Namespaces in JavaScript

par Nourdine FALOLA (Opinions, questions)

Date de publication : April, 22th 2009

Dernière mise à jour :

This article is aimed at a population of programers already familiar with web development and JavaScript. In this article you'll learn how to pack your libraries in JavaScript by simulating what we call a namespace.

I - What's a namespace..... 3
I-A - The problem..... 3
I-B - How to pull through..... 3
I-C - How to choose to put a code in a namespace and another one in another namespace..... 3
II - Creation of a namespace in JavaScript..... 3
II-A - Basis..... 3
II-B - JSON notation..... 5
II-C - Existence and unicity of a namespace..... 5
II-D - Les sous-espaces de noms..... 7
II-E - Les limites des espaces de noms en JavaScript..... 8
III - Exemples..... 10
III-A - Votre bibliothèque : myOwnJSLibrary.js..... 10
III-B - myOwnJSLibrary.Math.js..... 11
III-C - myOwnJSLibrary.Math.Statistic.js..... 12
III-D - myOwnJSLibrary.String.js..... 13
III-E - Utilisation dans un code client : test.html..... 14
IV - Annexe..... 14
IV-A - Le mot-clé "this"..... 14
IV-B - Les fonctions anonymes..... 15
IV-C - Importation de symboles..... 16
V - Remerciements..... 17

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

I - What's a namespace

Let us briefly recall what a namespace is (or a short introduction for those who don't know).

I-A - The problem

When you program, alone or in team, it may happen that you find in many source codes variables, functions, classes, etc.. with the same name but performing different tasks as provided for different contexts. If you need to work these codes within the same program, there'll be a conflict of names. With a compiled language, the compiler'll warn you about this conflict, but with an interpreted language like JavaScript you'll miss it and what you'll handle will be the last definition of your variables, functions, etc.. which have crushed the previous definitions.

I-B - How to pull through

Save us, doc! Well here is the usefulness of namespaces. These serve to bring your sources in a same area and to qualify them in a unique way (if you don't play to create two namespaces with the same names of course).

I-C - How to choose to put a code in a namespace and another one in another namespace

It's better to group under a single namespace functionalities that act together in the same context. For example, it's inappropriate to put in the same namespace features for mathematical treatment and containers. In this case, it's wise to create 2 namespaces : one for numbers ("Math" for example) and one for strings ("String" for example). In these two namespaces you can have an "Add". For "Math", it can add two numbers, for "String", it can concatenate the contents of a string to another (addition of strings), two actions completely differents which have the same name.

I hope that you're convinced of the usefulness of namespaces. Now, let see how to implement them in JavaScript.

II - Creation of a namespace in JavaScript

Will I shock you and make you ask what is the purpose of this article if I tell you that namespaces don't exist in JavaScript : No, they don't exist as such! But it's possible to simulate them.

In JavaScript, you can create objects and give them attributes (or properties) as you want, then what will be a namespace ? An object whose properties will be the functionalities you want to pack in a same domain.

II-A - Basis

Let's create a namespace, i.e. an object, myNamespace :

Creation of a namespace

```
var myNamespace = {} ;
```

Let's add some functionalities :

Addition of functionalities to a namespace

```
myNamespace.var1 = "NF";
myNamespace.var2 = 28;
myNamespace.displayVar1 = function() { alert(this.var1) ; } ;
myNamespace.displayVar2 = function() { alert(this.var2) ; } ;
myNamespace.createObject = function(){
  return new function()
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Addition of functionalities to a namespace

```
  {
    this.name = "no name";
    this.what = "object created by myNamespace.createObject";
    this.version = "v1.0";
    this.display = function() {
      alert(this.name+ " ("+this.what+") - "+ this.version);
    }
  }
};
```

You'll find in appendix a short reminder about the use of the keyword this.

And let's test our creation :

Use of functionalities of a namespace

