1. fs.stat 检测是文件还是目录
2. fs.exists 判断文件、目录是否存在

2. fs.mkdir 创建目录

3. fs.writeFile 创建写入文件

4.fs.appendFile 追加文件

5.fs.readFile 读取文件

6.fs.readdir 读取目录

7.fs.rename 重命名

8. fs.rmdir 删除目录

9. fs.unlink 删除文件

10. fs.createReadStream 从文件流中读取数据

11. fs.createWriteStream 写入文件

12.管道流

13.使用管道流上传文件

**1. fs.stat 检测是文件还是目录**

const fs = require('fs');

fs.stat('upload.js',(err,stats) =>{

if (err){

console.log(err);

} else {

console.log(stats);

console.log('文件：'+stats.isFile());

console.log('目录：'+stats.isDirectory());

}

})

1. **fs.exists 判断文件、目录是否存在**

断的方法

fs.exists(path, callback)

path：判断的文件夹、文件的路径  
callback：回调函数

Fs.exists(“dirName”,function(exits){

Console.log(exist? ‘创建成功’ :’失败’);

})

**2. fs.mkdir 创建目录**

fs.mkdir('logs',(err)=>{

if (err){

console.log(err);

} else {

console.log('目录创建成功');

}

})

**3. fs.writeFile 创建写入文件**

//如果文件不存在，则创建，内容被覆盖

fs.writeFile('logs/test.txt','写入一些内容',(err)=>{

if (err){

console.log(err);

} else {

console.log('写入成功');

}

})

**4.fs.appendFile 追加文件**

fs.appendFile('logs/test.txt','追加内容\n',(err)=>{

if (err){

console.log(err);

} else {

console.log('追加成功！');

}

})

**5.fs.readFile 读取文件**

fs.readFile('logs/test.txt','utf8',(err,datas)=>{

if (err){

console.log(err);

} else {

console.log(datas);

}

})

**6.fs.readdir 读取目录**

读取该目录下的目录名和文件名

fs.readdir('upload',(err,df)=>{

if (err){

console.log(err);

} else {

console.log(df);

}

})

**7.fs.rename 重命名**

7.1重命名文件

fs.rename('logs/test.txt','logs/temp.txt',(err)=>{

if (err){

console.log(err);

} else {

console.log('重命名成功！');

}

})

7.2复制文件

fs.rename('logs/temp.txt','upload/temp.txt',(err)=>{

if (err){

console.log(err);

} else {

console.log('复制成功！');

}

})

**8. fs.rmdir 删除目录**

注意：目录下不能有文件

fs.rmdir('logs/temp',(err)=>{

if (err){

console.log(err);

} else {

console.log('移除成功');

}

})

**9. fs.unlink 删除文件**

fs.unlink('logs/ass.txt',(err)=>{

if (err){

console.log(err);

} else {

console.log('文件已删除');

}

})

**10. fs.createReadStream 从文件流中读取数据**

var fileReadStream = fs.createReadStream('upload/temp.txt');

var str = ''

//分块读取

fileReadStream.on('data',(chunk)=>{

str+= chunk;

})

// 读取完成

fileReadStream.on('end',()=>{

console.log('读取完成:'+str);

})

//读取失败

fileReadStream.on('error',(err)=>{

console.log(err);

})

**11. fs.createWriteStream 写入文件**

var conent = '一些内容描述';

//创建一个可以写入到指定文件的流

var writeStream = fs.createWriteStream('logs/test3.txt');

//使用utf8编码写入文件

writeStream.write(conent,'utf8');

//标记文件末尾

writeStream.end();

//数据已写入时候触发

writeStream.on('finish',()=>{

console.log('写入完成');

})

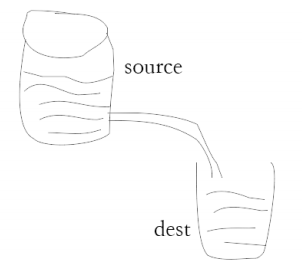
writeStream.on('error',(err)=>{

console.log(err);

})

**12. 管道流**

管道提供了一个输出流到输入流的机制。通常我们用于从一个流中获取数据并将数据传 递到另外一个流中。



如上面的图片所示，我们把文件比作装水的桶，而水就是文件里的内容，我们用一根管子(pipe)连接两个桶使得水从一个 桶流入另一个桶，这样就慢慢的实现了大文件的复制过程。

//创建一个可读流

var readStream = fs.createReadStream('upload/temp.txt');

//创建一个可写流

var writeStream = fs.createWriteStream('upload/output.txt');

//管道读写操作

readStream.pipe(writeStream);

**13.使用管道流上传文件**

前端表单

<form name="myform" method="post" action="upload" enctype="multipart/form-data">

用户名：<input type="text" name="username" id="username">

<br>

<input type="file" name="file" id="file" />

<br>

<input type="submit" name="" value="提交">

</form>

//引入模块

const Koa = require('koa'),

router = require('koa-router')(),

koaBody = require('koa-body'),

bodyParser = require('koa-bodyparser'),

render = require('koa-art-template'),

path = require('path'),

static = require('koa-static'),

fs = require('fs');

const app = new Koa();

////////////////////////////////////////////////////////////////////

//配置模板引擎koa-art-template

render(app, {

root: path.join(\_\_dirname, 'views'),

extname: '.html',

debug: process.env.NODE\_ENV !== 'production'

});

//静态资源

app.use(static('statics'));

//中间件bodyParser

app.use(bodyParser());

//中间件koaBody

app.use(koaBody({

multipart: true,

formidable: {

maxFileSize: 200\*1024\*1024 // 设置上传文件大小最大限制，默认2M

}

}));

///////////////////////////////////////////////////////////////////

router.get('/', async (ctx)=>{

await ctx.render('upload');

})

router.get('/detail/:aid/:key', async (ctx)=>{

console.log(ctx.params);

})

router.post('/upload', async (ctx) =>{

let datas = ctx.request.body;

// 上传单个文件

const file = ctx.request.files.file; // 获取上传文件

console.log(file);

// 创建可读流

const reader = fs.createReadStream(file.path);

let filePath = path.join(\_\_dirname, 'upload/') + file.name;

// 创建可写流

const upStream = fs.createWriteStream(filePath);

// 可读流通过管道写入可写流

reader.pipe(upStream);

ctx.body = datas.username+"上传成功！";

})

////////////////////////////////////////////////////////////////////

//开启路由

app

.use(router.routes())

.use(router.allowedMethods());

//监听

app.listen(3015,()=>{

console.log('Server running at http://localhost:3015');

})