stream

是Node.js提供的又一个仅在服务区端可用的模块，目的是支持“流”这种数据结构。

我们也可以把数据看成是数据流，比如你敲键盘的时候，就可以把每个字符依次连起来，看成字符流。这个流是从键盘输入到应用程序，实际上它还对应着一个名字：标准输入流（stdin）。

如果应用程序把字符一个一个输出到显示器上，这也可以看成是一个流，这个流也有名字：标准输出流（stdout）。流的特点是数据是有序的，而且必须依次读取，或者依次写入，不能像Array那样随机定位。

有些流用来读取数据，比如从文件读取数据时，可以打开一个文件流，然后从文件流中不断地读取数据。有些流用来写入数据，比如向文件写入数据时，只需要把数据不断地往文件流中写进去就可以了。

在Node.js中，流也是一个对象，我们只需要响应流的事件就可以了：data事件表示流的数据已经可以读取了，end事件表示这个流已经到末尾了，没有数据可以读取了，error事件表示出错了。

以流的形式读取文件。（读取二进制文件时要，chunk.toString()一下）

```
var rs = fs.createReadStream('fs.txt', 'utf-8');

rs.on('data', function(chunk) {
    console.log("DATA:");
    console.log(chunk);
});
rs.on('end', function() {
    console.log('END');
});
rs.on('error', function() {
    console.log(error);
});
```

以流的形式写入文件

```
var ws = fs.createWriteStream('fs.txt', 'utf-8');
ws.write('使用stream写入文件...\n',);
ws.write('END.');
ws.end();
```

使用Stream写入二进制数据

```
var ws2 = fs.createWriteStream('fs.txt');
ws2.write(new Buffer('使用Stream写入二进制数据...\n','utf-8'));
ws2.write(new Buffer('END', 'utf-8'));
ws2.end();
```

pipe

一个Readable流和一个Writable流串起来后，所有的数据自动从Readable流进入Writable流，这种操作叫pipe，在Node.js中，Readable流有一个pipe()方法，就是用来干这件事的。

让我们用pipe()把一个文件流和另一个文件流串起来，这样源文件的所有数据就自动写入到目标文件里了，所以，这实际上是一个复制文件的程序。

```
var rs = fs.createReadStream('sample.txt');
var ws = fs.createWriteStream('copied.txt');
rs.pipe(ws);
```

复制sample.txt的内容到copied.txt文件中，如果copied.txt文件不存在，就会先创建一个。

默认情况下，当Readable流的数据读取完毕，end事件触发后，将自动关闭Writable流。如果我们不希望自动关闭Writable流，需要传入参数：

```
readable.pipe(writable, { end: false });
```