Outils pour construire un code jQuery évolutif.

par Daniel Hagnoul (Mon cahier d'exercices sur jQuery & Co)

Date de publication : 3 septembre 2009

Dernière mise à jour :

Avec un JavaScript omniprésent et des codes de plus en plus complexes, le programmeur se doit de maîtriser les bases du langage, la structuration et la modularisation du code, il se doit d'écrire un code gérable, réutilisable et facilement évolutif.

Dans cet esprit, je vous propose de généraliser l'usage de la clôture jQuery (closure jQuery).

L'utilisation de la clôture implique celle de la fonction globale et d'un espace de noms.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

I - La clôture jQuery

I-A - Introduction

Non, il ne s'agit pas de clôturer le compte bancaire de jQuery ! Il s'agit simplement de protéger votre code.

En anglais on parle de "closure" et ce mot peut se traduire par fermeture, mais aussi par clôture que je trouve plus parlant.

I-B - C'est quoi ?

Il s'agit d'une fonction anonyme function(){...}

entourée d'une clôture (function(){...})();

et dédiée à jQuery : (function($){...})(jQuery);

I-C - Un espace privé

La clôture (function(){...})(); est un espace privé. Elle s'exécutera automatiquement.

La clôture jQuery (function($){...})(jQuery); est un espace privé, dans lequel le symbole $ est clairement et exclusivement attribué à jQuery. Elle s'exécutera automatiquement.

I-D - Plus de conflits pour le $

Si vous utilisez une autre bibliothèque ("framework"), la clôture vous assure qu'il n'y aura pas de conflit pour la maîtrise du symbole $.

Exemple n° 1

```
(function($){
 //jQuery
 $("#conteneur").css("backgroundColor","#FFCC66");
})(jQuery);

//Prototype
$('affiche').addClassName('active').show();

//jQuery
//erreur : $("#conteneur") is null !
$("#conteneur").css("color","red");
```

I-E - Objets, fonctions et variables locales

Tout ce qui est déclaré à l'intérieur de la clôture est inconnu en dehors à la seule condition que vous utilisiez le mot clé "var" comme il se doit.

(1) L'utilisation de l'espace de noms de l'objet global "window" par votre code est source d'inexactitude, d'instabilité et d'insécurité.
Ce qui était tolérable à l'époque de scripts de quelques lignes est rigoureusement à proscrire à l'époque du "web 2.0" où nous mélangeons des centaines de lignes de scripts d'auteurs différents. Le risque de collisions dans l'espace de noms de l'objet global "window" devient une certitude.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

La qualité de votre code est inversement proportionnelle aux nombres d'objets, de fonctions et de variables que vous avez introduits dans l'espace de noms de l'objet global "window".
Idéalement votre code ne doit dépendre que d'un seul objet global.

Exemple n° 2

```
(function($){
 a = "je suis a";
 var b = "je suis b";

 function mafunc(message){
  alert(message);
 }
})(jQuery);

alert(a);

//erreur : "b is not defined !"
alert(b);

//erreur : "mafunc() is not defined !"
mafunc("Hello world !");
```

I-F - L'opérateur this

Dès la clôture créée, le mot clé "this" représentant l'objet "window" sera disponible tant que vous ne modifierez pas sa valeur.

(2) *Je cite[2] :*
"Le mot-clé this fait référence à l'objet de contexte (ou objet courant). En général, dans une méthode, this fait référence à l'objet appelant."
"Du fait du « passage » de this aux fonctions, this n'a pas de valeur fixe pour une fonction. Cela signifie qu'une fonction n'a pas de « propriétaire » ou de « parent », même s'il s'agit d'une méthode. En d'autres mots, une méthode n'est pas liée à l'objet dont elle est une méthode."
"La distinction entre méthodes et fonctions se fait uniquement au moment de l'appel : les appels de méthode passent l'« objet parent » comme valeur de this, tandis que les appels de fonctions passent l'objet global en tant que this."
Fin de la citation[2].

