Namespace pattern*

Hannes Hirzel

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

Abstract

This modules explains how to avoid having too many global variables in JavaScript programs. This is accomplished by using the 'namespace pattern'. This pattern is used in many libraries.

1 Introduction

In this module a "pattern" is described which explains how to reduce the number of identifiers in JavaScript on the global scope level. This is useful if you want to write a library and avoid a clash of names with the user of your library.

2 The problem

Let us consider the following code in JavaScript.

```
    var n=3;
var x=0;
var text="abc";

var open = function(anArg) {
    console.log('within the "open" function');
    console.log("text=" + text);
    console.log("n=" + n);
    }

var doCalculation = function(i,j) {
    console.log('within doCalculation function');
};

open(text);
```

When we run the code in a **web console** or better in a **scratch pad** we get the following result

```
[07:39:08.141] within open function
[07:39:08.145] text=abc
[07:39:08.148] n=3
```

*Version 1.5: Feb 11, 2012 3:08 am -0600
†http://creativecommons.org/licenses/by/3.0/

The problem is that we now have five identifier on the global level.

Identifier

- n
- x
- text
- open
- doCalculation

If we add another JavaScript library it is likely that it might contain identifiers with the same name. This leads to a clash of names.

We want to keep the number of identifiers declared at a global level to a minimum.

3 Solution

We can reduce the number of global identifiers to one. This will be the **namespace** identifier. We declare an object `NSP1` to be an empty JavaScript object.

```
var NSP1 = {};
```

Then we add our variable and function declarations as properties and methods of `NSP1`.

```
    NSP1.n=3;
NSP1.x=0;
NSP1.text="abc";

NSP1.open = function(anArg) {
    console.log('within open function');
    console.log("text=" + this.text);
    console.log("n=" + this.n);
    }

NSP1.doCalculation = function(i,j) {
    console.log('within doCalculation function');
};

NSP1.open(text);
```

Result

We have successfully reduced the number of global variables from 5 to 1. The only downside is that we have to prefix the reference to the variables with `NSP1`.

4 References

Axel Rauschmayer, Patterns for modules and namespaces in JavaScript[1]

Stoyan Stefanov, JavaScript Patterns, O'Reilly 2010, 'Namespace pattern', p. 87

Avoid globals[2] (Opera Developer Article)

Addy Osmani, JavaScript patterns; Singleton pattern[3]

[1] http://www.2ality.com/2011/04/modules-and-namespaces-in-javascript.html

[2] http://dev.opera.com/articles/view/javascript-best-practices/#avoidglobals

[3] http://www.addyosmani.com/resources/essentialjsdesignpatterns/book/#singletonpatternjavascript