fs 文件系统模块

什么是fs 文件系统模块

fs模块是Node.js官方提供的、用来操作文件的模块。它提供了一系列的方法和属性，用来满足用户对文件的操作需求。

例如

- `fs.readFile()`方法，用来读取指定文件中的内容
- `fs.writeFile()`方法，用来向指定的文件中写入内容

如果要在JavaScript 代码中，使用fs模块来操作文件，则需要使用如下的方式先导入它:

```
const fs = require('fs')
```

读取文件

`fs.readFile()`的语法格式

使用 `fs.readFile()`方法，可以读取指定文件中的内容，语法格式如下:

```
fs.readFile(path[, options], callback)
//[]为可选参数，其余为必选参数
```

- `path`：文件路径
- `options`：配置选项，若是字符串则指定编码格式
  - `encoding`：编码格式
  - `flag`：打开方式
- `callback`：回调函数
  - `err`：错误信息
  - `data`：读取的数据，如果未指定编码格式则返回一个 Buffer

实例：以utf8的编码格式，读取指定文件的内容，并打印err和dataStr的值

```
const fs = require( 'fs ')
fs.readFile('./files/11.txt','utf8',function(err, dataStr){
    console.log(err)
    console.log( '-----')
    console.log(dataStr)
})
```

判断文件是否读取成功 可以判断err 对象是否为null，从而知晓文件读取的结果:

```
const fs = require('fs')
fs.readFile('./files/1.txt ','utf8',function(err, result) {
    if (err) {
        return console.log('文件读取失败!' + err.message)
    }
    console.log('文件读取成功，内容是：' + result)
})
// 如果没有return 后面还会执行console.log('文件读取成功，内容是：' + result)
```

写入文件

fs.writeFile()的语法格式

使用 fs.writeFile()方法，可以向指定的文件中写入内容，语法格式如下:

```
    fs.writeFile(file, data[, options], callback)
    //[]为可选参数，其余为必选参数
```

- file：文件路径
- data：写入内容
- options：配置选项，包含 encoding, mode, flag；若是字符串则指定编码格式,默认是utf8
- callback：回调函数

实例：向指定的文件路径中，写入文件内容。

```
const fs = require('fs')
fs.writeFile( './files/2.txt','Hello Node.js ! ',function(err) {
    console.log(err)
})
// 如果文件写入成功，则err的值等于null
```

练习-考试成绩整理

整理要求

使用fs文件系统模块，将素材目录下成绩.txt文件中的考试数据，整理到成绩-ok.txt文件中。

整理前，成绩.txt文件中的数据格式如下:

> 小红=99 小白=100 小黄=70 小黑=66 小绿=88

整理完成之后，希望得到的成绩-ok.txt文件中的数据格式如下:

> 小红: 99 小白: 100 小黄: 70 小黑: 66 小绿: 88

核心实现步骤

1. 导入需要的fs文件系统模块
2. 使用`fs.readFile()`方法，读取素材目录下的`成绩.txt`文件
3. 判断文件是否读取失败
4. 文件读取成功后，处理成绩数据
5. 将处理完成的成绩数据，调用`fs.writeFile()`方法，写入到新文件`成绩-ok.txt`中

```
//导入fs模块
const fs = require( 'fs')
//调用fs.readFile()读取文件的内容
fs.readFile('成绩.txt','utf8',function(err,dataStr){
    //判断是否读取成功
    if (err) {
        return console.log('读取文件失败!' + err. message)
    }
    //console.log('读取文件成功!' + dataStr)
    //先把成绩的数据，按照空格进行分割
    const arrOld = dataStr.split(' ')
    console.log(arrOld)
    //循环分割后的数组，对每一项数据，进行字符串的替换操作
    arrNew = []
    arrOld.forEach(item =>{
        arrNew.push(item.replace('=',': '))
    })
    console.log(arrNew)
    // 4.3把新数组中的每一项，进行合并，得到一个新的字符串
    const newStr = arrNew.join('\r\n')
    console.log(newStr)
    // 调用fs.writeFile()方法，把处理完毕的成绩，写入到新文件中
    fs.writeFile('成绩-ok.txt',newStr,function(err){
        if (err) {
            return console.log('写入文件失败!' + err.message)
        }
        console.log('成绩写入成功!')
    })
})
```

fs模块–路径动态拼接的问题

在使用fs模块操作文件时，如果提供的操作路径是以./或../开头的相对路径时，很容易出现路径动态拼接错误的问题。

原因:代码在运行的时候，会以执行node命令时所处的目录，动态拼接出被操作文件的完整路径。

解决方案:在使用fs模块操作文件时，直接提供完整的路径，不要提供./或../开头的相对路径，从而防止路径动态拼接的问题。或使用`__dirname`表示当前文件所处的目录

其它操作

验证路径是否存在：

- `fs.exists(path, callback)`
- `fs.existsSync(path)`

获取文件信息：

- `fs.stat(path, callback)`
- `fs.stat(path)`

删除文件：

- `fs.unlink(path, callback)`
- `fs.unlinkSync(path)`

列出文件：

- `fs.readDir(path[,options], callback)`
- `fs.readDirSync(path[, options])`

截断文件：

- `fs.truncate(path, len, callback)`
- `fs.truncateSync(path, len)`

建立目录：

- `fs.mkdir(path[, mode], callback)`
- `fs.mkdirSync(path[, mode])`

删除目录：

- `fs.rmdir(path, callback)`
- `fs.rmdirSync(path)`

重命名文件和目录：

- `fs.rename(oldPath, newPath, callback)`
- `fs.renameSync(oldPath, newPath)`

监视文件更改：

- `fs.watchFile(filename[, options], listener)`