**nodeJS**

[官方] Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。Node.js 的包管理器 npm，是全球最大的开源库生态系统。基于ECMAScript扩展的语言，内置扩展了如：文件系统、网络、操作系统、系统进程线程、HTTP等等模块，我们还可以通过安装第三方的模块来操作数据库、邮件、图片等等。

**块级作用域**：为了更好组织、管理代码，nodejs引入了模块，一个文件就是一个独立的模块，每个模块都有自己的独立作用域 - 模块作用域。

**exports** : 对象，通过这个对象，可以把一个模块作用域下的变量、函数等模块内的数据，对外进行提供访问。

**require**：返回值：返回的就是被引用的模块的exports对象。

var ex = require('./1.js'); 加载同目录下的1.js文件

**模块分类:**

**1.核心模块**Core Modules：

就是node原生提供的模块

加载 require('模块的名称'); ，没有路径

**var fs = require('fs');**

**2.文件模块** File Modules：

以单独的文件形式存在的模块

文件模块的加载一定需要路径，就算是当前目录，也需要加路径

. /表示当前目录 . . /表示上级目录 /表示根目录

var f1 = require('./f1.js');

**3.目录模块**Folder Modules**：**

以目录（固定格式）的形式存放的模块

目录模块的加载和核心模块的加载是一样的写法

var miaov = require('miaov');

寻找方法：当我们去加载的模块没有路径的时候，首先会在当前目录的文件夹中去查找对应的目录，在通过一个json的文件查找到配置的模块，如果没有就会找到父级的目录的文件夹目录下，一直找到根目录，如果根 ，目录还没有，则找核心模块，如果再没有，报错

**Process全局内置模块**

Process：是一个全局对象，可以在任何地方访问，提供了当前程序的经常的相关操作

process.argv：程序运行时的参数，通过运行程序的时候传入

process.exit(): 退出当前进程

**标准输入输出流** 标准输入流是从标准输入设备（键盘）流向程序的数据 标准输出流是从程序流向标准输出设备（屏幕）的数据

Process.stdout标准输出流对象Process.stdout.write(data <,charset> <,callback>)向标准输出设备写入数据

Process.stdin 标准输入流对象，使用时先开启

Process.openStdin() 开启标准输入流

Process.stdin.on('data', function(chunk){})

监听输入流的data事件 当输入回车的时候触发该事件 chunk为接收到的数据块，默认为Buffer数据

**Buffer全局内置模块**

**Buffer** 使用Buffer对象，我们可以更好的操作二进制数据 类数组 存储二进制数据

Buffer对象的创建 new Buffer(size) //创建一个长度为size的Buffer new Buffer(string ) // 将一个字符串转成Buffer new Buffer(array) //讲数组的所有值转成Buffer new Buffer(buffer) //

当一个Buffer对象被创建以后，其内容长度是不可变的

var bf1 = new Buffer(5);

bf1.fill(0); //设置Buffer的默认数据

bf1[0] = 0x61;

bf1[1] = 0x62;

bf1[6] = 0x62;

console.log(bf1.toString()); // toString 将Buffer数据转为字符串数据

**Buffer.write( str, num , num)**  参数一：要填充的字符串； 参数二：数字，从第几位才是填充；参数三：数字，填充几位字符；

var bf4 = new Buffer(5);

bf4.write('abcefg', 1,2);

console.log(bf4); //<Buffer 00 61 62 00 14>

console.log(bf4.toString()); //ab

**Buffer.toJSON()** 将一个Buffer转成一个特殊的字符JSON

var bf5 = new Buffer('miaov');

console.log( bf5.toJSON() );//{ type: 'Buffer', data: [ 109, 105, 97, 111, 118 ] }

**Buffer.byteLength(str)** 字节长度

console.log( Buffer.byteLength('miaov') ); //5

console.log( Buffer.byteLength('妙味') ); //6

**fs内置模块**

文件系统模块

提供了对文件、文件的相关操作

创建、重命名、删除、获取文件（文件夹）信息等

这是非全局的核心模块，需要引入 ，**Var fs = require('fs')** 内置模块的引用加载，不需要写路径

**操作文件方法：**

Var fs = require('fs') ;

1. **fs.writeFile**('./data/1.txt', 'miaov',callback); 改 向指定的文件中写入数据, 如果文件不存在，则是创建

2. **fs.appendFile**('./data/2.txt', '妙味'); 增 向指定的文件中添加数据，如果文件不存在，则是创建

3. **fs.readFile**('./data/1.txt', function(err, data) { // 读取指定的文件中添加数据，如果文件不存在会有一个失败的err

if (err) {

console.log('读取文件失败')

} else {

console.log(data.toString());

}

})

try { //这种方法与上面方法一样

var content = fs.readFileSync('./data/1.txt');

console.log(content.toString());

} catch(e) {

console.log('文件读取失败');

}

Error-First

错误优先回调模式

两条规则：

1.回调函数的第一个参数保留给一个Error对象，如果有操作发生，错误信息将通过回调函数的第一个参数返回

2.回调函数的第二个参数为成功返回的数据，如果没有错误发生，err（第一个参数）讲设置为null，成功返回的数据将通过第二个参数返回

4. fs.unlink('./data/1.txt' , callback); 删除文件

以上方法均为异步操作，如果转换为同步就要在后方加Sync，例如fs.writeFileSync(),以上四个方法均适用

**操作文件夹方法：**

Var fs = require('fs') ;

1. **fs.mkdirSync**('./data/miaov'); //同步的创建一个文件夹
2. var fileList = **fs.readdirSyn**c('./data/'); //同步的读取一个文件夹内的所有文件的名称，将所有名称放到一个数组中，并返回。

console.log(fileList) // [ '1', '1.jpg', '1.txt', 'a', 'b', 'miaov' ]

3. var stat = **fs.statSync**('./data/miaov); 返回一个fs.Stats对象

console.log(stat) // { dev: 350249129, mode: 33206, nlink: 1, uid: 0, gid: 0,rdev: 0 blksize: undefined, ino: 281474976719228, size: 36, blocks: undefined, atime: 2016-08-01T09:44:17.887Z,mtime: 2016-08-01T08:33:44.181Z, ctime: 2016-08-01T09:44:17.887Z, birthtime: 2016-08-01T09:44:17.887Z }

**Stats方法：**

.**isDirectory():** fs.statSync('./data/miaov).**isDirectory() 判断是不是一个文件夹 是的话返回true，反之false**

**.isFile()：** fs.statSync('./data/miaov)**.isFile()**  **判断是不是一个单一文件 是的话返回true，反之false**

var fileList = fs.readdirSync('./data');

for (var i=0; i<fileList.length; i++) {

var stat = fs.statSync('./data/' + fileList[i]);

if (stat.isDirectory()) {

console.log('文件夹：' + fileList[i]);

}

if (stat.isFile()) {

console.log('文件：' + fileList[i]);

}

}

1. **fs.rmdirSync**('./data/miaov'); 删除一个文件夹，但不能删除里面还有文件的文件夹

**fs.existsSync**('./data/miaov)判断是否有这个文件夹