



## Sommaire

|   |    |
|---|----|
| Bobo la tête .....  | 2  |
| Où comment échanger avec le lapin.....  | 2  |
| Et les puces ? .....  | 5  |
| Vous ne saviez pas que votre lapin avait des puces? .....                             | 5  |
| rfid .....  | 6  |
| Gagner ses galons de programmeur 1 <sup>er</sup> échelon .....                        | 9  |
| Solution de l'examen de passage pour être programmeur 1 <sup>er</sup> échelon : ..... | 14 |
| Vous aviez trouvé? Félicitation ! .....   | 14 |
| Fakir .....   | 15 |
| Bouclons .....  | 21 |
| Répète après moi ? .....  | 23 |
| Il a de grandes oreilles mais est-ce qu'il nous entend ? .....                        | 23 |



## Bobo la tête

### Où comment échanger avec le lapin

Nous n'avons pas encore utilisé le lapin pour « interagir » avec lui, imaginons qu'il dise « aie, bobo la tête à chaque fois que nous pressons le bouton sur sa tête, ça pourrait être amusant non ?



Euh c'est quoi tous ces boutons ? C'est grave docteur ?



Je pense que vous mangez trop de carottes !

Ne perdons pas les bonnes habitudes, recherchons sur le site <http://dev.karotz.com/sdk/> ce qui parle du bouton (button en anglais) on trouve ceci :

```
button
karotz.button.addListener(function(event) );
```

*add a listener on the button*

```
value of event : { "SIMPLE"; "DOUBLE", "TRIPLE", "MULTIPLE";
"LONG_START"; "LONG_STOP"; }
```

Mais revenez, pourquoi étiez-vous parti si vite ?

En fait ça doit être pour ça qu'il a attrapé des boutons notre lapin 😊

Il est vrai que de prime abord ça n'est pas très causant mais si vous voulez acquérir vos galons de programmeur chef il va falloir faire quelques efforts.

Appelons professeur Karotz pour nous éclairer



Le fait d'appuyer sur le bouton peut être considéré comme un évènement (event) mais il y a différentes façons d'appuyer sur le bouton, une fois, 2 fois, 3 fois, plus de fois encore, longtemps etc.

On va donc réagir lors de l'appui sur le bouton pour tester de quelle façon il a été appuyé et réagir en conséquence.

En langage informatique, au moins avec celui que nous utilisons qui est du javascript il y a plusieurs façon d'écrire cela.

En ce qui nous concerne voici la façon que nous retiendrons

```
karotz.button.addListener(Quefaire(si_appui_sur_le_bouton)
{ faire ceci }
);
```

La traduction en langage réel est

```
karotz.button.addListener(function(si_appui_sur_le_bouton)
{ karotz.tts.start("Aie, ça fait mal!", "fr", exitFunction); } );
```

Voici notre main.js

```
include("util.js");
var karotz_ip = "192.168.1.46" // ← ATTENTION : ici votre adresse IP
var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }
    return true;
}
var onKarotzConnect = function(data) {
    karotz.button.addListener(function(si_appui_sur_le_bouton) {
        // fait ceci
        karotz.tts.start("Aille, cela fait mal!", "fr", exitFunction);});
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```



Lors du lancement du programme on note un changement

```
ca: mon_appli_karotz.bat - Raccourci
F:\Users\alain\Downloads>java -jar karotz-vm.jar -k app_helloworld
host: 192.168.1.46
port: 9123
karotz NOT CONNECTED
connected
karotz interactive mode
```

Il ne rend pas la main et la lumière est blanche, si nous ne faisons rien pendant 15 mn il rendrait la main automatiquement.

Dans ce premier exemple, fidèle à la philosophie de ces tutoriels qui est « pourquoi faire compliqué lorsqu'on peut faire simple » on se rend compte qu'il se moque de la façon dont nous appuyons sur le bouton. Bien entendu nous allons agrémenter notre programme en faisant dire des choses différentes à notre lapin selon que l'on appuie une fois ou deux fois sur le bouton.

Cela pourrait s'écrire comme cela

```
karotz.button.addListener(Quefaire(si_appui_sur_le_bouton)
{
  Si l'appui sur le bouton est une fois
    faire ceci
  Si l'appui sur le bouton est deux fois
    faire cela
  etc...
});
```

En langage informatique ça donne cela, je vous laisse essayer