Exemple n° 3

```
(function($){
 //équivalent à window.confirm()
 var r = this.confirm("Bienvenue dans mon cahier d'excercices sur jQuery & Co.");

 if (r == true){
  //équivalent à window.alert()
  this.alert("You pressed OK!")
 } else {
  this.alert("You pressed Cancel!")
 }

 //Une fonction anonyme entourée d'une clôture. Elle s'exécutera automatiquement.
 (function(){
  this.alert("La valeur de this n'ayant pas été modifiée, sa valeur est : " + this);
 })();

 //http://james.padolsey.com/javascript/jquery-delay-plugin/
 $.fn.delay = function(time, callback){
  jQuery.fx.step.delay = function(){};
```


Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

```
  return this.animate({delay:1}, time, callback);
 }

 //On peut conserver la valeur actuelle de this en l'affectant à une variable, par exemple that
 var that = this;

 //équivalent à $(window).ready()
 $(this).ready(function(){

  alert("Dans $(window).ready(), that = " + that + ", et this = " + this);

  $("#conteneur h1").delay(2000, function(){
   $(this).css("color","#339900");

   alert("Dans cette fonction anonyme, that = " + that + ", et this = " + this);
  });
 });
})(jQuery);
```

Avec Internet Explorer 8 vous devrez actualiser la page pour obtenir l'affichage de toutes les boites de dialogues.

Avec Firefox, voici le contenu des trois dernières boites de dialogue :

- La valeur de this n'ayant pas été modifiée, sa valeur est : [object Window]
- Dans $(window).ready(), that = [object Window], et this = [object HTMLDocument]
- Dans cette fonction anonyme, that = [object Window], et this = [object HTMLHeadingElement]

I-G - Mais alors, comment ?

Je vous entends : "Si j'adopte son système, si je clôture tous mes programmes, comment appeler mon code ?"

Il faut conserver, au niveau global, un point d'entrée qui sera un objet ou une fonction. Mais pas n'importe comment.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

II - La fonction globale

II-A - Exemple d'une "news box"

II-A-1 - Description

Il s'agit de la version adaptée en jQuery d'une "news box" proposée par **MARCHA** en réponse à cette **question**.

"News box" : diffusion verticale continue, de bas en haut, de messages (brefs, en principe) attirant l'attention sur de nouvelles informations. La courte pause avant la reprise de la diffusion indique au lecteur que la série est terminée.

Exemple n° 4 : "news box" sans fonction globale

```
(function($){
 var speed = 1;
 var offset = 5;
 var interval = 60;
 var pos;
 var pos_initial;

 function anim() {
  $("#newslist").css({
   visibility:"visible",
   top:Math.floor(pos)
  });

  pos -= speed;

  if(pos < (-1 * $("#newslist").height())) {
   pos = pos_initial;
  }
 }

 function startAnim() {
  pos_initial = $("#newsbox").height() + offset;
  pos = pos_initial;
  setInterval("anim()", interval);
 }

 $(this).ready(function(){
  $("#newsbox").hover(
   function(){
    speed = 0;
   },
   function(){
    speed = 1;
   }
  );

  startAnim()
 });
})(jQuery);
```

II-A-2 - Bug !

Avec Firefox et l'extension Firebug, le programme s'interrompt sur l'erreur : "anim is not defined", car la fonction anim() est inaccessible.

Alors qu'en absence de clôture le programme donne entière satisfaction.

Je vous entends : "On nous agace avec la clôture et dès le premier véritable exemple rien ne va plus !"

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

II-A-3 - Pourquoi ?

La source de l'erreur est :
```
setInterval("anim()", interval);
```

Le problème vient de cette ligne de code qui demande à l'objet global "window" d'appeler la fonction anim() toutes les 60 millisecondes.

Or la fonction anim(), protégée par la clôture, est inaccessible pour l'objet global "window".

II-A-4 - Ouvrez la porte !

Hé oui ! Nous avons besoin d'un point d'entrée dans la clôture.

(3) Je cite[3] :
"Je vous rappelle qu'en JavaScript tout fonctionne en se fondant sur des objets. (...) Le langage JavaScript permet de créer simplement un objet en se fondant sur l'objet Object ou en utilisant une forme littérale dont la syntaxe est décrite par la notation JSON. (...) Il est possible d'ajouter, de modifier et de supprimer les entrées de l'objet tout au long de sa vie."

