一文件操作

# 1.fs的写入操作

## 1.异步写入文件

|  |
| --- |
| let fs = require('fs')  //1.写入文件  let data="三人同行，必有我师"  fs.writeFile("座右铭.txt",data,(err)=>{      console.log(err);  }) |
|  |

## 2.同步写入文件

|  |
| --- |
| let fs = require('fs')  //写入文件  let data="三人同行，必有我师"  fs.writeFileSync("zym.txt",data) //同步写入，没有返回值 |
|  |

### 注意:异步操作的效率更高

# 2.fs的追加写入

## 1）异步追加，其实和写入很像，注意是追加

|  |
| --- |
| let fs = require('fs')  //1.写入文件  let data="三人同行，必有我师"  // fs.writeFile("座右铭.txt",data,(err)=>{  //     console.log(err);  // })  // fs.writeFileSync("zym.txt",data) //同步写入，没有返回值  //2.追加写入  data="\n好好学习，天天向上"  fs.appendFile("座右铭.txt",data,(err)=>{          console.log(err);  }) |

## 2）同步追加

|  |
| --- |
| let fs = require('fs')  //1.写入文件  let data="三人同行，必有我师"  // fs.writeFile("./座右铭.txt",data,(err)=>{  //     console.log(err);  // })  // fs.writeFileSync("./zym.txt",data) //同步写入，没有返回值  //2.1.追加写入,异步  // data="\n好好学习，天天向上"  // fs.appendFile("./座右铭.txt",data,(err)=>{  //         console.log(err);  // })  //2.2 同步追加  data = "\n各人自扫门前雪，哪管他人瓦上霜"  fs.appendFileSync("./zym.txt",data) |

### 注意：追加也可以使用writeFile来实现

|  |
| --- |
| let fs = require('fs')  //1.写入文件  let data="三人同行，必有我师"  // fs.writeFile("./座右铭.txt",data,(err)=>{  //     console.log(err);  // })  // fs.writeFileSync("./zym.txt",data) //同步写入，没有返回值  //2.1.追加写入,异步  // data="\n好好学习，天天向上"  // fs.appendFile("./座右铭.txt",data,(err)=>{  //         console.log(err);  // })  //2.2 同步追加  data = "\n各人自扫门前雪，哪管他人瓦上霜"  fs.appendFileSync("./zym.txt",data)  data ="\n酒不醉人人自醉"  //扩展：用writeFile函数来追加内容  fs.writeFile("./座右铭.txt",data,{flag:'a'},err=>{      if(err){          console.log(err);          return      }      console.log("追加成功");  }) |

# 3.流式写入

|  |
| --- |
| let fs = require('fs')  let ws = fs.createWriteStream("./stream.txt")  ws.write("善有善报\n")  ws.write("恶有恶报\n")  ws.write("不是不报\n")  ws.write("时候未到\n")  ws.close() |
|  |

## 流式写入的好处

|  |
| --- |
|  |

## 4.文件读取

## 1）异步读取

|  |
| --- |
| let fs = require('fs')  //异步读取  fs.readFile('./座右铭.txt',(err,data)=>{      if(err){          console.log(err);  return      }      console.log("文件内容:\n",String(data));//也可以使用data.toString()  }) |
|  |

## 2）同步读取

|  |
| --- |
| let fs = require('fs')  //同步读取  let data = fs.readFileSync('./座右铭.txt')  console.log("文件内容:\n",String(data)); |
|  |

# 5.流式读取文件内容,

|  |
| --- |
| let fs = require('fs')  let rs = fs.createReadStream('./座右铭.txt')  rs.on('data',(chunk)=>{ //每一次默认读取64kb      console.log(chunk.toString());  })  rs.on('end',()=>{      console.log("读取完毕");  }) |
|  |

## 这个读取流也是一个可迭代对象，它有一个forEach方法

|  |
| --- |
| let fs = require('fs')  let rs = fs.createReadStream('./座右铭.txt')  // rs.on('data',(chunk)=>{ //每一次默认读取64kb  //     console.log(chunk.toString());  // })  // rs.on('end',()=>{  //     console.log("读取完毕");  // })  rs.forEach(line=>{      console.log(String(line));  }) |
|  |

### 注意：不用调用readStream的close方法否则可能会抛异常或者拿不到数据

# 练习，实现文件复制功能

## 1.使用readFile和writeFile函数

|  |
| --- |
| let fs = require('fs')  //方式1，使用readFile  fs.readFile('./结构化思维.pdf',(err,data)=>{      if(err){          console.log(err);          return      }      fs.writeFile('./jghsw.pdf',data,err=>{          console.log(err);      })  }) |

## 2.使用流来复制1

|  |
| --- |
| let fs = require('fs')  // //方式1，使用readFile  // fs.readFile('./结构化思维.pdf',(err,data)=>{  //     if(err){  //         console.log(err);  //         return  //     }  //     fs.writeFile('./jghsw.pdf',data,err=>{  //         console.log(err);  //     })  // })  //方式2，使用流  let rs = fs.createReadStream('./结构化思维.pdf')  let ws = fs.createWriteStream('./jghsw2.pdf')  rs.on('data',chunk=>{      ws.write(chunk)  })  rs.on('end',()=>{      console.log("复制完成");  }) |

