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。

什么是流？流是一种抽象的数据结构。想象水流，当在水管中流动时，就可以从某个地方（例如自来水厂）源源不断地到达另一个地方（比如你家的洗手池）。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）。

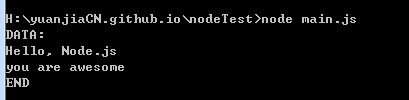
如果应用程序把字符一个一个输出到显示器上，这也可以看成是一个流，这个流也有名字：标准输出流（stdout）。流的特点是数据是有序的，而且必须依次读取，或者依次写入，不能像Array那样随机定位。

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。

下面是一个从文件流读取文本内容的示例：

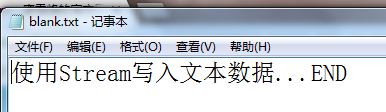
var fs = require("fs");  
var rs = fs.createReadStream("sample.txt","utf-8");  
rs.on("data",function (chunk) {  
 console.log("DATA:");  
 console.log(chunk);  
});  
rs.on("end",function () {  
 console.log("END");  
});  
rs.on("error",function (err) {  
 console.log("error:" + err);  
});

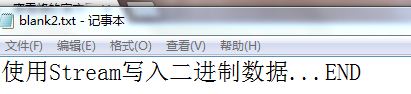


要注意，data事件可能会有多次，每次传递的chunk是流的一部分数据。

要以流的形式写入文件，只需要不断调用write()方法，最后以end()结束：

var fs = require("fs");  
var ws1 = fs.createWriteStream("blank.txt","utf-8");  
ws1.write("使用Stream写入文本数据...\n");  
ws1.write("END");  
ws1.end();  
  
var ws2 = fs.createWriteStream("blank2.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();





所有可以读取数据的流都继承自stream.Readable，所有可以写入的流都继承自stream.Writable。

### pipe

就像可以把两个水管串成一个更长的水管一样，两个流也可以串起来。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 fs = require("fs");  
var rs = fs.createReadStream("text1.txt");  
var ws = fs.createWriteStream("text2.txt");  
rs.pipe(ws);

然后就多了个text2.txt文件（text1文件需要提前准备好，不然会报错），打开以后和text1.txt一模一样。

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 });