+arguments[2]+"]" );
     if(allowed){
      var regex='</?(?!(' + specified.join('|') + '))\b[^>]*>';
      mon_html=mon_html.replace(new RegExp(regex, 'gi'), '');
     } else{
      var regex='</?(' + specified.join('|') + ')\b[^>]*>';
      mon_html=mon_html.replace(new RegExp(regex, 'gi'), '');
     }
    }
    return mon_html;
}
```

Que nous appellerons ici (en jaune)

```
description = mon_xml.docNode.selectNode("/channel/item[0]/description").getText();
description = enleve_balise(description);
```

Et on essaie. Et on applaudit des 2 mains parce que je pense que c'est mérité. Mais on n'en a pas terminé, il y a d'abord ces tags lus à la fin, c'est plutôt moyen non ?
Pas de soucis enfin en ce qui concerne ce site puisque les tags sont à la fin de la description, je vais donc rechercher la position du mot « tags » et couper ma chaine juste avant.
Mais vous l'avez compris, ce qui vaut pour ce site ne vaut pas forcément pour d'autres sites, imaginons un site où les tags seraient au début ☺
Pire un site qui parle des tags dans sa description mais sans positionner lui-même des tags ça ne serait pas cool.
Bien ! On ne peut pas tout faire ici non plus donc en ce qui nous concerne on va demander la position du mot « tags » dans notre chaîne par cette instruction :

lastIndexOf

```
var pos = description.lastIndexOf("Tags",description.length) ;
```

Si "Tags" n'était pas trouvé « position » contiendra « -1 » sinon la position voulue il suffit donc de tronquer après cette position par

```
if (pos > 0){description = description.substring(1,pos);}
```

On remet notre boucle pour lire toutes les news mais cette fois ci on va commencer par la dernière.
Cela s'écrit :

```
for (i = nbre_item - 1; i >-1; i--) {
```

On oublie les oreilles par contre on aimerait qu'il ne nous lise pas à chaque fois les mêmes news.
Comment savoir si une news a déjà été lue ? Et surtout comment mémoriser l'information dans notre lapin car il a une vie notre lapin, d'autres applications vont s'intercaler entre 2 appels à l'appli lecture de notre rss…
Pour connaitre la dernière news lue la première chose à laquelle on pense c'est la valeur de i qui nous indique la news en cours lorsqu'on arrête la lecture. Bien mais est-ce une bonne idée ? Que se pass-t-il si lorsque nous revenons sur ce site il y a 10 news de moins par exemple ? On risque de ne rien retrouver.
Avez-vous regardé le fichier xml avec attention ? Avez-vous vu ce tag :

```
<pubDate>Fri, 13 Jan 2012 13:18:21 GMT</pubDate>
```

Voilà qui est bien, les news sont horodatées. Il suffira de parcourir toutes les news depuis la dernière et de comparer sa date avec la date (que nous aurons mémorisée)
Si la date de la news à lire est inférieure à celle de la dernières news lue alors on la zappe, sinon on la lit.
On a de la chance, il en faut un peu, le format est toujours le même, on va donc pouvoir « formater » cette date afin de pouvoir faire des comparaisons dessus.

D'abord on la lit en prenant soin de déclarer les variables que l'on utilise dans la partie « globale »:

```
date = mon_xml.docNode.selectNode("/channel/item[" + numero + "]/pubDate").getText();
```

Ensuite on va formater cette date à savoir faire une concaténation Année_mois_jour_heure_minute_seconde
Si on veut trier des dates il faut mettre l'année en premier puis le mois et le jour.
Essayer de trier autrement
10012012 (10 janvier 2012)
09032012(9 mars 2012)
On voit bien que 09032012 sera AVANT 10012012 ça n'est pas tout à fait ce que nous voulons alors qu'avec le système retenu
20120110 (10 janvier 2012)
20120309(9 mars 2012)
20120309 sera APRES 20120110 ce qui est vrai.
Pour l'instant dans date nous avons
« Tue, 06 Dec 2011 13:10:12 GMT »
Nous allons isoler l'année

```
date_item = date.substring(12,16) ;
```

Pour isoler le mois c'est un peu plus compliqué car il n'est pas numérique et nous voilà parti pour faire 12 if genre