```
var onKarotzConnect = function(data) {
  karotz.button.addListener(function(si_appui_sur_le_bouton) {
    if (si_appui_sur_le_bouton == "SIMPLE")
      karotz.tts.start("J'adore les scrounches scrounches!",
"fr", exitFunction);
    if (si_appui_sur_le_bouton == "DOUBLE")
      karotz.tts.start("Aille, cela fait mal!", "fr", exitFunction);
  });
  return true;
};
```



## Et les puces ?

Vous ne saviez pas que votre lapin avait des puces?



Surtout ne prenez pas l'insecticide ça ne servirait pas à grand-chose ☺

Je veux parler des puces RFID, et oui encore un acronyme, encore en anglais ceci étant nous n'allons pas appeler le professeur Karotz à l'aide nous devrions nous en sortir sans lui, l'explication est très simple en fait et l'anglais vaut presque le français.

RFID = Radio Frequency Identification Device

Si vous avez eu l'occasion de prendre l'autoroute peut-être avez-vous vu cette file noté « T » qui permet aux voitures de passer sans s'arrêter (et sans payer en apparence du moins) il y a un émetteur « radio » dans la voiture et autour de la barrière de péage un récepteur « radio » lui aussi et pour que la facture arrive chez la bonne personne on imagine que le boîtier qui est placé dans la voiture a un code unique qui permet de lier ce boîtier à son propriétaire. Et bien c'est la même chose avec vos accessoires, les flatnanos jaune et vert fournis avec votre lapin par exemple ou ceux que vous avez pu acquérir, nanoztag ou books.

Donc vous avez des émetteurs, le lapin est un récepteur nous allons essayer de faire fonctionner les 2 ensembles.

Vous ne m'avez même pas attendu pour aller sur le site que vous connaissez à présent par cœur (<http://dev.karotz.com/sdk/>) vous avez cherché (et trouvé) ce qui concerne le RFID



## rfid

```
karotz.rfid.addListener(function(data);
```

*add a listener on rfid*

data is an object that contain information about the tag :

**the tag id :**  
data.id

**If the tag goes in or out the reader range:**

data.direction : enum { IN = 1; OUT = 2; }

### preprogrammed data

```
data.app : enum {none = 0; meteo = 1; trafic = 2;}  
data.type : enum { NONE_TYPE = 0; FLAT = 1; NANOZTAG = 2; ZTAMPS = 3;  
}  
data.pict : enum { NONE_TYPE = 0;}  
data.color : enum { NONE_COL = 0; RED = 1; BLUE = 2; GREEN = 3;  
YELLOW = 4;  
PINK = 5; BLACK = 6; GREY = 7; ORANGE = 8; PURPLE = 9; WHITE = 10; }  
(color are going to change due to specific color of tag sold with  
Karotz )
```



Et si nous tentions l'impossible, encore une fois ? Faire simple alors que ça paraît compliqué ! L'idée vous séduit ? Moi oui ! Alors on essaie !

Reprenons notre première instruction de test du bouton que voici :

```
karotz.button.addListener(function(si_appui_sur_le_bouton)  
{ karotz.tts.start("Aie, ça fait mal!", "fr", exitFunction); } );
```

Et adaptons là au RFID ce qui donnerait ceci

```
karotz.rfid.addListener(function(si_puce_rfid)  
{ karotz.tts.start("Mais j'ai des puces!", "fr", exitFunction); } );
```

Et essayons, qu'est-ce qu'on risque ? Le lapin ne va pas exploser de toute façon ;)

Votre main.js ressemble à ceci

```
include("util.js");  
var karotz_ip = "192.168.1.46" // ← ATTENTION : ici votre adresse IP  
var exitFunction = function(event) {  
    if ((event == "CANCELLED") || (event == "TERMINATED")) {  
        exit();  
    }  
    return true; }  
  
var onKarotzConnect = function(data) {  
    karotz.rfid.addListener(function(data) {  
        karotz.tts.start("Mais j'ai des puces !", "fr", exitFunction); });  
    }  
    karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

## karotz - tutoriel pour débutant sous Windows -



On démarre le programme, on passe une puce rfid sous la bouche du lapin et on s'extasie devant le génie que nous sommes 😊

Alors plutôt sympa cette fonction RFID et pas compliquée du tout ;)

Comment ? Qu'est-ce que vous dites ? Ça ne fonctionne pas ? Mais ce n'est pas possible ça !

Je vais vous faire un aveu, je fais le fier comme ça, celui qui sait tout mais je vous assure je suis aussi un débutant en lapin.

La différence entre vous et moi c'est que moi je sais que cette instruction DOIT fonctionner, on ne peut pas s'être trompé à ce point là et pourtant l'informatique m'apprend tous les jours qu'il faut rester humble devant ces technologies, ne pas avoir de certitudes, savoir se remettre en question, savoir reconnaître qu'on peut s'être trompé.

Surtout ne pas se décourager, relire, traquer l'erreur de frappe, la virgule qui pourrait manquer, refaire son raisonnement et si vous êtes sûr que ça doit fonctionner alors il faut se dire que le problème est ailleurs (et j'ai découvert ce problème après être allé me coucher déçu mais optimiste pour la suite) en me réveillant le lendemain, mais oui mais c'est bien sur, ne faut-il pas dire au lapin quels sont les événements qu'on veut qu'il traite, après tout si je ne veux pas qu'il traite les puces RFID j'ai le droit, si je ne veux pas qu'il se passe quelque chose en appuyant sur le bouton j'ai le droit alors où est le fichier dans lequel nous lui donnerions ces droits ? Ce qui est bien c'est qu'il n'y en a pas 50 dans notre répertoire « app\_helloworld » on devrait le trouver facilement.