```
alert(myNamespace.var1);    // NF
myNamespace.var1 = "NF is the author";
myNamespace.displayVar1(); // NF is the author

alert(myNamespace.var2);    // 28
myNamespace.var2++;
myNamespace.displayVar2(); // 29

var o = myNamespace.createObject();
o.version = "v2.0";
o.display();
// no name (object created by myNamespace.createObject) - v2.0
```

Some explanations are essential.

First, we created an object called myNamespace that we'll suppose unique (you must do that it's the case).

Then, we have added to it some functionalities which constitute our library functionalities :

- A variable var1 which contains a string
- A variable var2 which contains a number
- A function which displays var1
- A function which displays var2
- A function which creates an object with 4 properties (3 variables and a display function of its variables)

Finally, we use our library functionalities :

- Display var1
- Modify the value of var1
- Display var1 (new value)
- Display var2
- Modify the value of var2
- Display var2 (new value)
- Create an object o
- Modify the value of o.version
- Display the values of o's properties

At this step, you have packed up your functionalities (i.e. your library), you have solved potential conflicts of naming by bringing together these functionalities in a same namespace (or domain) and, if the choice of the namespace is pertinent, we recognize immediatly the use of your library.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

II-B - JSON notation

Maybe the previous code is not readable for you, not quite intuitive. Perhaps you used to use a different syntax because of your experience and you discover it with difficulty. Note that this is not the only possible syntax.

You can, for instance, use the JSON notation :

JSON notation

```
var myNamespace = {
  var1 : "NF",
  var2 : 28,
  displayVar1 : function() { alert(this.var1) ; },
  displayVar2 : function() { alert(this.var2) ; },
  createObject : function(){
    return {
      name : "no name",
      what : "object created by myNamespace.createObject",
      version : "v1.0",
      display : function() {
        alert(this.name+ " ("+this.what+") - "+ this.version);
      }
    }
  }
};
```

This notation is strictly equivalent to the previous one, neater because more readable and showing immediatly that the functionalities of your library are owned by the same set myNamespace.

However, nothing prevents you from mixing the 2 notations. For example, if you have implemented your library in a file and in a second you want to add new ones. In the first file, write the previous code and in the second one :

Extension of a namespace

```
myNamespace.newFunction = function() {
  alert("new function added") ;
};
```

II-C - Existence and unicity of a namespace

If you need to split into several files the implementation of your namespace as in the previous section, I advise you to test its existence at the beginning of each add-on file, because you have no guarantee that users of your library include all the necessary scripts into their web pages, or declare them in the right order.

Test of existence of the name of the variable

```
if (typeof myNamespace == "undefined")
{
  alert("myNamespace is not defined. Include myNamespace.js before this script");
}
else
{
  myNamespace.newFunction = function() {
    alert("new function added") ;
  };
}
```

Since the beginning of this article, I tell you that namespaces allow us to avoid name conflicts. "Interesting" you can say, but if we pack our code in a namespace what prevents :

- that several namespaces can't have the same name ?

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- that an object or a variable can't have the same name as our namespace ?

Answer : nothing, back to the beginning.

En fait pas tout à fait, car déjà nous avons évité qu'au sein de vos propres développements vous ayez des conflits de noms et en travail collaboratif nous avons limité la possibilité de conflits aux noms des espaces. C'est déjà pas mal. Comment atteindre l'étape ultime et éviter tout conflit entre votre espace et un objet ou une variable ? Impossible. Vous ne pouvez garantir que ceux qui utiliseront votre espace de noms feront attention à ne pas créer autre chose portant le même nom.

Cependant vous pouvez mettre en place un système pour détecter si à la fin du chargement d'une page web utilisant votre espace celui-ci n'a pas été écrasé par une autre variable de même nom. Vous pouvez aussi étoffer le test du listing précédent pour non seulement tester l'existence du nom de votre espace, mais aussi vérifier qu'il s'agit toujours de votre espace.

Pour cela ajoutons une propriété author à notre espace dans laquelle vous mettez votre nom. Cette propriété nous sert à identifier l'espace de noms :

Identifier l'espace de noms

```
var myNamespace = {
  author : "Nourdine FALOLA",
  var1 : "NF",
  var2 : 28,
  displayVar1 : function() { alert(this.var1) ; },
  displayVar2 : function() { alert(this.var2) ; },
  createObject : function(){
    return {
   name : "no name",
   what : "objet créé par myNamespace.createObject",
   version : "v1.0",
   display : function() {
     alert(this.name+ " ("+this.what+") - "+ this.version);
   }
    }
  }
};
```

Et structurons chaque fichier d'extension de notre espace de la façon suivante :

Test de l'existence et de l'unicité de l'espace de noms

