JS模块化编程（2）——规范

Alex Sun
2014-11-18

随着项目规模的不断扩大，一个项目所需要的模块也越来越多，JS的文件个数也越来越多。在这种情况下，我们不可能再像之前那样，使用多个script标签来加载文件，因为JS文件是同步加载的，一次性加载过多的JS文件会导致长时间的阻塞。因此我们需要一种方法，来按需加载，只有当需要某个模块的时候才把它加载过来。而关于这种按需加载的方法，需要一个较为统一的规范，这样才方便开发和使用。目前在服务器端，使用的是CommonJS规范；在浏览器端，常用的是AMD和CMD规范。

1. CommonJS

首先，CommonJS不仅仅是一个模块化的规范，而是它的规范中包含了模块化这一部分。CommonJS是服务器端JS的一套规范，包括FS，IO，Binary，Network等一系列内容，由于JS标准中没有模块机制，因此CommonJS中也包含了一个模块化的规范，即Modules，可以在这里查看CommonJS官方文档。

在CommonJS中，每一个JS文件就是一个模块，每个模块有一个module对象，module下有一个exports对象，所有对外的借口都是挂载在exports下的。当使用的时候，用require来加载所需要的模块。例如：

```
/** math.js **/
exports.add = function(num1, num2) {
    return num1 + num2;
};

/** main.js **/
var math = require("math");
var num = math.add(3, 4);
```

然而存在着一个问题就是循环引用，即两个模块之间互相依赖，标准中解决方法如下：

> If there is a dependency cycle, the foreign module may not have finished executing at the time it is required by one of its transitive dependencies; in this case, the object returned by “require” must contain at least the exports that the foreign module has prepared before the call to require that

> led to the current module's execution.

看个例子：

```
/** a.js **/
// a1
var b = require("b.js");
//a2

/** b.js **/
// b1
var a = require("a.js");
// b2

/** main.js **/
var a = require("a.js");
var b = require("b.js");
```

在执行a.js的时候，需要依赖b.js，此时加载b.js，然而在加载b.js的过程中，发现它依赖a.js，因此此时b.js中拿到的a只是a1部分，a2那一部分拿不到。

在CommonJS中，所有的模块加载都是同步的，由于CommonJS主要是在服务器端，所有的问价资源也都是在服务器上，因此同步加载资源开销并不会太大。然而在浏览器端，如果所有资源依然是同步加载，就会导致长时间的阻塞。在CommonJS向浏览器端兼容的过程中，出现了AMD(Asynchronous Module Definition)规范。

2. AMD

AMD是浏览器端的模块规范，可以在这里查看到详细说明。AMD加载模块是异步的，因此不会影响后续代码的执行。其核心接口为define()和require()，define()用来定义一个模块，require()用来引入一个模块。

define()函数原型为：

```
define(id?, dependencies?, factory);
```

其中：

- id是一个字符串，用来标识该模块；
- dependencies是一个数组，用来声明该模块所依赖的模块，数组中的每一项都是一个模块的id；
- factory是一个函数或者对象，用来作为模块的导出。

同时，define下还有一个amd属性，其值是一个对象，即define.amd={}，在实现AMD规范的过程中，可以为该对象添加属性。

下面是一个例子：

```
define("alpha", ["require", "exports", "beta"], function(require,
exports, beta) {
    exports.verb = function() {
        return beta.verb();
        //Or:
        return require("beta").verb();
    }
});
```

AMD不仅可以异步加载模块，还可以像CommonJS那样同步加载模块，例如：

```
define(function(require, exports, module) {
    var a = require('a'),
        b = require('b');

    exports.action = function() {};
});
```

在这种情况下，第一个参数必须为require。

上面主要都是define()的用法，下面来看看require()的用法。如下所示：

```
// require(String)
define(function(require) {
    var a = require('a');
});

// require(Array, Function)
define(function(require) {
    require(['a', 'b'], function(a, b) {
        //modules a and b are now available for use.
    });
});
```

此外，AMD还支持一些配置，具体可以参看文档。

目前实现AMD规范的库很多，其中使用比较广泛的是RequireJS。

3. CMD(Common Module Definition)

CMD的规范可以参考这里。可以发现，大体上，它与AMD还是比较相似的，但是也存在着很多不同。

首先，AMD是需要依赖前置的，即在define中，要首先声明需要哪些依赖；而在CMD中，遵循的是就近原则，即只在需要的时候才require进来，例如：

```
define(function(require, exports, module) {
var a = require("a");
a.doSomething();
});
```

另外，CMD也可以同步或者异步加载模块，不过直接使用require的时候是异步加载，同步加载的话使用的是require.async。

目前实现CMD规范的主要是SeaJS。

参考：

浅谈 JavaScript 模块化编程
CommonJS
AMD
CMD 模块定义规范