Et si c'était dans le fichier descriptor.xml ? Le nom me plait bien, ouvrons vite ce fichier. L'ouvrir avec quoi me direz-vous ? C'est peut-être même Windows qui vous le demande si vous vous êtes précipité pour l'ouvrir, c'est qu'ils sont impatients ces jeunes 😊

Pour l'ouvrir on utilisera Wordpad (comme pour le main.js, je vous invite donc à relire la méthode sur le premier tutoriel)

Une fois ouvert voici ce qu'on y voit :

```
<descriptor>
  <version>1.0.0</version>
  <accesses>
    <access>tts</access>
    <access>button</access>
  </accesses>
</descriptor>
```

Pas besoin d'être expert pour comprendre qu'on y parle d'accès, intéressant, pour remarquer le mot « Button » ah oui on ne s'était pas posé de question pour lui, le mot « TTS » (la parole), je pense qu'on est sur le bon fichier.



Alors osons ! Insérons ceci dans le fichier

```
<descriptor>
  <version>1.0.0</version>
  <accesses>
    <access>tts</access>
    <access>button</access>
    <access>rfid</access>
  </accesses>
</descriptor>
```

Enregistrons ce fichier et exécutons de nouveau notre programme.

Passons la clé jaune sous sa bouche et alléluia ça marche cette fois ci.

Que se passe-t-il si nous passons la clé verte ? Et une autre clé si vous en avez en votre possession ? Nous obtenons le même résultat.

Conclusion : notre programme (minimaliste mais suffisant) lit DU rfid, nous allons apprendre à lire LE rfid (enfin pour ceux qui veulent)





## Gagner ses galons de programmeur 1<sup>er</sup> échelon

Jusqu'à présent vous étiez programmeur stagiaire et je reconnais que vous vous êtes plutôt bien débrouillé, je propose pour celles et ceux que ça intéresse d'aller un petit peu plus loin dans la programmation à l'aide de cette instruction rfid qui présente quand même une certaine complexité dans l'énoncé qui en est fait sur le site Internet.

Pour les autres pas d'inquiétude allez directement au chapitre suivant, il ne vous manquera rien pour la suite des tutoriels et vous pourrez toujours faire un copier / coller du main.js pour tester malgré tout les possibilités des puces.

On voit bien avec notre exemple sur les puces qu'on a bien « zappé » tous les paramètres mais en avez-vous mesuré les conséquences ?

1. que l'on passe n'importe quelle puce rfid il fait la même chose, c'est bien la peine d'avoir plusieurs clés rfid
2. j'ai 2 lapins, 2 clés jaunes et j'aimerais que la clé jaune de mon lapin 1 ne fonctionne pas sur le lapin 2 et que la clé jaune du lapin 2 ne fonctionne pas sur le lapin 1 (et réciproquement 😊)

Alors on imagine bien que tous les paramètres qui accompagnent les explications sur l'instruction de gestion de la clé rfid vont servir à cela. Mais comment ?

Pour illustrer mon propos nous allons imaginer une famille, la famille Karotz, il y a le père, la mère, le fils, la fille.



Lorsqu'on veut parler :

De la famille on dira Karotz

Lorsqu'on veut parler d'un membre on parlera de « type » et donc comme Karotz.type on a père, mère, fils, fille

On peut imaginer un jeu des 7 familles chaque famille ayant sa couleur, rouge jaune, bleu, vert etc

On associera donc la famille karotz à une couleur de cette façon comme Karotz.color on aura yellow (jaune) red(rouge) green(vert) etc  
Etc.

## karotz - tutoriel pour débutant sous Windows -



Si on relit la définition de l'instruction

```
karotz.rfid.addListener(function(data);
```

```
  the tag id :  
  data.id
```

```
  If the tag goes in or out the reader range:  
  data.direction : enum { IN = 1; OUT = 2; }
```

```
  preprogrammed data
```

```
  data.app : enum {none = 0; meteo = 1; trafic = 2;}  
  data.type : enum { NONE_TYPE = 0; FLAT = 1; NANOZTAG = 2; ZTAMPS = 3;  
  }  
  data.pict : enum { NONE_TYPE = 0; }  
  data.color : enum { NONE_COL = 0; RED = 1; BLUE = 2; GREEN = 3;  
  YELLOW = 4;  
  PINK = 5; BLACK = 6; GREY = 7; ORANGE = 8; PURPLE = 9; WHITE = 10; }  
  (color are going to change due to specific color of tag sold with  
  Karotz )
```

On note une certaine analogie avec ma famille Karotz, il vaut mieux me direz-vous sinon ça n'était pas la peine de faire tout ça ☺)

