1. fs模块 （文件操作）.

！！要把**cmd**的位置放在工作目录中

文件包含文件和目录，统称为文件

fs.stat( path, callback )查看文件的状态

path 文件的路径

callback 回调函数，用于获取结果

err 可能产生的错误信息

stats 具体的文件状态

//引入fs模块

const fs=require('fs');

//查看文件的状态

//参数1：路径

//参数2：回调函数，获取结果，不通过保存至变量了

fs.stat('global.js',function(err,stats){

if (err) throw err;

console.log(stats); //err：可能产生的错的 stats：具体的结果

});

//var rs = statSync('lobal.js'); 在主程序执行，不在事件队列

1. 同步（Sync）和 异步

**同步：**等待前边的代码执行完才能执行后边的，会阻止后边代码的执行（如果保错），通过返回值来回去结果（需要保存在变量或打印）

**异步：**不会阻止后边代码执行，后边操作可以先执行，通过回调函数来获取结果。

目录操作：

1）fs.stat(path,callback) / statSync(path) 查看文件状态

2）fs.mkdir(path,callback)/mkdirSync(path) 创建目录

3）fs.rmdir(path,callback)/fs.rmdirSync(path) 移除目录（前提是文件夹里面要是空的）

4）fs.readdir( path,callback(err,result) )/fs.readdirSync(path)读取目录

result 读取的结果，格式为数组

例：fs.readdir('mydir',function(err,result){ //result读取的文件列表

if(err) throw err;

console.log(result);

})

1. fs.writeFile(path , data , callback ) 创建文件/写入文件

fs.writeFileSync(path,data)

data指写入的数据

若文件不存在，会创建并写入数据

若已经存在，会清空内容然后写入数据，即 清空写入。

例：fs.writeFile('1.txt','吔屎啦你',function(err){

if(err) throw err;

console.log('文件创建成功');

})

1. fs.appendFile(path , data , callback ) 追加写入文件

fs.appendFileSync(path,data)

若文件不存在，会创建文件并写入数据

若文件已存在，则在文件末尾追加写入数据

例：

//创建数组，每个元素是一个课程名称，遍历数组，分别获取元素，将数据使用同步方法写入data.txt

var kecheng = ['语文','数学','英语','理综'];

for(var key in kecheng){

fs.appendFileSync('data.txt',kecheng[key] + '\n');

}

1. fs.readFile(path,callback(err,data))读取文件

fs.readFileSync(path)

data 读取处理的结果是buffer形式

fs.readFile('data.txt',function(err,data){

if (err) throw err;

console.log(data.toString());

})

1. fs.unlink(path,callback)/fs.cunlinkSync(path) 删除文件
2. fs.existsSync(path) 判断文件是否存在 只有同步
3. 文件流

var readStream = fs.createReadStream(文件路径) 创建可读流

readStream.on('data',function(chunk){ })

有数据流入时，自动触发事件

通过回调函数来获取，chunk就是获取的数据流

readStream.on('end',function(){ })

当服务结束时，自动触发事件

pipe 管道 可将数据从可读流添加到可写流