1. 课程目标

1. 理解模块与模块化
2. 了解各种模块化规范及其实现
3. 区别各个模块化规范之间的区别
4. 掌握基于CommonJS和ES6模块化规范的编码

2. 模块化的理解

1). 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

2). 一个模块的组成

私有的数据--->内部的变量
私有的行为(操作数据)--->内部的函数
向外暴露n个行为

3). 模块化

描述一种特别的编码项目JS的方式: 以模块为单元一个一个编写的
模块化的项目: JS编码时是按照模块一个一个编码的

4). 模块化的进化过程

1. 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
2. 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)
3. 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'
            function foo() {
               module2.xxx()
               console.log('foo()'+data)
            }
            function bar() {
               console.log('bar()'+data)
            }

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

3. 模块化规范

---

1). 常见的模块化规范

```
1. CommonJS
2. AMD
3. CMD
4. ES6
```

2). CommonJS(掌握)

1. 实现
   服务器端: Node.js
   浏览器端: Browserify
2. 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')
3. 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

3). AMD(了解)

```
1. 实现
   浏览器端: require.js(常称:requireJS)
2. 基本语法
   定义暴露模块:
      define([依赖模块名], function(){
        return 模块
      })
   引入模块:
      require(['模块1', '模块2'], function(m1, m2){
        //使用m1与m2
      })
   配置:
      //配置
      require.config({
        //基本路径
        baseUrl: 'src/',
        //映射: 模块标识名: 路径
        paths: {
          //自定义模块
          'a': 'modules/a',  // 不能加后缀
          'b': 'modules/b',
          //第三方库
          'jquery': 'libs/jquery-1.10.1',
        }
      })
```

4). CMD(了解)

```
1. 实现
    浏览器端: sea.js(常称:seaJS)
2.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'])
```

5). ES6(掌握)

```
1. ES6内置了模块化的实现
2. 基本语法
    * 定义暴露模块: export
        默认暴露(暴露一个数据):
            export default 对象
        一般暴露(暴露多个数据):
            export const a = value1
            export let b = value2

            const c = value1
            let d = value2
            export {c, d}

    * 引入使用模块: import
        引入default模块:
            import xxx  from '模块路径/模块名'
        引入一般模块
            import {a, b} from '模块路径/模块名'
            import * as module1 from '模块路径/模块名'
3. 问题:
    所有浏览器还不能直接识别ES6模块化的语法
4. 解决:
    使用webpack编译打包
```

4. 详细教程

- 01_模块化进化史
- 02_CommonJS规范_Node
- 03_CommonJS规范_Browserify
- 04_AMD规范_RequireJS
- 05_CMD规范_SeaJS
- 06_ES6模块化_Webpack