Il suffit de remplacer data par karotz et tout devient limpide

```
  the tag id :  
  karotz.id
```

```
  If the tag goes in or out the reader range:  
  karotz.direction : enum { IN = 1; OUT = 2; }
```

```
  preprogrammed data
```

```
  karotz.app : enum {none = 0; meteo = 1; trafic = 2;}  
  karotz.type : enum { NONE_TYPE = 0; FLAT = 1; NANOZTAG = 2; ZTAMPS =  
  3; }  
  karotz.pict : enum { NONE_TYPE = 0; }  
  karotz.color : enum { NONE_COL = 0; RED = 1; BLUE = 2; GREEN = 3;  
  YELLOW = 4;  
  PINK = 5; BLACK = 6; GREY = 7; ORANGE = 8; PURPLE = 9; WHITE = 10; }  
  (color are going to change due to specific color of tag sold with  
  Karotz )
```

Mais on fait ça pour de vrai ou pour de faux ? Pour de faux malheureusement mais c'est pour mieux comprendre. Mais il suffit de reprendre le jeu de 7 familles et les lapins ne s'appellent plus Karotz mais Data et le tour est joué ;)



## karotz - tutoriel pour débutant sous Windows -



Je ne sais plus si je vous ai dit qu'il fallait aussi être un peu magicien pour programmer ?

Voilà qui est fait !



Nous allons nous intéresser, dans un premier temps, à 2 attributs du rfid, le type (est-ce un flat, un nano...) et à la couleur (est-il vert, jaune....) Partant du principe que vous avez tous le flat jaune et le flat vert c'est avec ces 2 flats que nous allons « jouer ». De la même façon que nous avons testé si le bouton était appuyé une fois ou deux fois là nous allons tester si nous avons un flat et nous allons ajouter un ET, si c'est un flat est-il en plus vert ? (ou jaune)

En fait le lapin nous dira « le flat est vert » lorsque nous passerons le flat vert et « le flat est jaune » lorsque nous passerons le flat jaune.

Comment comprendre cette ligne et comment la traduire en langage de programmation ?

```
data.type : enum { NONE_TYPE = 0; FLAT = 1; NANOZTAG = 2; ZTAMPS = 3; }
```

En français on dirait est-ce que le data.type est égal à FLAT

On pourrait l'écrire en informatique if (data.type == "FLAT")

Mais on lit :

```
data.type : enum
```

Ce qui veut dire qu'on attend une valeur numérique et non une chaîne de caractères et on ajoute que pour FLAT il faut tester la valeur 1

On écrira donc

```
If (data.type == 1) (on mettrait 2 pour tester si c'est un nanoztag et 3 un ztamps.
```

Les guillemets ont disparu car c'est du numérique mais on peut les mettre quand même ça fonctionnera aussi

Pour la couleur c'est exactement la même chose on écrira

```
If (data.color == 3) 3 c'est pour le vert(green), 4 pour le jaune (yellow), 1 pour le rouge (red) etc
```

Et si on veut tester flat ET vert on écrira

```
If ((data.type == 1) && (data.color == 3))
```



Voici le main.js

```
include("util.js");
var karotz_ip = "192.168.1.46"//<= attention ici votre adresse IP
var exitFunction = function(event) {
  if ((event == "CANCELLED") || (event == "TERMINATED")) {
    exit();}return true;}

var onKarotzConnect = function(data) {
  karotz.rfid.addListener(function(data) {
    if ((data.type == "1")&&(data.color == "4"))
      karotz.tts.start("le flate est jaune", "fr", exitFunction);
    if ((data.type == "1") && (data.color == "3"))
      karotz.tts.start("le flate est vert", "fr", exitFunction);
    return true;});});
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Vous vous souvenez du passage au péage ? Vous n'aimeriez pas recevoir la facture de votre voisin ? Comment éviter cela ? En rendant unique chaque RFID, votre clé jaune est unique, bien qu'en apparence identique à celle que j'ai entre les mains et pour cela elle a un identifiant (ID) comme l'adresse MAC de votre lapin (pas l'adresse IP puisqu'on a vu qu'elle pouvait changer dans le temps)

Donc si vous avez 2 clés rfid jaunes entre les mains il va falloir trouver leur adresse RFID afin de pouvoir les différencier par programme en utilisant le paramètre

`data.id`

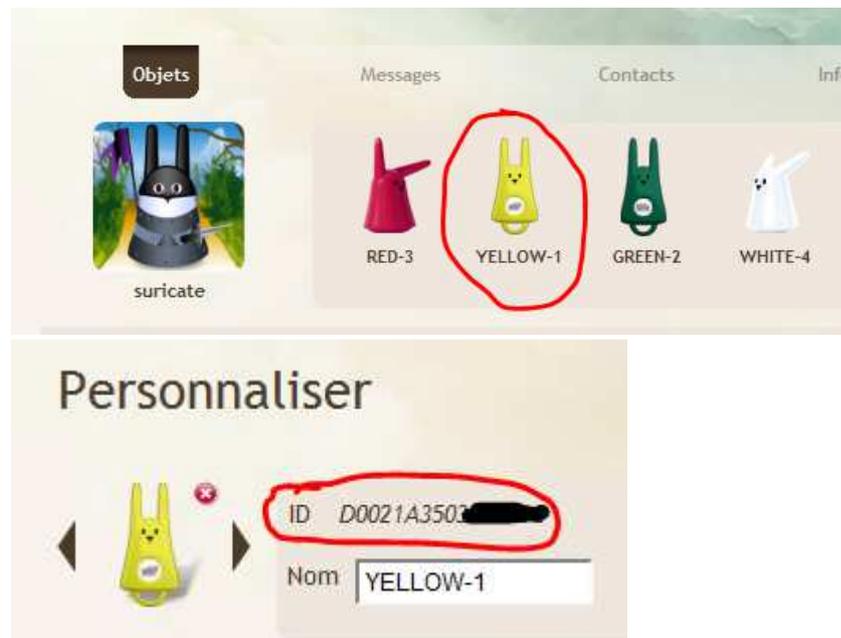
Et toujours la même question : mais comment je connais ce numéro, moi ?

Vous vous souvenez où vous avez trouvé votre adresse IP ?

C'est au même endroit. Il faut se connecter ici

<http://www.karotz.com/login>

Puis cliquer sur l'objet qui nous intéresse





Et si vous ne savez plus laquelle des 2 clés jaunes que vous avez entre les mains est la vôtre exécutez ce programme (main.js)