```
(function()
{
  var f = function()
  {
    if (typeof myNamespace == "undefined")
    {
      alert("the namespace myNamespace was deleted.");
      return false;
    }
    else if (typeof myNamespace.author == "undefined" ||
             myNamespace.author != "Nourdine FALOLA")
    {
      alert("the namespace myNamespace was altered.");
      return false;
    }

    return true;
  }

  if (window.addEventListener) // non-IE browser
  {
    window.addEventListener("load",f,false);
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Test de l'existence et de l'unicité de l'espace de noms

```
  }
  else if (window.attachEvent) // IE browser
  {
    window.attachEvent("onload",f);
  }

  if (!f())
    return;

  // put your extension code here

})(); // run the anonymous function
```

Vous trouverez en annexe un rappel sur l'utilisation des fonctions anonymes

Que réalise la fonction anonyme dont la définition représente tout notre fichier d'extension et qui est exécutée dans la foulée :

- elle définie une fonction f vérifiant l'existence de la variable portant le nom de notre espace, l'existence d'une propriété author contenant le nom de l'auteur de la bibliothèque.
- elle attache un événement à la fin du chargement de la page appelant ce code (exécution de la fonction f)
- elle teste l'exécution de f renvoie bien true et stoppe le script si false
- elle étend l'espace de noms par ajout de nouvelles propriétés

Détaillons chaque étape.

D'abord tout le code de votre fichier d'extension est encapsulé dans une fonction anonyme exécutée dès la fin de sa définition grâce à la syntaxe suivante :

Syntaxe de l'exécution d'une fonction anonyme

```
(function() {
// code
})() ;
```

Une fonction f est définie. Cette fonction réalise plusieurs tests pour vérifier que la variable représentant l'espace de nom existe et qu'il s'agit bien de celle que vous avez créée dans le fichier initial (par vérification de l'existence de la propriété author et de sa valeur), si c'est le cas elle renvoie true, sinon false.

Cette fonction est ensuite attachée à l'événement onload de la page, c'est-à-dire qu'elle sera exécutée dès la fin du chargement de la page. Pourquoi à la fin du chargement de la page ? Parce que si un utilisateur écrase votre espace, il y a de fortes chances que ce soit avant le chargement complet de la page.

Puis le retour de cette fonction est testé afin de voir tout de suite si on manipule bien la bonne variable myNamespace. Si oui, on passe à la définition des nouvelles propriétés qui vont étendre notre espace de noms, si non, la fonction s'arrête là et retourne.

Comprenez bien que si la variable n'est pas celle que vous croyez, un message d'alerte vous le dira :).

II-D - Les sous-espaces de noms

Maintenant vous savez regrouper vos sources dans un domaine et créer différents domaines pour des contextes spécifiques. C'est pas mal, mais vous pouvez encore améliorer la donne en ajoutant plusieurs niveaux de nommage. Pourquoi plusieurs niveaux de nommage ?

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Tout d'abord si vous avez créé différents espaces de noms, c'est pour éviter les conflits de noms et regrouper vos fonctionnalités suivant des contextes particuliers. Mais de la même façon qu'il pouvait être pertinent d'avoir deux fonctionnalités portant le même nom, il en est de même pour les espaces de noms (et là vous commencez à avoir mal à la tête).

Ensuite, il serait bien pratique de n'avoir qu'une seule entrée pour identifier les fonctionnalités provenant de votre (vos) bibliothèque(s) et structurer suivant une hiérarchie pertinente vos fonctionnalités. En plus, n'avoir qu'une entrée vous permet de n'avoir à vous soucier que d'une possibilité de conflit de nom entre votre espace de noms global et une variable lambda.

Avec ce que vous avez appris précédemment, vous avez tous les outils pour agrémenter votre espace de noms principal de sous-espaces. Comme vous l'avez vu, un espace de noms en JavaScript est un objet dont les propriétés sont les fonctionnalités de votre bibliothèque. De la même façon, vous pouvez créer une propriété qui aura elle-même comme propriétés d'autres fonctionnalités, il s'agit alors d'un sous-espace de noms, un espace dans l'espace.

Ajoutons à notre espace de nom précédent deux sous-espaces et un sous-espace à l'un d'entre eux. Nous aurons alors 3 niveaux de profondeur.

