1.commonJS

1.1说明:

- 每一个js文件都可以当成一个模块,
- 在服务器端,模块的加载是运行时同步加载的(node 是基于commonjs编写的)
- 在浏览器端,模块需要提前编译打包处理(浏览器不认识require()语法)

1.2 基本语法:

暴露模块:1.export.xxx=value;2.module.exports=value

暴露的模块到底是什么? 上面两个方式暴露的都是exports对象.

引入模块:

require(xxx):

1.第三方模块:xxx为模块名,(npm保命)

2.自定义模块:xxx为模块文件路径.

1.3 实现

1.3.1 服务端实现

Node.js

1.3.2 浏览器端实现

Browserify

也成为Commonjs的浏览器端的打包工具

打包命令:

```
browserify ./js/src/app.js -o ./js/dist/boundle.js
```

ES6

1.3.3 区别Node与Browerify

2.AMD

2.1 规范

说明:专门用于浏览器端,模块的加载是异步的,依赖requirejs

定义暴露模块

```
//定义没有依赖的模块
define(function(){
   return 模块 //通过return暴露模块
})

//定义有依赖的模块,下面的语法方式是
define(['module1','module2'],function(m1,m2){
```

```
    return 模块
})
```

引入使用模块

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

2.2 实现(浏览器端)

需要Require.js

3.CMD

规范:(common module definition,通用模块定义)

seajs

说明:专门用于浏览器端,模块加载是异步的,模块使用时才会加载执行

基本语法

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
})


//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
})
```

引入使用模块

```
define(function(require){
    var m1 = require("./module1")
    var m2 = require("./module4")
    m1.show()
    m2.show()
})
```

实现(浏览器)

Sea.js

4.ES6

4.1 规范

依赖模块需要编译打包处理(部分浏览器可能不支持)

语法:

导出模块:export:

引入模块:import:

4.2 实现(浏览器端)

使用Babel将ES6编译为ES5代码

使用Browserify编译打包js

使用Babel编译:

```
Babel js/src -d js/lib
```

使用Browserify编译js:

```
browserify js/lib/app.js -o js/lib/bundle.js
```