```
include("util.js");
var karotz_ip = "192.168.1.46"//< attention ici votre adresse IP
var exitFunction = function(event) {
  if ((event == "CANCELLED") || (event == "TERMINATED")) {
    exit();
  } return true;
}

var onKarotzConnect = function(data) {
  karotz.rfid.addListener(function(data) {
    if ((data.type == "1") && (data.color == "4") && (data.id == "D0021A3503[REDACTED]"))
      karotz.tts.start("le flate est jaune et c'est le mien", "fr", exitFunction);
    if ((data.type == "1") && (data.color == "4") && (data.id != "D0021A3503[REDACTED]"))
      karotz.tts.start("le flate est jaune mais ce n'est pas le mien", "fr",
exitFunction);
    return true;
  });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Ici c'est votre identifiant que vous devez renseigner

Vous notez au passage que si pour tester une égalité on écrit « == »  
Pour une différence on n'écrit pas « <> » mais on écrit « != » en fait plutôt  
que « différent » il faut lire « not égal » (n'est pas égal à)

Avez-vous gagné vos galons de programmeur premier échelon ?

Pour cela il va falloir passer un examen ;)

En fait il n'y a pas besoin d'aller chercher le numéro ID de vos clés sur  
Internet il suffit de la passer devant le lapin et de lui demander de nous  
lire le data.id

Pas trop dur ? A vous de jouer alors, je compte sur vous pour chercher un  
peu ;)

Ne me décevez pas ! Et attendez pour passer à la prochaine page



## Solution de l'examen de passage pour être programmeur 1<sup>er</sup> échelon :

```
include("util.js");
var karotz_ip = "192.168.1.46"//⚡ ATTENTION ici votre adresse IP !
var exitFunction = function(event) {
  if ((event == "CANCELLED") || (event == "TERMINATED")) {
    exit();
  } return true;
}

var onKarotzConnect = function(data) {
  karotz.rfid.addListener(function(data) {
    karotz.tts.start(data.id, "fr", exitFunction);
  });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

## Vous aviez trouvé? Félicitation !

Vous voici promu au grade de programmeur premier échelon





## Fakir



Ce petit exercice m'a donné des idées, pas à vous ?

En fait c'est tout simplement génial on va pouvoir demander au lapin directement les informations que nous recherchons, je ne sais pas pour vous mais les événements retournées pour l'appui sur le bouton comme « MULTIPLE », « LONG\_START », « LONG\_STOP » ou pour les clés RFID le data.direction IN ou OUT correspondent à quoi dans la vraie vie ? Alors appuyons sur le bouton et demandons au lapin de « deviner » tel un fakir et de nous dire l'information qu'il reçoit

Essayez ce petit programme (votre main.js)

```
include("util.js");
var karotz_ip = "192.168.1.29"//⚡ ATTENTION ici votre adresse IP !
var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    } return true;
}

var onKarotzConnect = function(data) {
    karotz.button.addListener(function(event) {
        karotz.tts.start(event, "fr", exitFunction);
    });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Il va nous permettre de nous dire ce que l'on fait avec le bouton, essayer l'appui simple, le double, le multiple, le longtemps appuyé...

Et j'attends vos remarques (désobligeantes) car si sur le principe ça fonctionne plutôt pas mal ce serait 100 fois mieux s'il n'y avait pas besoin de relancer à chaque fois et pour revenir à notre ID RFID, je ne sais pas pour vous mais moi je n'ai pas le temps de le noter de toute façon ce serait bien que la fenêtre reste ouverte..

Il va donc falloir remédier à cela.

On va commencer par quelque chose de très simple, demander à ce que la fenêtre noire qui s'ouvre lorsqu'on exécute notre .bat ne se referme pas car il ne vous aura pas échappé qu'il s'affiche a priori des informations intéressantes.