Extrait de la page 5 de la citation 3
```
var obj = new Object();

var obj = { ... };
// avec la notation JSON

obj["attribut"] = "valeur1";
// similaire à obj.attribut = "valeur1";

obj["methode"] = function(param1, param2) {
  ...
};
// similaire à obj.methode = function( ... ){ ... };
```

Fin de la citation[3].

II-B - Exemple d'une "news box" utilisant une fonction globale

II-B-1 - Fonction globale et objet global

Nous pouvons donc écrire la fonction anim() de la manière suivante :

```
var anim = function() { ... };
```

Mais nous devons omettre le mot clé "var" pour rendre anim() accessible depuis l'extérieur de la clôture.

C'est cette construction :

```
anim = function() { ... };
```

que j'appelle une fonction globale.

Un fragment de code existant rarement seul, je me servirai de la fonction globale ou de l'objet global :

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

```
monObjet = { ... };
```

comme point d'entrée permettant de communiquer avec la clôture.

II-B-2 - Exemple

Exemple n° 5 : "news box" utilisant une fonction globale

```
(function($){
 var speed = 1;
 var offset = 5;
 var interval = 60;
 var pos;
 var pos_initial;

 anim = function() {
  $("#newslist").css({
   visibility:"visible",
   top:Math.floor(pos)
  });

  pos -= speed;

  if(pos < (-1 * $("#newslist").height())) {
   pos = pos_initial;
  }
 };

 function startAnim() {
  pos_initial = $("#newsbox").height() + offset;
  pos = pos_initial;
  setInterval("anim()", interval);
 }

 $(this).ready(function(){
  $("#newsbox").hover(
   function(){
    speed = 0;
   },
   function(){
    speed = 1;
   }
  );

  startAnim()
 });
})(jQuery);
```

Attention, dans la fonction

```
window.setInterval(expression, millisecondes);
```

expression peut s'écrire sous trois formes :

1. un texte :

```
window.setInterval("anim()", 60);
```

2. le nom de la fonction :

```
window.setInterval(anim, 60);
```


Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

3. une fonction anonyme :

```
window.setInterval(function(){anim();}, 60);
```

La forme 1 est la seule qui impose l'usage d'une fonction globale, car l'objet "window" convertit l'expression "anim()" en une recherche de la fonction window.anim() qu'il ne peut trouver lorsqu'elle est placée dans un espace privé.

Ce qui précède s'applique également à la fonction :

```
window.setTimeout(expression, millisecondes)
```

II-C - Pollution de l'espace de noms

En choisissant de promouvoir la clôture, espace privé, je souhaitais également libérer l'espace de noms "window", or chaque fonction globale ou objet global encombrera un peu plus cet espace de noms. De plus si comme il se doit, votre code est un composant réutilisable, le risque de collisions avec un autre composant du même nom est très élevé. Il est clair que l'utilisation de la clôture implique l'usage d'un espace de noms privé.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

III - Un espace de noms ("namespacing")

III-A - Rappel

Nous avons vu que la clôture offre un espace privé où le symbole $ est exclusivement attribué à jQuery.

Nous avons vu que la fonction globale permet d'obtenir un point d'entrée dans la clôture jQuery mais aux prix d'une pollution de l'espace de noms.

III-B - Introduction

(4) Je cite[4] :
"Un espace de noms est une notion permettant de lever une ambiguïté sur des termes qui pourraient être homonymes sans cela. Il est matérialisé par un préfixe identifiant de manière unique la signification d'un terme. Au sein d'un même espace de noms, il n'y a pas d'homonymes. Les espaces de noms aident à la construction de programmes modulaires".
Fin de la citation[4]

(5) Je vous invite à lire, au minimum, les pages 3 à 6 de l'excellent tutoriel de **Nourdine Falola : Espaces de noms (ou namespace) en JavaScript**. Ces pages constituent l'introduction indispensable à la suite de cet article.

Idéalement, votre code ne doit être accessible qu'au travers d'un et d'un seul objet global.

III-C - Comment ?

Comme notre espace de noms contiendra du code jQuery, nous devons utiliser la clôture jQuery.