```
if (date.substring(8,11) == "Jan"){date_item = date_item + "01"}
...
if (date.substring(8,11) == "Dec"){date_item = date_item + "12"}
```

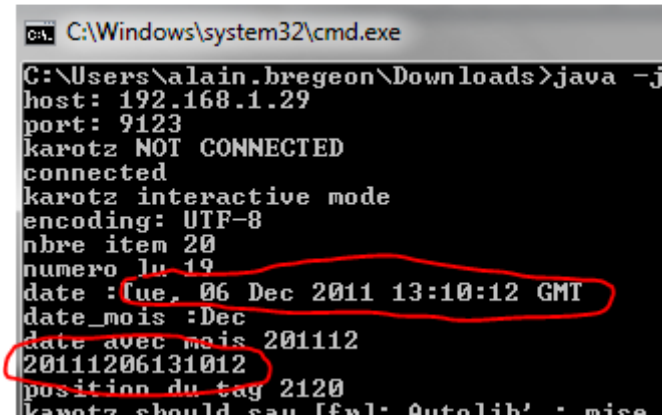
On pourrait utiliser l'instruction switch mais pas sur qu'il y en ait moins à saisir, on verra peut-être cela dans un autre tutoriel.
Les mois sont en anglais voici les abréviations

Janvier	Février	Mars	Avril	Mai	Juin	Juillet	Août	Septembre	Octobre	Novembre	Décembre
Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec

Ensuite le jour et l'heure et le reste

```
date_item = date_item + date.substring(5,7) + date.substring(17,19) + date.substring(20,22) + date.substring(23,25) ;
```

Voici ce que ça donne



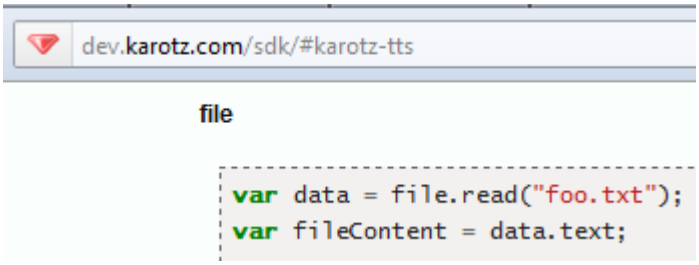
Il ne « reste » plus qu'à gérer cette date, la sauvegarder quand on quitte l'application, donc la relire lorsque qu'on démarre et sauter les items déjà lu (dont la date est inférieur à celle-ci, on pourrait mettre inférieur ou égal mais on peut imaginer que la dernière lecture ait été interrompu avant la lecture complète et il n'est pas aberrant de relire la dernière lue, de toute façon lorsqu'il ne restera plus que ça à gérer on devrait y arriver.

Comment sauvegarder une information

Comment sauvgarder notre date_item, comment la retrouver au démarrage alors que le lapin va forcément avoir d'autres applications, peut-être même sera-t-il éteint.

Il n'y a donc qu'une façon : enregistrer l'information sur le disque. Enfin façon de parler. Le lapin n'a pas de disque mais il a de la mémoire, non volatile c'est-à-dire qui garde l'information même quand le lapin est éteint, c'est exactement ce qu'il nous faut.

Sur notre site favori j'avoue qu'il n'y a pas grand-chose concernant la gestion des fichiers,



Il y a la lecture mais rien sur l'écriture. Mais on en a surmonté d'autre on va donc essayer un file.write

Tout d'abord que se passe-t-il si on veut lire un fichier qui n'existe pas (la question est de savoir si on doit créer au préalble un fichier vide ou pas.

Mais au fait comment allons-nous tester en local ? Il ets clair que nous ne pourrons pas écrire dans le lapin.

En fait quand l'application est « embarquée » il existe un répertoire avec votre application donc quand on va écrire en local dans notre répertoire de l'pplication, ça fera la même chose dans le lapin, il n'y a donc pas de chemin à indiquer.

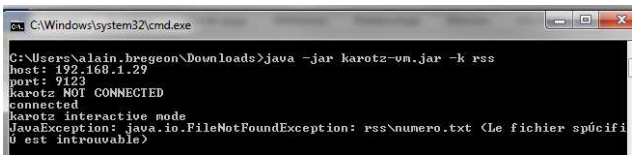
Testons la lecture

File dans notre descriptor.xml

```
<accesses>
   <access>http</access>
      <access>file</access>
      <access>tts</access>
      <access>button</access>
      <access>multimedia</access>
</accesses>
```

```
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    var dernier_numero_lu = file.read("numero.txt");
    log("dernier numero lu " + dernier_numero_lu.text);
```

Voici le résultat qui a le mérite d'être clair !



Fichier introuvable et ça ne rend pas la main.

Il va donc falloir fournir l'appli avec ce fichier déjà présent (i peut être

vide mais quant à le créer autant mettre 00000000000000 dedans (14 fois zéro)

Je crée donc un fichier numero.txt dans le répertoire de mon appli. Et je re teste
YESSSS ça fonctionne
On ajoute le test après le calcul de la date de l'item en cours, juste avant de lire l'item.