## 3.使用readStream的forEach方法和writeStream

|  |
| --- |
| let fs = require('fs')  let rs = fs.createReadStream('./结构化思维.pdf')  let ws = fs.createWriteStream('./jghsw2.pdf')  rs.forEach(line => {      ws.write(line)  }); |

### 4.使用rs.pipe方法

|  |
| --- |
| let fs = require('fs')  // //方式1，使用readFile  // fs.readFile('./结构化思维.pdf',(err,data)=>{  //     if(err){  //         console.log(err);  //         return  //     }  //     fs.writeFile('./jghsw.pdf',data,err=>{  //         console.log(err);  //     })  // })  //方式2，使用流  let rs = fs.createReadStream('./结构化思维.pdf')  let ws = fs.createWriteStream('./jghsw3.pdf')    rs.pipe(ws) |

### 注意：使用流的方式比较好

# 6.文件移动和重命名

|  |
| --- |
| let fs = require('fs')  fs.rename('./zmycpy.txt','./zym2.txt',err=>{       if(err){          console.log(err);          return       }       console.log("重命名成功");  }) |

## nodjs我文件移动也是用rename方法来实现的

|  |
| --- |
| let fs = require('fs')  fs.rename("./zym2.txt","./test/z.txt",err=>{      if(err){          console.log(err);          return      }      console.log("文件移动成功");  }) |

# 7.删除文件，

## 1）使用fs模块的unlink函数

|  |
| --- |
| let fs = require('fs')  fs.unlink('./zym copy.txt',err=>{      if(err){          console.log(err);          return      }      console.log('文件删除成功');  }) |

## 2）使用rm方法

|  |
| --- |
| let fs = require('fs')  //1.fs.unlink()  // fs.unlink('./zym copy.txt',err=>{  //     if(err){  //         console.log(err);  //         return  //     }  //     console.log('文件删除成功');  // })  //2.fs.rm  fs.rm('./zym copy.txt',err=>{          if(err){              console.log(err);              return          }          console.log('文件删除成功');  }) |

二、文件夹操作

# 1.创建文件夹，这里是异步方法，也同步方法xxSync

|  |
| --- |
| let fs = require('fs')  //1.创建文件夹  fs.mkdir("testFolder",(err)=>{      if(err){          console.log(err);          return      }      console.log('文件夹创建成功');  }) |

## 如果需要创建多级目录，需要在中间条件一个对象参数{recursive:true}

|  |
| --- |
| fs.mkdir("testFolder/test1/hello",{recursive:true},(err)=>{      if(err){          console.log(err);          return      }      console.log('文件夹创建成功');  }) |
|  |

# 2.读取文件夹, 这里是异步方法，也同步方法xxSync

|  |
| --- |
| //2.读取文件夹的内容  fs.readdir('./testFolder',{recursive:true},(err,files)=>{      if(err){          console.log(err);          return      }      console.log(files);  }) |

# 3.删除空文件夹, 这里是异步方法，也同步方法xxSync

|  |
| --- |
| fs.rmdir('./testFolder/test',(err)=>{      if(err){          console.log(err);          return      }      console.log("删除成功");  }) |

## 不能够删除多级目录，只能一级一级删除，

# 4.上面的方法只能删除空文件夹，非空文件夹需要自己写一个程序来实现，如下面的功能

## myFileUtils.js

|  |
| --- |
| let fs = require('fs')  //删除非空目录的函数  function deleteFolder(filePath) {      let files = []      if(fs.existsSync(filePath)){         files = fs.readdirSync(filePath)         files.forEach(file=>{           //构建完整路径，因为readdirSync得到的是最后一级路径           let fullPath = `${filePath}/${file}`           //判断是文件还是文件夹           let stats = fs.statSync(fullPath)             if(stats.isDirectory()){               //是文件夹就要遍历               deleteFolder(fullPath)           } else {              //是文件就直接删除              fs.rmSync(fullPath)           }        })         fs.rmdirSync(filePath)      }      console.log("删除完成...");  }  module.exports = deleteFolder |

## 然后在folderdemo.js中导入这个函数来使用

|  |
| --- |
| let fs = require('fs')  let deleteFolder = require('./myFileUtils')  deleteFolder("./wjj1") |

# 5. nodejs建议你使用rm方法来输出多级目录可以把文件和文件夹一起删除

## 如我们新一个有mydir目录里面有一个file1文件和一个dir1文件夹，dir1文件夹里面有一个dir2文件夹，dir2文件夹里面有一个file2文件

|  |
| --- |
|  |

## 运行下面的代码

|  |
| --- |
| let fs = require('fs')  fs.rm('./mydir',{recursive:true},err=>{      if(err){          console.log(err);          return      }      console.log("finished...");  }) |