La première solution qui vient à l'esprit est de construire un objet global dans l'espace de noms window à l'intérieur d'une clôture jQuery.

On peut également construire un objet global dans l'espace de noms jQuery :

```
(fonction($){
 $.monObjet = { ... } ;
})(jQuery);
```

à l'intérieur d'une clôture jQuery.

III-D - Solution

Après avoir testé ces deux approches, une troisième solution a ma préférence.

Il s'agit toujours de construire un objet global dans l'espace de noms window, mais c'est la clôture jQuery qui génère l'objet grâce à l'écriture JSON.

Suivant votre degré d'expérience dans l'écriture de code JSON cette solution peut vous paraître complexe, mais il me semble qu'elle est d'un usage plus simple que les deux approches évoquées ci-dessus. De même, la gestion des exceptions et la création des sous-espaces de noms me semblent plus faciles et plus fiables.

JavaScript est très efficace pour résoudre une expression d'appel complexe, vous ne devez pas craindre la très légère diminution des performances qui en résulte, car elle est compensée largement par l'aspect auto documentaire et la fiabilité de votre code.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

III-E - Espace de noms dvjh

En notation JSON, l'objet global dvjh peut s'écrire comme suit :

```
try {
  var dvjh = {
    isdvjh: "dvjh/danielhagnoul",
    isMe: function(){
      return this.isdvjh;
    },
    infos: function(){
      return "Espace de noms dvjh.";
    }
  };
}
catch(err){
  alert(err);
}
```

Avec la génération par la clôture jQuery :

```
try {
  var dvjh = (function($) {
    //notre clôture peut toujours contenir variables,
    //fonctions et objets privés.

    var isdvjh = "dvjh/danielhagnoul";

    return {
      isMe: function(){
        return isdvjh;
      },
      infos: function(){
        return "Espace de noms dvjh";
      }
    };
  })(jQuery);
}
catch(err){
  alert(err);
}
```

Et le sous-espace de noms dvjh.math :

```
try {
  dvjh.math = (function($) {
    var isdvjh = "dvjh_math/danielhagnoul";

    if (dvjh.isMe() != "dvjh/danielhagnoul/") return null;

    return {
      isMe: function(){
        return isdvjh;
      },
      infos: function(){
        return "Espace de noms dvjh.math";
      }
    };
  })(jQuery);
}
catch(err){
  alert(err);
}
```

La gestion des exceptions et le test sur isMe nous assurent de l'existence d'un objet dvjh et que cet objet dvjh est bien le nôtre.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

III-F - Composition de l'espace de noms dvjh le 9 août 2009

L'espace de noms dvjh doit être vu comme un "proof of concept" (une démonstration de faisabilité), il vous montre comment réaliser un espace de noms en jQuery, il ne s'agit pas d'un espace de noms de référence.

J'ai toutefois pris soin de le remplir soit avec des codes utiles pour jQuery soit avec des exemples de code intéressants.

Ci-dessous, le contenu de l'interface de l'espace de noms dvjh et de ses sous-espaces.

III-F-1 - dvjh

Exception type dvjh : dvjh.exception(message, args);
alert(contenu de l'exception) : dvjh.displayException(exc);
Séparateur de texte : dvjh.separateur(s);
Interface de dvjh : dvjh.infos();

Le fichier dvjh.js contient également des fonctions utiles pour jQuery :

(6) **jQuery delay plugin de James Padlosey**

(7) **Regex Selector for jQuery de James Padlosey**

et le selecteur contains dans une clôture annexe.

(8) Ainsi que **Function.prototype.memoize de Keith Gaughan** qui est utilisé dans dvjh.math et dvjh.math.statistic.

