Nodejs 中的 fs 模块的使用

主讲教师：（大地）
合作网站：www.itying.com （IT 营）
我的专栏：https://www.itying.com/category-79-b0.html

目录

1. fs.stat 检测是文件还是目录

```
const fs = require('fs')
fs.stat('hello.js', (error, stats) =>{
  if(error){
    console.log(error)
  } else {
    console.log(stats)
    console.log(`文件：${stats.isFile()}`)
    console.log(`目录：${stats.isDirectory()}`)
  }
})
```

2. fs.mkdir 创建目录

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


fs.mkdir('logs', (error) => {
  if(error){
    console.log(error)
  } else {
    console.log('成功创建目录：logs')
  }
})
```

3. fs.writeFile 创建写入文件

```
fs.writeFile('logs/hello.log', '您好 ~ \n', (error) => {
  if(error) {
    console.log(error)
  } else {
    console.log('成功写入文件')
  }
})
```

4. fs.appendFile 追加文件

```
fs.appendFile('logs/hello.log', 'hello ~ \n', (error) => {
  if(error) {
    console.log(error)
  } else {
    console.log('成功写入文件')
  }
})
```

5.fs.readFile 读取文件

```
const fs = require('fs')
fs.readFile('logs/hello.log', 'utf8', (error, data) =>{
  if (error) {
    console.log(error)
  } else {
    console.log(data)
  }
})
```

6.fs.readdir 读取目录

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


fs.readdir('logs', (error, files) => {
  if (error) {
    console.log(error)
  } else {
    console.log(files)
  }
})
```

7.fs.rename 重命名

```
const fs = require('fs')
fs.rename('js/hello.log', 'js/greeting.log', (error) =>{
  if (error) {
    console.log(error)
  } else {
    console.log('重命名成功')
  }
})
```

8. fs.rmdir　删除目录

```
fs.rmdir('logs', (error) =>{
  if (error) {
    console.log(error)
  } else {
    console.log('成功的删除了目录：logs')
  }
})
```

9. fs.unlink 删除文件

```
fs.unlink(`logs/${file}`, (error) => {
  if (error) {
    console.log(error)
  } else {
    console.log(`成功的删除了文件: ${file}`)
  }
})
```

10. fs.createReadStream 从文件流中读取数据

```
const fs = require('fs')
var fileReadStream = fs.createReadStream('data.json')
let count=0;
var str='';
 fileReadStream.on('data', (chunk) => {
   console.log(`${ ++count } 接收到：${chunk.length}`);
   str+=chunk
```

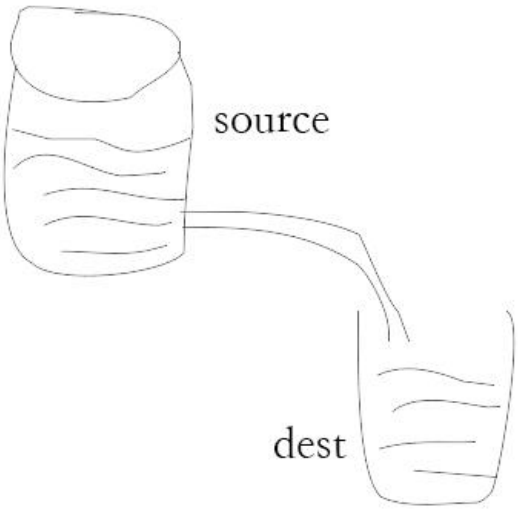
```
})
fileReadStream.on('end', () => {
  console.log('--- 结束 ---');
  console.log(count);
  console.log(str);
})

fileReadStream.on('error', (error) => {
  console.log(error)
})
```

11. fs.createWriteStream　写入文件

```
var fs = require("fs");
var data = '我是从数据库获取的数据，我要保存起来';
// 创建一个可以写入的流，写入到文件 output.txt 中
var writerStream = fs.createWriteStream('output.txt');
// 使用 utf8 编码写入数据
writerStream.write(data,'UTF8');
// 标记文件末尾
writerStream.end();
// 处理流事件 --> finish 事件
writerStream.on('finish', function() {   /*finish - 所有数据已被写入到底层系统时触发。*/
  console.log("写入完成。");
});
writerStream.on('error', function(err){
  console.log(err.stack);
});
console.log("程序执行完毕");
```

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。

以下实例我们通过读取一个文件内容并将内容写入到另外一个文件中。

```
var fs = require("fs");
// 创建一个可读流
var readerStream = fs.createReadStream('input.txt');
// 创建一个可写流
var writerStream = fs.createWriteStream('output.txt');

// 管道读写操作
// 读取 input.txt 文件内容，并将内容写入到 output.txt 文件中
readerStream.pipe(writerStream);
console.log("程序执行完毕");
```