**global namespace object**

[复制代码](javascript:void(0);)

// global object

var MYAPP = {};

// constructors

MYAPP.Parent = function() {

};

MYAPP.Child = function() {

};

// a variable

MYAPP.some\_var = 1;

// an object container

MYAPP.modules = {};

// nested objects

MYAPP.modules.module1 = {};

MYAPP.modules.module1.data = {

a : 1,

b : 2

};

MYAPP.modules.module2 = {};

[复制代码](javascript:void(0);)

**Drawbacks**

• A bit more to type; prefixing every variable and function does add up in the total amount of code that needs to be downloaded

• Only one global instance means that any part of the code can modify the global instance and the rest of the functionality gets the updated state

• Long nested names mean longer (slower) property resolution lookups

**General Purpose Namespace Function**

[复制代码](javascript:void(0);)

// unsafe

var MYAPP = {};

// better

if ( typeof MYAPP === "undefined") {

var MYAPP = {};

}

// or shorter

var MYAPP = MYAPP || {};

// using a namespace function

MYAPP.namespace('MYAPP.modules.module2');

// equivalent to:

// var MYAPP = {

// modules: {

// module2: {}

// }

// };

MYAPP.namespace = function(ns\_string) {

var parts = ns\_string.split('.'), parent = MYAPP, i;

// strip redundant leading global

if (parts[0] === "MYAPP") {

parts = parts.slice(1);

}

for ( i = 0; i < parts.length; i += 1) {

// create a property if it doesn't exist

if ( typeof parent[parts[i]] === "undefined") {

parent[parts[i]] = {};

}

parent = parent[parts[i]];

}

return parent;

};

// assign returned value to a local var

var module2 = MYAPP.namespace('MYAPP.modules.module2');

module2 === MYAPP.modules.module2;

// true

// skip initial `MYAPP`

MYAPP.namespace('modules.module51');

// long namespace

MYAPP.namespace('once.upon.a.time.there.was.this.long.nested.property');

[复制代码](javascript:void(0);)

**References*:***

*JavaScript Patterns -*by Stoyan Stefanov (O`Reilly)