```
date_item = date_item + date.substring(5, 7) + date.substring(17, 19) + date.substring(20, 22) + date.substring(23, 25);
log(date_item);
if (dernier_numero_lu.text > date_item) {//cet itema déjà été lu
    log("numero superieur");
    fin_lecture = 1;
    return true;
    }
description = mon_xml.docNode.selectNode("/channel/item[" + numero + "]/description").getText();
description = enleve_balise(description);
```

Et voilà un tutoriel « riche » qui se termine.
J'espère que vous aurez autant de plaisir à le lire et le mettre en pratique que j'en ai eu à vous le préparer.
Ne vous y trompez pas, j'ai l'air de parler comme un livre mais avant de commencer ce tutoriel je ne connaissais rien aux flux rss, rien au XMLDom, c'est grâce à vous, pour vous que je teste et que j'apprends.

Il m'arrive de bloquer quelques heures sur un point particuliers, quelque chose qui ne fonctionne pas comme je voudrais mais vous êtes ma motivation.

Pour aller plus loin quelques liens RSS à creuser

Les fables de la fontaine
Contes pour enfants contessede Ségur
http://librivox.org/rss/3697
http://www.generation-nt.com/divers/services.php
Et page suivante le code complet toujours utilisable en copier /coller

```
include("util.js");
include("xmldom.js");
var lien, titre, auteur, mon_xml, nbre_item, i, j, date, date_item, dernier_numero_lu;
var fin_lecture = 0;
var lecture_description = false;
var karotz_ip="192.168.1.29";//ici votre adresse IP
var   enleve_balise = function(mon_html){
    if(arguments.length < 3) {
     mon_html=mon_html.replace(/<\/?(?!\!)[^>]*>/gi, '');
    } else {
     var allowed = arguments[1];
     var specified = eval("["+arguments[2]+"]" );
     if(allowed){
      var regex='</?(?!(' + specified.join('|') + '))\b[^>]*>';
      mon_html=mon_html.replace(new RegExp(regex, 'gi'), '');
     } else{
      var regex='</?(' + specified.join('|') + ')\b[^>]*>';
      mon_html=mon_html.replace(new RegExp(regex, 'gi'), '');
     }
    }
    return mon_html;
}
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        file.write("numero.txt", date_item);
        exit();
    }
    if (event == "SIMPLE") {
        if (!lecture_description) {
            lecture_description = true; ;
            log("lecture_description :" + lecture_description);
            karotz.ears.moveRelative(360, 0);
        }
        else {
            karotz.ears.moveRelative(0, 360);
            karotz.tts.stop(fin_description);
        }
    }
    return true;
}
var lecture = function(numero) {
    fin_lecture = 0;
    titre = mon_xml.docNode.selectNode("/channel/item[" + numero + "]/title").getText();
    date = mon_xml.docNode.selectNode("/channel/item[" + numero + "]/pubDate").getText();
    log("date :" + date);
    date_item = date.substring(12, 16);
    if (date.substring(8, 11) == "Jan") { date_item = date_item + "01" }
    if (date.substring(8, 11) == "Feb") { date_item = date_item + "02" }
    if (date.substring(8, 11) == "Mar") { date_item = date_item + "03" }
    if (date.substring(8, 11) == "Apr") { date_item = date_item + "04" }
    if (date.substring(8, 11) == "May") { date_item = date_item + "05" }
    if (date.substring(8, 11) == "Jun") { date_item = date_item + "06" }
    if (date.substring(8, 11) == "Jul") { date_item = date_item + "07" }
    if (date.substring(8, 11) == "Aug") { date_item = date_item + "08" }
    if (date.substring(8, 11) == "Sep") { date_item = date_item + "09" }
    if (date.substring(8, 11) == "Oct") { date_item = date_item + "10" }
    if (date.substring(8, 11) == "Nov") { date_item = date_item + "11" }
    if (date.substring(8, 11) == "Dec") { date_item = date_item + "12" }
    date_item = date_item + date.substring(5, 7) + date.substring(17, 19) + date.substring(20, 22) + date.substring(23, 25);
    if (dernier_numero_lu.text > date_item) {//cet item a déjà été lu
        fin_lecture = 1;
        return true;
    }
    description = mon_xml.docNode.selectNode("/channel/item[" + numero + "]/description").getText();
    description = enleve_balise(description);
    var pos = description.lastIndexOf("Tags", description.length);
    if (pos > 0) { description = description.substring(1, pos); }
    karotz.tts.start(titre + ".", "fr", fin_titre);
}
var fin_titre = function(event) {
    if ((!lecture_description) && ((event == "CANCELLED") || (event == "TERMINATED"))) {
        fin_lecture = 1;
    }
    if ((lecture_description) && ((event == "CANCELLED") || (event == "TERMINATED"))) {
        log("je passe en lecture conte ");
        karotz.tts.start(description, "fr",fin_description);
    }
    return true;
}
var fin_description = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        fin_lecture = 1;
        lecture_description = false;
    }
    return true;
}
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    dernier_numero_lu = file.read("numero.txt");
    mon_xml = new XMLDoc(http.get("http://www.actu-auto-buzz.fr/component/option,com_ninjarsssyndicator/feed_id,1/format,raw//"));
    nbre_item = mon_xml.docNode.selectNode('/channel').getElements("item").length;
    for (i = nbre_item - 1; i > -1; i--) {
        lecture(i);
        while (fin_lecture == 0) { };
    } //fin du for i
    exit();
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```