Ajouter des sous-espaces de noms

```
myNamespace.subNamespace1 = {
  displayMsg : function(){
    alert("you're in subNamespace1");
  }
};

myNamespace.subNamespace2 = {
  displayMsg : function(){
    alert("you're in subNamespace2");
  }
};

myNamespace.subNamespace1.subSubNamespace = {
  displayMsg : function(){
    alert("you're in subSubNamespace");
  }
};
```

Testons les extensions de notre espace de noms global :

Utiliser les sous-espaces de noms

```
myNamespace.subNamespace1.displayMsg();
myNamespace.subNamespace2.displayMsg();
myNamespace.subNamespace1.subSubNamespace.displayMsg();

// you're in subNamespace1
// you're in subNamespace2
// you're in subSubNamespace
```

Vous constatez qu'au premier niveau vous avez les fonctionnalités de l'espace de noms myNamespace (var1, var2, displayVar1, displayVar2, createObject), au second niveau les fonctionnalités de subNamespace1 et subNamespace2 et au troisième niveau les fonctionnalités de subSubNamespace !

II-E - Les limites des espaces de noms en JavaScript

Jusque là nous avons été super optimistes sur notre conception d'un espace de noms en JavaScript. Pourtant, j'ai dit au début de cet article que la notion d'espace de noms n'existe pas dans ce langage et que ce que nous pouvions faire, au mieux, c'est l'émuler. Tout ceci implique certaines limitations.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Le JavaScript est un langage interprété. Aucune vérification n'est faite à la compilation puisqu'il n'y a pas de compilation. Donc si un problème survient c'est à l'exécution, ce qui est très embêtant pour l'utilisateur (et pour vous en tant que concepteur). C'est pourquoi nous avons fait ce que nous pouvions pour détecter en amont les problèmes possibles et prévenir par une alerte le cas échéant et en stoppant les traitements voués à l'échec.

Pourtant nous ne sommes pas capable de prévenir tous les problèmes. La principale limitation est que nous ne pouvons pas nous assurer de la bonne utilisation de notre bibliothèque par des tiers. Entre autres ils peuvent :

- Ecraser tout ou partie des fonctionnalités de notre bibliothèque en les redéfinissant (vil gredin !).
  Si cette fonction était utilisée quelque part en interne dans votre bibliothèque autant dire que ça ne marche plus.

Ecrasement d'une fonctionnalité dans le code client

```
// Script utilisateur
myNamespace.displayVar1 = null;
```

- Ecraser nos tests de validité sur le chargement de la page en affectant un callback sur l'événement onload façon old school (traditional binding).

Ecrasement du test de validité de l'espace de noms dans le code client

```
// Script utilisateur après l'inclusion du Listing 3 8
window.onload = function(){ alert("gotcha !!") } ;
```

- Ecraser notre espace de noms APRES que l'on ait testé sa validité en ajoutant un autre callback à la suite du notre (avec addEventListener ou attachEvent suivant le navigateur). Auquel cas il n'y a pas d'alerte pour le prévenir du méfait.

Contournement du test de validité de l'espace de noms dans le code client

```
// Script utilisateur après l'inclusion du Listing 3 8
var f = function()
{
  myNamespace = null;
  alert("gotcha again !!");
}

if (window.addEventListener) // non-IE browser
{
  window.addEventListener("load",f,false);
}
else if (window.attachEvent) // IE browser
{
  window.attachEvent("onload",f);
}
```

- Utiliser des raccourcis pour ne pas avoir à écrire le nom de l'espace ou du sous-espace de noms, de la classe, de la variable, de la fonction en entier. De ce fait nous ne pouvons garantir que sa nouvelle variable ne sera pas écrasée par une autre qu'il utilisera ailleurs.
  Ce problème s'apparente à celui que l'on peut rencontrer dans des langages compilés comme le C++ où l'on utilise l'instruction using pour ne pas avoir à préfixer les fonctionnalités par leur espace de noms, ce qui peut induire des ambiguïtés. Donc à utiliser avec d'extrêmes précautions, voire à proscrire.

Utilisation de raccourcis d'écriture dans le code client

```
// Script utilisateur
var raccourci = myNamespace.displayVar1;
raccourci() ; // affiche la valeur de myNamespace.var1
[...]
raccourci = myNamespace.subNamespace1.displayMsg;
raccourci() ; // affiche "you're in subNamespace1"
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Utilisation de raccourcis d'écriture dans le code client