## karotz - tutoriel pour débutant sous Windows -

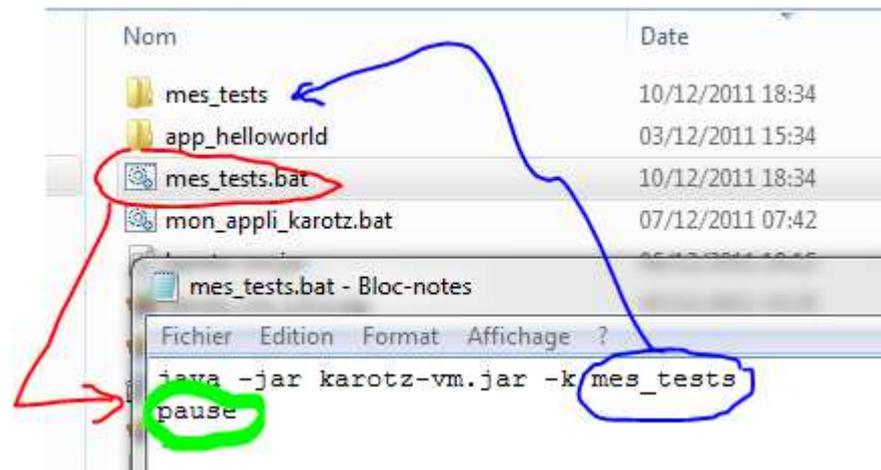


Pour ce faire on va dupliquer le fichier .bat (on pourrait modifier l'original mais si comme moi vous avez gardé l'application des « on ne dit pas » qui s'exécutent tous les 10 minutes alors il vaut mieux ne pas toucher au .bat lancé par le planificateur des tâches.

J'en entends qui se disent « mais comment peut-il avoir gardé son application en tâche de fond puisqu'on passe notre temps à modifier le main.js », c'est pas faux comme aurait dit Perceval (petite référence à la série Kamelott pour les fans dont je fais parti), j'ai en fait depuis longtemps dupliqué le répertoire app\_helloworld en mes\_tests. (pour dupliquer un répertoire ou un fichier, clic droit avec la souris sur son icône puis copier puis clic droit sur la page blanche et coller

On ouvre ensuite le nouveau fichier . bat (souvenez-vous, clic droit puis modifier), j'ai appelé le mien mes\_tests.bat

Et on le modifie comme ci-dessous (on met le bon répertoire et on ajoute pause sur la 2<sup>ème</sup> ligne



Cette ligne pause fera que la fenêtre restera ouverte

Voici ce que ça donne

```
C:\Windows\system32\cmd.exe
C:\Users\alain.bregeon\Downloads>java -jar karotz-vm.jar -k mes_tests
host: 192.168.1.29
port: 9123
karotz NOT CONNECTED
connected
karotz interactive mode
karotz should say [fr]: MULTIPLE
C:\Users\alain.bregeon\Downloads>pause
Appuyez sur une touche pour continuer...
```

Pour le data.id du rfid

```
C:\Windows\system32\cmd.exe
C:\Users\alain.bregeon\Downloads>java
host: 192.168.1.29
port: 9123
karotz NOT CONNECTED
connected
karotz interactive mode
karotz should say [fr]: D0021A35038E3
C:\Users\alain.bregeon\Downloads>paus
Appuyez sur une touche pour continuer
```

On a donc le temps de le lire



Petit programme fakir avant de revenir à des choses plus sérieuses ;)

```
include("util.js");
include("tts2.js");
var karotz_ip = "192.168.1.29"//⬅ ATTENTION : ici votre adresse IP !

var buttonListener = function(event) {
  if (event == "DOUBLE") {
    karotz.tts.stop();
    exit();
  }
  return true;
}
var exitFunction = function(event) {
  if ((event == "CANCELLED") || (event == "TERMINATED")) {
    exit();
  } return true;
}
var je_vois = "Je vois ";
var letype = "heu je ne vois pas bien le type finalement...";
var lacouleur = "la couleur est indéfinie "
var onKarotzConnect = function(data) {
  karotz.rfid.addListener(function(data) {
    if (data.type == 1){letype = "un flat ";}
    if (data.type == 2){letype = "un nanoztag ";}
    if (data.type == 3){letype = "un ztamps ";}
    je_vois = je_vois + letype;

    karotz.tts.start(je_vois, "grave", exitFunction );
  });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Ce programme permet au lapin de vous dire si vous venez de passer un flat, un nanoztag, un ztamps ou un rfid inconnu pour lui.

Bien entendu il n'est pas optimisé, 3 instructions IF qui se suivent ça n'est pas normal on imagine qu'il existe un If, Else (Si....sinon)

Par contre nous allons à présent demander à notre Karotz fakir d'annoncer la couleur et là il y a 10 possibilités on va essayer de faire autrement.

La première idée qui me vient à l'esprit c'est un tableau comme celui que nous avons utilisé pour les phrases lues aléatoirement et l'indice de ce tableau sera le chiffre correspondant à la couleur

Tableau indice 0 pour aucune couleur

Tableau indice 1 pour rouge

Etc.



Voici le main.js qui va bien

```
include("util.js");
include("tts2.js");
var karotz_ip = "192.168.1.29"//⚠ ATTENTION : ici votre adresse IP !

