**JS模块化**

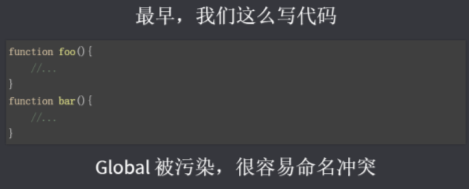
# 第1章：模块化入门

## 理解什么是模块/模块化

### 模块

1. 将一个复杂的程序依据一定的规则(规范)封装成几个块(文件), 并进行组合在一起
2. 这些拆分的文件就是模块，模块内部数据/实现是私有的, 只是向外部暴露一些接口(方法)与外部其它模块通信

### 1.1.2模块化的进化史









### 1.1.3 什么是模块化

当开发的项目是遵循模块化规范的项目，那么这个项目就是模块化的项目

## 1.2为什么要模块化？

1. 部署简单
2. 降低复杂度，提高解耦性
3. 避免命名冲突(减少命名空间污染)
4. 更好的分离, 按需加载
5. 更高复用性，高可维护性

## 1.3 模块化概念带来的问题



1. 请求过多
2. 依赖模糊
3. 难以维护

# 第2章：模块化规范

前言：一个大的项目必定会使用模块化，使用模块化就会使用相应的模块化规范，现在比较流行的模块化规范有以下几种：CommonJS, AMD, ES6, CMD

## 2.1. CommonJS

### 2.1.1 规范

1) http://wiki.commonjs.org/wiki/Modules/1.1

2) 每个文件都可当作一个模块

3) 在服务器端: 模块的加载是运行时同步加载的

4) 在浏览器端: 模块需要提前编译打包处理

### 2.1.2 基本语法

|  |  |
| --- | --- |
| 暴露模块 | 引入模块 |
| module.exports = value | 第三方模块: require(xxx), xxx为模块名 |
| exports.xxx = value | 自定义模块: require(xxx), xxx为模块文件路径 |

### 2.1.3 实现

|  |  |
| --- | --- |
| 服务器端 | 浏览器端 |
| Node.js  http://nodejs.cn/ | Browserify(CommonJs的打包工具)  http://browserify.org/ |
| Node.js运行时动态加载模块(同步) | Browserify是在转译(编译)时就会加载打包(合并)require的模块 |

## 2.2. AMD

### 2.2.1. 规范

1) Asynchronous Module Definition(异步模块定义)

2) <https://github.com/amdjs/amdjs-api/wiki/AMD>

3) 专门用于浏览器端, 模块的加载是异步的

### 2.2.2. 基本语法

1) 定义暴露模块

|  |
| --- |
| //定义没有依赖的模块  define(function(){  return 模块  }) |

表 1没有依赖的模块

|  |
| --- |
| //定义有依赖的模块  define(['module1', 'module2'], function(m1, m2){  return 模块  }) |

表 2有依赖的模块

2) 引入使用模块

|  |
| --- |
| require(['module1', 'module2'], function(m1, m2){  使用m1/m2  }) |

表 3注意显示声明依赖注入

### 2.2.3 实现

1) Require.js

2) <http://www.requirejs.cn/>

3) <http://www.ruanyifeng.com/blog/2012/11/require_js.html>

## 2.3 CMD

### 2.3.1. 规范

1) Common Module Definition(通用模块定义)

2) https://github.com/seajs/seajs/issues/242

3) 专门用于浏览器端, 模块的加载是异步的

4) 模块使用时才会加载执行

### 2.3.2. 基本语法

1) 定义暴露模块

|  |
| --- |
| //定义没有依赖的模块  define(function(require, exports, module){  exports.xxx = value  module.exports = value  }) |

表 1没有依赖的模块

|  |
| --- |
| //定义有依赖的模块  define(function(require, exports, module){  //引入依赖模块(同步)  var module2 = require('./module2')  //引入依赖模块(异步)  require.async('./module3', function (m3) {    })  //暴露模块  exports.xxx = value  }) |

表 2有依赖的模块

2) 引入使用模块

|  |
| --- |
| define(function (require) {  var m1 = require('./module1')  var m4 = require('./module4')  m1.show()  m4.show()  }) |

表 3注意当前模块不需要暴露的时候就只需要写require

### 2.3.3 实现

1)Sea.js

2) http://www.zhangxinxu.com/sp/seajs/

## 2.4 ES6

### 2.4.1 规范

1) http://es6.ruanyifeng.com/#docs/module

2) 依赖模块需要编译打包处理

### 2.4.2基本语法

1. 导出模块: export
2. 引入模块: import

### 2.4.3 实现

1) 使用Babel将ES6编译为ES5代码： babel 源文件路径 -d 编译后文件路径

2) 使用Browserify编译打包js browserify 源文件路径/源文件 -o 编译后文件路径/文件

# 第3章：扩展阅读

1. <https://github.com/seajs/seajs/issues/588(前端模块化开发那点历史)>
2. <http://zccst.iteye.com/blog/2215317(CommonJS，AMD，CMD区别)>
3. http://www.zhihu.com/question/20351507/answer/14859415(AMD和CMD 的区别)
4. http://www.ruanyifeng.com/blog/2012/10/javascript\_module.html(Javascript模块化编程)