```
raccourci = myNamespace.var1;
[...]
raccourci(); // ERREUR raccourci is not a function
```

Mais tout ça n'est pas vraiment de notre ressort. Nous avons fait le maximum pour qu'il n'y ait pas de problème (on peut aussi mettre à disposition une documentation indiquant aux utilisateurs ce qu'il faut faire ou ne pas faire pour que ça marche au mieux). Au-delà de la conception de nos bibliothèques nous ne pouvons garantir aux utilisateurs le bon fonctionnement de leurs programmes s'ils n'utilisent pas correctement nos outils.

- Raccourcis d'utilisation (var toto = namespace.ns1.ns2 ;) équivalent de using namespace std ; en C++

III - Exemples

Et voilà, nous en sommes à présent à mettre en application tout ce que vous avez appris dans cet article. Je vous propose de mettre en oeuvre une mini bibliothèque illustrative et un cas d'utilisation, ce qui vous permettra d'appréhender dans sa globalité la notion d'espace de noms en JavaScript.

III-A - Votre bibliothèque : myOwnJSLibrary.js

myOwnJSLibrary.js

```
<!-- <![CDATA[
var myOwnJSLibrary = {
  author : "Nourdine FALOLA",

  addEventListener : function(o,typeEvent,callback){
    if (o.addEventListener){
      o.addEventListener(typeEvent,callback,false);
    }
    else if (o.attachEvent){
      o.attachEvent("on" + typeEvent,callback);
    }
  },

  displayException : function(){
    if (arguments.length < 1){
      // pas de paramètres passés à la fonction
      return;
    }

    var exc = arguments[0];
    if (!(exc instanceof Object) || exc.message === undefined){
      // le paramètre n'est pas une exception
      return;
    }

    // on affiche toutes les propriétés de l'exception
    // (elles diffèrent suivant le navigateur)
    var msg = "";
    for (var prop in exc){
      msg += exc[prop] + "\n";
    }
    alert(msg);
  }

};
// ]]> -->
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III-B - myOwnJSLibrary.Math.js

myOwnJSLibrary.Math.js

```
<!-- <![CDATA[
(function(){
  var f = function(){
    if (typeof myOwnJSLibrary == "undefined"){
      alert("the namespace myOwnJSLibrary was deleted.");
      return false;
    }
    else if (typeof myOwnJSLibrary.author == "undefined" ||
 myOwnJSLibrary.author != "Nourdine FALOLA"){
      alert("the namespace myOwnJSLibrary was altered.");
      return false;
    }

    return true;
  }

  if (window.addEventListener){ // non-IE browser
    window.addEventListener("load",f,false);
  }
  else if (window.attachEvent){ // IE browser
    window.attachEvent("onload",f);
  }

  if (!f())
    return;

  // extends myOwnJSLibrary

  myOwnJSLibrary.Math = {

    PI : Math.PI,

    add : function(){
      try{
        switch(arguments.length){
          case 0:
            return undefined;
          case 1:
            if (typeof arguments[0] != "number"){
              return undefined;
            }
            return arguments[0];
          default:
            var ret = 0;
            for (var i=0; i<arguments.length; i++){
              if (typeof arguments[i] != "number"){
                return undefined;
              }
              ret += arguments[i];
            }
            return ret;
        }
      }
      catch(exc){
        myOwnJSLibrary.displayException(exc);
      }
    }
  };

})(); // run the anonymous function

// ]]> -->
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III-C - myOwnJSLibrary.Math.Statistic.js

myOwnJSLibrary.Math.Statistic.js

```
<!-- <![CDATA[
(function(){
  var f = function(){
    if (typeof myOwnJSLibrary == "undefined"){
      alert("the namespace myOwnJSLibrary was deleted.");
      return false;
    }
    else if (typeof myOwnJSLibrary.author == "undefined" ||
 myOwnJSLibrary.author != "Nourdine FALOLA"){
      alert("the namespace myOwnJSLibrary was altered.");
      return false;
    }
    else if (typeof myOwnJSLibrary.Math == "undefined"){
      alert("the namespace myOwnJSLibrary.Math was deleted.");
      return false;
    }

    return true;
  }

  if (window.addEventListener){ // non-IE browser
    window.addEventListener("load",f,false);
  }
  else if (window.attachEvent){ // IE browser
    window.attachEvent("onload",f);
  }

  if (!f())
    return;

  // extends myOwnJSLibrary.Math

  myOwnJSLibrary.Math.Statistic = {

    fact : function(n){
      try{
        if (n>0){
          return n*this.factorielle(n-1);
        }
        else if (n==0){
          return 1;
        }

        return undefined;
      }
      catch(exc){
        myOwnJSLibrary.displayException(exc);
      }
    },

    combi : function(n,k){
      if (k>n){
        return undefined;
      }

      return this.fact(n)/(this.fact(k)*this.fact(n-k));
    }

  };