var buttonListener = function(event) {
  if (event == "DOUBLE") {
    karotz.tts.stop();
    exit();
  }
  return true;
}

var exitFunction = function(event) {
  if ((event == "CANCELLED") || (event == "TERMINATED")) {
    exit();
  } return true;
}

var je_vois = "Je vois ";
var letype = "heu je ne vois pas bien le type finalement...";

var lescouleurs = new Array();
lescouleurs[0] = "indaifinie";
lescouleurs[1] = "rouge";
lescouleurs[2] = "bleue";
lescouleurs[3] = "verte";
lescouleurs[4] = "jaune";
lescouleurs[5] = "rose";
lescouleurs[6] = "noir c'est noir il n'y a plus d'espoir, hum je m'aigare";
lescouleurs[7] = "grise";
lescouleurs[8] = "mandarine, clémentine non j'ai trouvez, orange";
lescouleurs[9] = "pourpre mais c'est quoi donc cette couleur?";
lescouleurs[10] = "comme celle du cheval blanc d'henri 4, je vous laisse trouver";

var onKarotzConnect = function(data) {
  karotz.rfid.addListener(function(data) {
    if (data.type == 1){letype = "flat "};
    if (data.type == 2){letype = "nanoztag "};
    if (data.type == 3){letype = "ztamps "};
    je_vois = je_vois + "une puce R F I D de type "
    je_vois = je_vois + letype
    je_vois = je_vois + " et de couleur " + lescouleurs[data.color];

    karotz.tts.start(je_vois, "grave", exitFunction );
  });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

C'est plutôt bien, je ne sais pas ce que vous en pensez ? Perfectible bien entendu mais fonctionnel.

Quoique....

J'ai entre les mains le bookz, pour ceux qui n'ont pas ça ressemble à ça



Il y a une puce RFID dans chaque « oreille » mais Je ne vois pas de

## karotz - tutoriel pour débutant sous Windows -



data.type = bookz dans le descriptif et je ne vois pas toutes les couleurs présentes dans data.color.

Alors je suis curieux de savoir quelle est la valeur du data.type qui est retournée ainsi que le code de la couleur.

Bien entendu on sait comment faire, on peut demander au lapin de nous lire l'information mais je préfère profiter de l'occasion qui m'est donnée pour vous expliquer comment « débbugger » (oh quel vilain mot, en fait essayer de comprendre pourquoi ce qui devrait fonctionner ne fonctionne pas) un programme. Il y a différentes façon pour cela, l'une consiste à « tracer » le déroulement des instructions en demandant au programme (en plus de ce qu'il a déjà à faire) de nous faire des petits coucou à des endroits stratégiques, genre « coucou je commence le programme », « coucou je suis dans la boucle qui teste le rfid », « coucou voici le rfid qui m'est envoyé », « coucou je vais enfin parler » etc.

Dans le langage de programmation que nous utilisons, car vous utilisez un langage (sans le savoir peut-être mais toutes ces instructions ne sortent pas d'un chapeau), le javascript on utilise une fonction :

log(string texte qui s'affiche) ;

Ainsi pour voir le type lu et la couleur lue il suffit d'ajouter cette ligne

```
var onKarotzConnect = function(data) {
  karotz.rfid.addListener(function(data) {
    log("le type lu est : " + data.type + " le code couleur lu est : " + data.color);
    if (data.type == 1){letype = "flat "};
```

Cette phrase apparaît comme cela dans notre fenêtre lors de l'exécution du programme

```
cmd C:\Windows\system32\cmd.exe
C:\Users\alain.bregeon\Downloads>java -jar karotz-vm.jar -k mes_tests
host: 192.168.1.29
port: 9123
karotz NOT CONNECTED
connected
karotz interactive mode
le type lu est : 5 le code couleur lu est : 13
```

Je vois donc que j'ai effectivement un type 5 (bookz) et une couleur 13, (vert foncé) qui ne sont pas documentés ! Il ne me reste plus qu'à tester toutes les couleurs du books pour noter les numéros et ainsi compléter mon programme

```
lescouleurs[11] = "bordeaux, l'abus d'alcool est mauvais pour la santez";
lescouleurs[12] = "violet foncez";
lescouleurs[13] = "vert foncez";
lescouleurs[14] = "vert clair";
lescouleurs[15] = "jaune";
lescouleurs[16] = "gris foncez";
```

```
var onKarotzConnect = function(data) {
  karotz.rfid.addListener(function(data) {
    log("le type lu est : " + data.type + " le code couleur lu est : " +
    data.color);
```

## karotz - tutoriel pour débutant sous Windows -



```
if (data.type == 1){letype = "flat "};  
if (data.type == 2){letype = "nanoztag "};  
if (data.type == 3){letype = "ztamps "};  
if (data.type == 5){letype = "bookz "};
```

ça ne vous lasse pas vous de devoir relancer le programme à chaque fois?

Moi si !





## Bouclons

Et quand ça me lasse il faut que je trouve une solution !

