JS模块化

- 模块化的理解
- 什么是模块?
    - 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
    - 块的内部数据/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
- 一个模块的组成
    - 数据--->内部的属性
    - 操作数据的行为--->内部的函数
- 模块化
    - 编码时是按照模块一个一个编码的, 整个项目就是一个模块化的项目
- 模块化的进化过程
    - 全局function模式 :
        - 编码: 全局变量/函数
        - 问题: 污染全局命名空间, 容易引起命名冲突/数据不安全
    - namespace模式 :
        - 编码: 将数据/行为封装到对象中
        - 解决: 命名冲突(减少了全局变量)
        - 问题: 数据不安全(外部可以直接修改模块内部的数据)
    - IIFE模式/增强
        - IIFE : 立即调用函数表达式--->匿名函数自调用
        - 编码: 将数据和行为封装到一个函数内部, 通过给window添加属性来向外暴露接口
        - 引入依赖: 通过函数形参来引入依赖模块

```
(function(window, module2){
  var data = 'atguigu.com'
  function foo() {
     module2.xxx()
     console.log('foo()'+data)
  }
  function bar() {
     console.log('bar()'+data)
  }

  window.module = {foo}
})(window, module2)
```

- 模块化规范
    - CommonJS
        - Node.js : 服务器端
        - Browserify : 浏览器端　也称为js的打包工具
        - 基本语法:
            - 定义暴露模块 : exports

```
exports.xxx = value
module.exports = value
```

引入模块 : require

```
var module = require('模块名/模块相对路径')
```

        - 引入模块发生在什么时候?
            - Node : 运行时, 动态同步引入
            - Browserify : 在运行前对模块进行编译/转译/打包的处理(已经将依赖的模块包含进来了),
              运行的是打包生成的js, 运行时不存在需要再从远程引入依赖模块
    - AMD : 浏览器端
        - require.js
        - 基本语法

- 定义暴露模块: define([依赖模块名], function(){return 模块对象})
- 引入模块: require(['模块1', '模块2', '模块3'], function(m1, m2){//使用模块对象})
- 配置:

```
require.config({
  //基本路径
  baseUrl : 'js/',
  //标识名称与路径的映射
  paths : {
    '模块1' : 'modules/模块1',
    '模块2' : 'modules/模块2',
    'angular' : 'libs/angular',
    'angular-messages' :
'libs/angular-messages'
  },
  //非AMD的模块
  shim : {
    'angular' : {
        exports : 'angular'
    },
    'angular-messages' : {
        exports : 'angular-messages',
        deps : ['angular']
    }
  }
})
```

- CMD : 浏览器端
  - sea.js
  - 基本语法
    - 定义暴露模块:

```
define(function(require, module,
exports){
  通过require引入依赖模块
  通过module/exports来暴露模块
  exports.xxx = value
})
```

        - 使用模块seajs.use(['模块1', '模块2'])
- ES6
    - ES6内置了模块化的实现
    - 基本语法
        - 定义暴露模块 : export
            - 暴露一个对象:

```
export default 对象
```

            - 暴露多个:

```
export var xxx = value1
export let yyy = value2

var xxx = value1
let yyy = value2
export {xxx, yyy}
```

        - 引入使用模块 : import
            - default模块:

```
import xxx  from '模块路径/模块名'
```

            - 其它模块

```
import {xxx, yyy} from '模块路径/模块名'
import * as module1 from '模块路径/模块名'
```

    - 问题: 所有浏览器还不能直接识别ES6模块化的语法
    - 解决:
        - 使用Babel将ES6--->ES5(使用了CommonJS) ----浏览器还不能直接支行
        - 使用Browserify--->打包处理----浏览器可以运行