})(); // run the anonymous function
// ]]> -->
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III-D - myOwnJSLibrary.String.js

myOwnJSLibrary.String.js

```
<!-- <![CDATA[
(function(){
  var f = function(){
    if (typeof myOwnJSLibrary == "undefined"){
      alert("the namespace myOwnJSLibrary was deleted.");
      return false;
    }
    else if (typeof myOwnJSLibrary.author == "undefined" ||
 myOwnJSLibrary.author != "Nourdine FALOLA"){
      alert("the namespace myOwnJSLibrary was altered.");
      return false;
    }

    return true;
  }

  if (window.addEventListener){ // non-IE browser
    window.addEventListener("load",f,false);
  }
  else if (window.attachEvent){ // IE browser
    window.attachEvent("onload",f);
  }

  if (!f())
    return;

  // extends myOwnJSLibrary

  myOwnJSLibrary.String = {

    add : function(){
      try{
        switch(arguments.length){
          case 0:
            return undefined;
          case 1:
            return "" + arguments[0];
          default:
            var ret = "";
            for (var i=0; i<arguments.length; i++){
             ret += arguments[i];
            }
            return ret;
        }
      }
      catch(exc){
        myOwnJSLibrary.displayException(exc);
      }
    },

    format : function(){
      switch(arguments.length){
        case 0:
          return undefined;
        case 1:
          return "" + arguments[0];
        default:
          var ret = arguments[0];
          var expr = "\\{n\\}";

          for (var i=1; i<arguments.length; i++){
            var motif = new RegExp(expr.replace("n",i-1),"g");
            ret = ret.replace(motif,"" + arguments[i]);
          }
          return ret;
      }
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

myOwnJSLibrary.String.js

```
    }

  };

})(); // run the anonymous function

// ]]> -->
```

III-E - Utilisation dans un code client : test.html

Code client utilisant la bibliothèque myOwnJSLibrary

```
<html>
  <head>
<script type="text/javascript" src="myOwnJSLibrary.js"></script>
<script type="text/javascript" src="myOwnJSLibrary.Math.js"></script>
<script type="text/javascript" src="myOwnJSLibrary.Math.Statistic.js"></script>
<script type="text/javascript" src="myOwnJSLibrary.String.js"></script>
<script type="text/javascript">
  <!-- <![CDATA[
  myOwnJSLibrary.addEventListener(window, "load", function(){

    try{
      alert(myOwnJSLibrary.Math.PI);
      alert(myOwnJSLibrary.Math.add(5,6,7,8,9));
      alert(myOwnJSLibrary.Math.add(1,7,"8",9));
      alert(myOwnJSLibrary.Math.Statistic.combi(4,2));
      alert(myOwnJSLibrary.String.add(5,6,7,"8",9));
      alert(myOwnJSLibrary.String.add(null,6,7,"8",{ name : "un objet" }, "viva españa"));
      alert(myOwnJSLibrary.String.format("{0}, {1}",{},123));
    }
    catch(exc){
      myOwnJSLibrary.displayException(exc);
    }

  });
// ]]> -->
</script>
  </head>
  <body>
  </body>
</html>
```

IV - Annexe

IV-A - Le mot-clé "this"

Dans des langages objets comme Java ou C++, this désigne au sein d'une classe l'instance de la classe sur laquelle on agit. Ce n'est pas le cas en JavaScript. En JavaScript, "this" désigne l'objet qui appelle la méthode dans laquelle il se trouve (oui il y a une subtile différence !). Prenons un exemple :

this en JavaScript

```
function who(){
  alert(this.name);
}

var o1 = { name : "objet 1"  };
var o2 = { name : "objet 2"  };

o1.who1 = who;
o2.who2 = who;

o1.who1(); // affiche "objet 1"
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

this en JavaScript

```
o2.who2(); // affiche "objet 2"
```

S'il s'agit d'une méthode au sein d'une classe, vous utilisez le this de la même manière qu'en C++.

