[复制代码](javascript:void(0);)

MYAPP.namespace('MYAPP.utilities.array');

MYAPP.utilities.array = (function () {

// dependencies

var uobj = MYAPP.utilities.object,

ulang = MYAPP.utilities.lang,

// private properties

array\_string = "[object Array]",

ops = Object.prototype.toString;

// private methods

// ...

// end var

// optionally one-time init procedures

// ...

// public API

return {

inArray: function (needle, haystack) {

for (var i = 0, max = haystack.length; i < max; i += 1) {

if (haystack[i] === needle) {

return true;

}

}

},

isArray: function (a) {

return ops.call(a) === array\_string;

}

// ... more methods and properties

};

}());

[复制代码](javascript:void(0);)

**Revealing Module Pattern**

Privacy pattern

The above can become:

[复制代码](javascript:void(0);)

MYAPP.utilities.array = (function () {

// private properties

var array\_string = "[object Array]",

ops = Object.prototype.toString,

// private methods

inArray = function (haystack, needle) {

for (var i = 0, max = haystack.length; i < max; i += 1) {

if (haystack[i] === needle) {

return i;

}

}

return−1;

},

isArray = function (a) {

return ops.call(a) === array\_string;

};

// end var

// revealing public API

**return {**

**isArray: isArray,**

**indexOf: inArray**

**};**

}());

[复制代码](javascript:void(0);)

Modules That Create Constructors

The only difference is that the immediate function that wraps the module will return a function at the end, and not an object.

[复制代码](javascript:void(0);)

MYAPP.namespace('MYAPP.utilities.Array');

MYAPP.utilities.Array = (function () {

// dependencies

var uobj = MYAPP.utilities.object,

ulang = MYAPP.utilities.lang,

// private properties and methods...

Constr;

// end var

// optionally one-time init procedures

// ...

// public API -- constructor

Constr = function (o) {

this.elements = this.toArray(o);

};

// public API -- prototype

Constr.prototype = {

constructor: MYAPP.utilities.Array,

version: "2.0",

toArray: function (obj) {

for (var i = 0, a = [], len = obj.length; i < len; i += 1) {

a[i] = obj[i];

}

return a;

}

};

// return the constructor

// to be assigned to the new namespace

return Constr;

}());

var arr = new MYAPP.utilities.Array(obj);

[复制代码](javascript:void(0);)

**Importing Globals into a Module**

In a common variation of the pattern, you can pass arguments to the immediate function that wraps the module. You can pass any values, but usually these are references to global variables and even the global object itself. Importing globals helps speed up the global symbol resolution inside the immediate function, because the imported variables become locals for the function.

[复制代码](javascript:void(0);)

MYAPP.utilities.module = (function (app, global) {

// references to the global object

// and to the global app namespace object

// are now localized

}(MYAPP, this));

[复制代码](javascript:void(0);)

**References*:***

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