(9) Je cite[9] :
"La mémoization est une technique d'optimisation de code consistant à réduire le temps d'exécution d'une fonction en mémorisant ses résultats d'une fois sur l'autre. Bien que liée à la notion de cache, la mémoization désigne une technique bien distincte de celles mises en oeuvre dans les algorithmes de gestion de la mémoire cache."
"Une fonction mémoizée stocke les résultats de ses appels précédents dans une table et, lorsqu'elle est appelée à nouveau avec les mêmes paramètres, renvoie la valeur stockée au lieu de la recalculer."
"Une fonction peut être mémoizée seulement si elle est référentiellement transparente, c'est-à-dire si sa valeur de retour ne dépend que de la valeur de ses arguments."
"La mémoization est une façon de diminuer le temps de calcul d'une fonction, au prix d'une occupation mémoire plus importante."
Fin de la citation[9]

III-F-2 - dvjh.math

Modulo informatique : dvjh.modulo(a, b);
Nombre de Fibonacci : dvjh.fib(n);
Nombre d'or à la puissance n : dvjh.phi(n);
Interface de dvjh.math : dvjh.math.infos();

dvjh.math montre comment créer un sous-espace de noms. En dehors du **modulo informatique**, dvjh.math donne un exemple d'application de la mémoization pour le calcul du Nombre de Fibonacci.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

III-F-3 - dvjh.math.statistic

Factorielle de n : dvjh.math.statistic.fact(n);
Nombre d'arrangements à k éléments : dvjh.math.statistic.arran(n,k);
Nombre de combinaisons à k éléments : dvjh.math.statistic.combi(n,k);
Interface de dvjh.math.statistic : dvjh.math.statistic.infos();

dvjh.math.statistic montre comment créer un sous-sous-espace de noms et donne un exemple d'application de la mémoization pour le calcul de la Factorielle.

III-F-4 - dvjh.calendar

Date vers jour julien : dvjh.calendar.date_to_jj(an,mois,jours,heures,minutes,secondes);
Jour julien vers date : dvjh.calendar.jj_to_date(jj);
Jour de la semaine : dvjh.calendar.joursemaine(jj);
Jour julien vers jour julien modifié : dvjh.calendar.jj_to_mjd(jj);
Jour julien modifié vers jour julien : dvjh.calendar.mjd_to_jj(mjd);
Interface de dvjh.calendar : dvjh.calendar.infos();

Les astronomes et les historiens amateurs ne se poseront pas la question de l'utilité des Jours Juliens. Pour les autres, je signale que la manipulation de dates par l'intermédiaire des JJ est beaucoup plus simple qu'il n'y paraît.

(10) Je cite[10]:
"La période julienne est une numérotation continue et décimale des jours, indépendante du calendrier, de l'année, du mois, du jour de la semaine et du numéro du jour dans le mois."
"Elle a pour origine le 1er janvier de l'an 4713 avant notre ère (c'est-à-dire le 1er janvier - 4712 en notation des astronomes)."
"La continuité des jours permet de calculer un intervalle de temps sans risque d'erreur même s'il couvre à la fois le calendriers julien et grégorien. Les formules de calcul tiennent compte du changement de calendrier qui a eu lieu en 1582."
"Exemple : le 1er janvier 1001 (calendrier julien) à 0 h correspond au jour julien 2 086 673,5 ; le 1er janvier 2001 (1000 ans plus tard, mais en calendrier grégorien) à 0 h correspond au jour julien 2 451 910,5. Entre ces deux dates il s'est donc écoulé 365 237 jours. Remarquez qu'il ne s'est pas écoulé 365 250 comme on pourrait le déduire un peu rapidement. Il manque les 10 jours dûs à l'introduction du calendrier grégorien et les 3 jours bissextiles qui n'ont pas été ajoutés en 1700, 1800 et 1900."
Fin de la citation[10] .

III-F-5 - dvjh.effects

Animer en suivant l'ordre : dvjh.effects.byOrder(jQueryObjets, params, duration, easing, callback);
Interface de dvjh.effects : dvjh.effects.infos();

```
dvjh.effects.byOrder($('.image_menu_on'), {width:400,
height:300}, 1000, 'easeInSine', function(){alert('wow!')} );
```

La fonction byOrder() applique à tour de rôle la même animation à chaque objet jQuery contenu dans l'array jQueryObjets.

Le code de la fonction byOrder() n'est pas de moi, je sais l'avoir vu et copié sur l'internet mais j'ai perdu la référence. Mille excuses à l'auteur !

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

III-F-6 - dvjh.utilities

Regroupement de fonctions globales.
Interface de dvjh.utilities : dvjh.utilities.infos();

Sers de point d'ancrage pour les fonctions globales (objets globaux).