Utiliser this comme en C++

```
function MyClass(){
  name : "instance of MyClass",
  who : function(){
    alert(this.name);
  }
}

var obj = new MyClass();
obj.who(); // "instance of MyClass"
obj.name = "new instance of MyClass";
obj.who(); // "new instance of MyClass"
```

Cependant il s'agit toujours en réalité de la règle de l'objet appelant la méthode. Si vous affectez la méthode who à un nouvel objet qui n'est pas une instance de MyClass, this fait maintenant référence à cet objet.

this relatif à l'objet appelant

```
var weird = {};
weird.name = "weird object";
weird.who = obj.who;
weird.who(); // "weird object"
```

IV-B - Les fonctions anonymes

Vous avez vu dans cet article qu'on utilisait la syntaxe suivante :

Syntaxe d'une fonction anonyme

```
(function(){
  // votre code ici
})();
```

Cela est strictement équivalent à :

Ecriture équivalente à une fonction anonyme

```
function myFunction(){
  // votre code ici
}
myFunction();
```

C'est-à-dire qu'il y a une définition de fonction suivie de l'exécution de cette dernière. La différence entre les deux notations est que dans la première la fonction ne porte pas de nom, c'est pourquoi on désigne ce genre de fonction comme étant une fonction anonyme.

L'intérêt des fonctions anonymes est de pouvoir exécuter du code à l'issu de la définition de celui-ci ou encore d'affecter une fonction à un gestionnaire d'événement (ou autre) sans avoir besoin de nommer la fonction en la déclarant dans un premier temps, puis en l'utilisant ensuite. En ce qui concerne la première utilisation, on peut obtenir la même chose sans encapsuler le code dans une fonction. Cependant en encapsulant votre code dans une fonction anonyme, vous réduisez la portée des variables qui y sont déclarées. Vous travaillez dans un contexte local à la fonction.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Contexte local à une fonction anonyme

```
/* contexte local à la fonction anonyme */
var n = 10;
var foo = function(){
  alert("global foo");
};

alert(n); // 10
foo(); // global foo

(function(){
  var n = 50;
  var foo = function (){
    alert("anonymous foo");
  };

  alert(n); // 50
  foo(); // anonymous foo
})();

alert(n); // 10
foo(); // global foo
```

Affectation d'une fonction anonyme à une propriété d'un objet

```
/* fonction anonyme et gestionnaire d'événement */
objet.onclick = function (event){
  // votre code ici
}
```

Vous pouvez exploiter cet état de fait pour simuler ce qu'on appelle des variables et méthodes privées dans d'autres langages objets. En effet les variables et méthodes définies dans une fonction anonymes ne sont accessibles que dans cette méthode. C'est une façon d'encapsuler les fonctionnalités.

Au passage, remarquez qu'on utilise les fonctions anonymes (et le mot-clés this) avec la notation JSON :

Notation JSON, fonction anonyme et this

```
var obj = {
  var1 : "variable 1",
  var2 : "variable 2",
  getVars : function(){
    alert(this.var1 + " et " +  this.var2);
  }
};
```

IV-C - Importation de symboles

A la section 3.5, je vous énumérais les limites des espaces de noms en JavaScript. En particulier je vous montrais avec le Listing 3 15 qu'un utilisateur de votre espace de noms pouvait utiliser des raccourcis d'écriture et qu'une utilisation maladroite pouvait causer des problèmes.

Cependant, à l'intérieur de votre librairie, vous pouvez vous-même utiliser ce genre de raccourcis d'écriture si la notation complète commence à devenir fastidieuse avec de multiples niveaux de noms. Si votre code est encapsulé dans une fonction anonyme, les raccourcis que vous emploierez seront définis localement à cette fonction et n'interfèrerons pas avec le code utilisateur. Par contre, à votre charge de ne pas utiliser les mêmes noms dans une même portée (et même dans des portées différentes au risque de semer la confusion).

Cette technique porte le nom d'importation de symboles.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Importation de symboles

```
myLibrary.sub1.sub2.sub3.Event = {
  namespace :  "myLibrary.sub1.sub2.sub3.Event"
}
[...]
var EVENT = myLibrary.sub1.sub2.sub3.Event;
alert(EVENT.namespace); // "myLibrary.sub1.sub2.sub3.Event"
```

V - Remerciements

Je remercie Didier Mouronval (aka **Bovino**) pour sa relecture attentive, Pierre Schwartz (aka **Khayyam90**) pour ses conseils et Grégory Dumas (aka **freegreg**) pour son aide à la génération de cet article. Je remercie toute l'équipe Web pour l'intérêt et les conseils manifestés pour cet article.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2009 - Nourdine FALOLA. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.