Idéalement il faudrait une « boucle » dans notre programme, boucle qui consisterait à attendre un évènement, ici lecture d'un RFID, mais en même temps comment sortir de cette boucle car il faut bien aussi que ça se termine à un moment donné.

Intéressons-nous de plus près à cette ligne

```
karotz.tts.start(je_vois, "fr", exitfunction );
```

Cette instruction, celle qui lit à voix haute enchaîne sur « exitfunction »

```
var exitFunction = function(event) {  
    if((event == "CANCELLED") || (event == "TERMINATED")) {  
        exit();  
    }  
    return true;  
}
```

En bon français cela se lit comme ceci (les 2 fonctions)

*Lit la phrase et va voir ce qu'il se passe*

*Si c'est interrompu (cancelled) ou terminé (terminated) alors tu*

*sors de tout ça (exit) sinon reviens (return) à la lecture de la phrase*

Il « suffit » donc de changer légèrement cette phrase en supprimant le exit pour que le programme ne sorte plus via cet fonction. A la place on réinitialise la variable « je\_vois » pour se préparer à la prochaine phrase

*Si c'est interrompu (cancelled) ou terminé (terminated) alors tu*

*initialises je\_vois pour a prochaine lecture sinon reviens (return) à*

*la lecture de la phrase actuelle*

Traduction en programme

```
var exitFunction = function(event) {  
    if ((event == "CANCELLED") || (event == "TERMINATED")) {  
        je_vois = "Je vois ";  
    } return true;  
}
```

Mais alors me direz-vous comment sort-on du programme ?

Il y a cette fonction qui semble dire qu'on sortira du programme en appuyant 2 fois sur le bouton

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

On essaie alors !

Ben oui mais on a beau s'agacer sur le bouton il ne se passe rien !

Une idée ?

Ça n'est pas si loin que ça pourtant, ça remonte au début de ce tutoriel, le premier exercice que nous avons fait concerne justement le test du bouton ;)

## karotz - tutoriel pour débutant sous Windows -



Il faut ajouter la ligne en jaune dans la fonction principale

```
var onKarotzConnect = function(data) {  
    karotz.button.addListener(buttonListener);  
    karotz.rfid.addListener(function(data) {
```

Voici le main.js dans son intégralité. Vous pouvez être fier de vous vous avez fait du bon travail.

```
include("util.js");  
include("tts2.js");  
var karotz_ip = "192.168.1.46"//<= ATTENTION : ici votre adresse IP !  
  
var buttonListener = function(event) {  
    if (event == "DOUBLE") {  
        karotz.tts.stop();  
        exit();  
    }  
    return true;  
}  
  
var exitFunction = function(event) {  
    if ((event == "CANCELLED") || (event == "TERMINATED")) {  
        je_vois = "Je vois ";  
    } return true;  
}  
  
var je_vois = "Je vois ";  
var letype = "heu je ne vois pas bien le type finalement...";  
  
var lescouleurs = new Array();  
lescouleurs[0] = "indaifinie";  
lescouleurs[1] = "rouge";  
lescouleurs[2] = "bleue";  
lescouleurs[3] = "verte";  
lescouleurs[4] = "jaune";  
lescouleurs[5] = "rose";  
lescouleurs[6] = "noir c'est noir il n'y a plus d'espoir, hum je m'aigare";  
lescouleurs[7] = "grise";  
lescouleurs[8] = "mandarine, clémentine non j'ai trouvez, orange";  
lescouleurs[9] = "pourpre mais c'est quoi donc cette couleur?";  
lescouleurs[10] = "comme celle du cheval blanc d'henri 4, je vous laisse trouver";  
lescouleurs[11] = "bordeaux, l'abus d'alcool est mauvais pour la santez";  
lescouleurs[12] = "violet foncez";  
lescouleurs[13] = "vert foncez";  
lescouleurs[14] = "vert clair";  
lescouleurs[15] = "jaune";  
lescouleurs[16] = "gris foncez";  
  
var onKarotzConnect = function(data) {  
    karotz.button.addListener(buttonListener);  
    karotz.rfid.addListener(function(data) {  
        if (data.type == 1){letype = "flote ";}  
        if (data.type == 2){letype = "nanoztag ";}  
        if (data.type == 3){letype = "ztamps ";}  
        if (data.type == 5){letype = "bookz ";}  
        je_vois = je_vois + "une puce R F I D de type "  
        je_vois = je_vois + letype  
        je_vois = je_vois + " et de couleur " + lescouleurs[data.color];  
  
        karotz.tts.start(je_vois, "fr", exitFunction );  
    }  
});  
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```



## Répète après moi ?

### Il a de grandes oreilles mais est-ce qu'il nous entend ?

Un autre moyen de « commander » notre lapin est tout simplement notre parole, si on lui disait (gentiment) ce que nous voulons qu'il fasse, le ferait-il ?

Voilà un nouveau challenge intéressant, j'ai l'impression que vous êtes impatient.

Mais vous allez devoir patienter un peu, cela sera le premier exercice de notre prochain tutoriel.