III-F-7 - dvjh.utilities.contentTable()

```
dvjh.utilities.contentTable('conteneur', 'tdm', 6, {tdmTitre:'Sommaire'});
```

(11) Cette fonction globale (**idée et code originaux** de **Giminik**, version jQuery de Daniel Hagnoul) permet de construire une table des matières.

III-G - Fichiers

Le code de l'espace de noms dvjh et de ses sous-espaces est trop long pour être reproduit ici.

Je vous invite à copier et examiner les fichiers composant l'espace de noms et à regarder les tests.

Mirroir 1 : dvjh-v1.0.2.zip

Mirroir 2 : dvjh-v1.0.2.zip

Test n° 1 : dvjh.math et dvjh.statistic

Test n° 2 : dvjh.calendar

Test n° 3 : dvjh.effects

Test n° 4 : dvjh.utilities.contentTable()

III-H - Nota bene

Lorsque vous construirez votre propre espace de noms, pensez à lui donner un nom suffisamment long pour que le risque de collision avec un autre espace de noms soit réduit, il eût été préférable de nommer mon espace de noms danielhagnoul plutôt que dvjh, mais on se défait difficilement d'une mauvaise habitude.

IV - Copyright

Les "citations de ..." et les "codes attribués à ..." ont leurs propres copyrights. Seuls les textes et les codes de l'auteur ont le copyright ci-dessous.

V - Remerciements

Je remercie **Monsieur Didier Mouronval** (**Bovino**) pour son travail de relecture, ses bons conseils et ses suggestions pertinentes.

Je remercie très sincèrement Messieurs **Didier Mouronval** (**Bovino**) et **Rodrigue Hunel** (**Kerod**), sans leurs gentillesses, leurs efficacités et leurs soutiens ce tutoriel et le site "Mon cahier d'exercices sur jQuery & Co" n'existeraient tout simplement pas.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

VI - Excuses

N'ayant plus rien écrit depuis très longtemps, j'avais l'angoisse de la page blanche, une vraie difficulté à traduire mes codes en mots, en phrases, en un écrit plus ou moins valable. À tel point que j'avais proposé, avec insistance, à **Monsieur Didier Mouronval** d'écrire ce tutoriel avec moi. Mais par manque de temps il n'a pu s'y mettre avant que je me décide à affronter l'ouvrage.

Je présente donc toutes mes excuses à **Monsieur Didier Mouronval** pour ne pas l'avoir attendu.

Il voudra peut-être bien comprendre qu'une fois le système d'édition XML assimilé, assez rapidement ma foi, je me suis lancé et le principal était terminé en 3 jours.

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"

1 : **Global Domination by Douglas Crockford (http://yuiblog.com/blog/2006/06/01/global-domination/)**
2 : **L'opérateur this**
3 : **Programmation orientée objet avec le langage JavasScript (1ère partie) par Thierry Templier.**
4 : **Espace de noms (http://fr.wikipedia.org/wiki/NameSpace#Langages_de_programmation)**
5 : **Espaces de noms (ou namespace) en JavaScript par Nourdine FALOLA**
6 : **jQuery delay plugin de James Padlosey (http://james.padolsey.com/javascript/jquery-delay-plugin/)**
7 : **Regex Selector for jQuery de James Padlosey (http://james.padolsey.com/javascript/regex-selector-for-jquery/)**
8 : **Function.prototype.memoize par Keith Gaughan (http://talideon.com/weblog/2005/07/javascript-memoization.cfm)**
9 : **Mémoization (http://fr.wikipedia.org/wiki/Memoization)**
10 : **PERIODE JULIENNE, JOUR JULIEN et MJD par Jean-Paul Cornec (http://pagesperso-orange.fr/jean-paul.cornec/MJD.htm)**
11 : **GÉNÉRER UNE TABLE DES MATIÈRES À L'AIDE DU DOCUMENT OBJECT MODEL ET JAVASCRIPT par Matthieu PETIOT**

Creative Commons License : Attribution-Share Alike 2.0 Belgium Mention obligatoire : "Auteur : Hagnoul Daniel (http://www.developpez.net/forums/u285162/danielhagnoul/)"