### 效果：mydir这个文件夹和里面的所有文件和子文件夹都被删除了

|  |
| --- |
|  |

三、fs模块查看资源状态

# 1.fs.stat方法-异步

|  |
| --- |
|  |

## 小案例

|  |
| --- |
| const { log } = require('console')  let fs = require('fs')  fs.stat('./filesop',(err,data)=>{      if(err){          log("err")          return      }      log(data)  }) |
| **结果：**  Stats {  dev: 3629183472,  mode: 16822,  nlink: 1,  uid: 0,  gid: 0,  rdev: 0,  blksize: 4096,  ino: 1688849861167807,  size: 0,  blocks: 8,  atimeMs: 1708379698978.8975,  mtimeMs: 1708361649123.1953,  ctimeMs: 1708361649123.1953,  birthtimeMs: 1708361610889.2927,  atime: 2024-02-19T21:54:58.979Z,  mtime: 2024-02-19T16:54:09.123Z,  ctime: 2024-02-19T16:54:09.123Z,  birthtime: 2024-02-19T16:53:30.889Z  } |

# 2.fs.statSync方法-同步

## 小案例

|  |
| --- |
| const { log } = require('console')  let fs = require('fs')  // fs.stat('./filesop',(err,data)=>{  //     if(err){  //         log("err")  //         return  //     }  //     log(data)  // })  let sta = fs.statSync('./folderop')  log(sta) |

## 效果

|  |
| --- |
|  |

## stat方法返回的对象有很多属性和方法，其中比较重要的是isFile()和isDirectory()

|  |
| --- |
| let sta = fs.statSync('./folderop')  log(sta)  log(sta.isFile()) //false  log(sta.isDirectory()) //true |

### 效果

|  |
| --- |
|  |

# fs模块相对路径的bug的解决办法

|  |
| --- |
| 在开发nodejs中，我们往往需要读取文件或者写入文件，最常用的模块就是fs核心模块。一个最简单的读取文件的代码如下：  **fs.readFile("./test.txt", function (error, result) {} )**  ​ 该代码的意思就是从当前目录下的test.txt文件中读取数据，然后在result中打印输出。当在我们本地运行该代码应该没什么问题，可以成功执行。但是如果这样的代码在项目中运行，同时把该项目部署到云服务器上时，很有可能会出现bug。bug的提示就是无法在该目录找到test.txt文件。由于**运行环境的不同**，以上的相对路径的写法导致最后读取的位置是不同的。回调函数也是如此。  ​ 正确的写法应该是使用"path.join()"的方式实现：(\_\_dirname表示当前文件的目录名)  **fs.readFile(path.join(\_\_dirname, "./test.txt"), function (error, result) {} )**  ​ 这样的实现在不同机子的环境上都能成功运行。 |
|  |

### 这样子就可以保证不管你在那里运行这个文件，它始终在指定的绝对路径里面创建文件

# 练习：批量文件重命名

## 1，新建一个exercise文件夹，然后新建一个js文件和10个txt文件，

|  |
| --- |
|  |

## 2.我们新建一个myfile文件夹，然后把这些提醒他文件移动到里面

|  |
| --- |
|  |

## 3. 我们在js我们里面吧1-9的txt文件改为01-09开头，代码如下

## 我的版本

|  |
| --- |
| let fs = require('fs')  let path = require('path')  let files = fs.readdirSync(\_\_dirname+'/myfile')  // console.log(files);  files.forEach(file =>{      let prefix =file.split('-')[0]      let remain = file.split('-')[1]      // console.log(prefix);      if(prefix.length<2){          prefix = '0' + prefix      }      let fullname = path.join(\_\_dirname,'/myfile/'+prefix+'-'+remain)      let oldname = path.join(\_\_dirname,'/myfile/'+file)      fs.renameSync(oldname,fullname)  }) |

### 效果

|  |
| --- |
|  |

## 老师的写法

|  |
| --- |
| let fs = require('fs')  let files = fs.readdirSync('./myfile')  files.forEach(file =>{      let [pre,name] = file.split('-')      if(Number(pre)<10){          pre = '0'+pre      }      let newName = pre+'-'+name      // console.log('./myfile/'+newName);      fs.renameSync( `./myfile/${file}`,`./myfile/${newName}`)  }) |

## 效果

|  |
| --- |
|  |

# 练习2.我们把上面的myfile文件夹里面的02-file2.txt删除，然后要求把03-10文件的开头依次变为03-09

|  |
| --- |
|  |

## 代码如下

|  |
| --- |
| /练习，批量文件重命名,删除了02开头的文件，然后把03-10的文件改为以02-09开头  let fs = require('fs')  let files = fs.readdirSync('./myfile')  //1.01文件是不用处理的我们把03-10开头的文件过滤出来  let newFiles = files.filter(item=>!item.startsWith('01'))  // console.log(newFiles); //得到03-10开头的文件  newFiles.forEach(file=>{      //得到开头和'-'后面的部分      let [pre,name] = file.split('-')      let newpre = `0${Number(pre)-1}`      // console.log(newpre);//得到02-09      let newname = newpre+ '-'+name //得到新文件名      // console.log(newname);      fs.renameSync( `./myfile/${file}`,`./myfile/${newname}`)  }) |

### 效果

|  |
| --- |
|  |