


## Basic JavaScript Part 8: Namespaces

 January 26th, 2011

Here are the links to the previous installments:

1. [Functions](#)
2. [Objects](#)
3. [Prototypes](#)
4. [Enforcing New on Constructor Functions](#)
5. [Hoisting](#)
6. [Automatic Semicolon Insertion](#)
7. [Static Properties and Methods](#)

In my [previous post](#), I showed how you can 'emulate' static members in JavaScript without having a dedicated syntax for it. For this post, I'm going to discuss namespaces in JavaScript. In order to reduce the number of objects and functions that are added to the global scope in our applications, using namespaces in JavaScript is definitely a recommended practice. Just like static members, namespaces don't have any dedicated syntax built into the language either. But we're able to get the same benefits by creating a single global object and adding all our objects and functions to this object.

```
var AppSpace = AppSpace || {};  
  
AppSpace.Podcast = function {  
  this.title = 'Astronomy Cast';  
  this.description = 'A fact-based journey through the galaxy.';  
  this.link = 'http://www.astronomycast.com';  
};  
  
AppSpace.Podcast.prototype.toString = function() {  
  return 'Title: ' + this.title;  
}
```

This way we lower the possibility of naming collisions when using our code in conjunction with other JavaScript libraries. We can also use the same naming conventions for namespaces as in any other programming language that does provide syntactical support. Suppose we want to use a namespace like *"MyCompany.MyApplication.Model"*. We can accomplish this by using the same approach as shown earlier:

```
var MyCompany = MyCompany || {};  
MyCompany.MyApplication = {};  
MyCompany.MyApplication.Model = {};
```

or something like the following:

```
var MyCompany = MyCompany || {  
  MyApplication: {  
    Model: {}  
  }  
};
```

However, this approach can become very cumbersome and hard to maintain when our code expands over time. In order to overcome this issue we can use a general purpose namespace function to achieve the same thing using a single line of code.

```
var model = namespace('MyCompany.MyApplication.Model');  
  
model.Podcast = function {  
  this.title = 'Astronomy Cast';  
  this.description = 'A fact-based journey through the galaxy.';  
  this.link = 'http://www.astronomycast.com';  
};  
  
model.Podcast.prototype.toString = function() {
```

```
    return 'Title: ' + this.title;
}
```

Here's an example of how we can implement such a namespace function.

```
function namespace(namespaceString) {
    var parts = namespaceString.split('.'),
        parent = window,
        currentPart = '';

    for(var i = 0, length = parts.length; i < length; i++) {
        currentPart = parts[i];
        parent[currentPart] = parent[currentPart] || {};
        parent = parent[currentPart];
    }

    return parent;
}
```

Using this approach requires less typing and the resulting code also looks less verbose. Some JavaScript libraries, like YUI and the Dojo Toolkit, provide their own implementations for such a namespace utility function. I encourage you to take a look at their implementations as well.

Until next time.