流传输（stream）

读取流：fs.createReadStream(‘地址’)

写入流：fs.createWriteStream(‘地址’);

读取流.pipe(写入流);

.on(‘error’,err=>fn) 读取/写入失败

.on(‘data’,data=>fn) 读取到数据时

.on(‘finish’,()=>fn) 写入完成时

压缩(zlib)

res.setHeader('Content-Encoding','gzip');

let rs = fs.createReadStream(`www${pathname}`);

let gzip = zlib.createGzip();

rs.pipe(gzip).pipe(res);

需要备注头“Content-Encoding” 为gzip，否则浏览器不会打开网页，而是会下载该页面

对于图片类文件，压缩效率非常低下，而文档类则极其高效

读取流、写入流、读写流（压缩、加密）

NodeJS中的模块是运作在单例模式下的，不同环境下引入相同的模块、相当于同一个对象

例如：在A中引入C

在B中引入C，并导出C

在A中引入B导出的C

比较A直接引入的C和A引入B导出的C，发现判断两者相等

当nodeJS发现引入模块的地址是一个文件夹时，会自动寻找该文件夹中的index文件

原生Node：

1.数据——GET、POST、文件 √

2.压缩——zlib

3.流——stream

4.接口管理——router

--------------------------------------------------------------------------------

stream——读一块、发一块：

1.读取流

2.写入流

3.读写流——压缩、加密

--------------------------------------------------------------------------------

zlib

--------------------------------------------------------------------------------

流——任何连续的数据都叫流：文件流、网络流、音频流...

1.好处：减少内存占用、增强系统性能(利用率高)

2.分3种：读取、写入、读写

--------------------------------------------------------------------------------

读取流：req、fs.createReadStream

写入流：res、fs.createWriteStream

读写流：zlib.createGzip

1.压缩文件

2.压缩网络数据——Content-Encoding

--------------------------------------------------------------------------------

setHeader 设置response头——随便折腾

writeHeader 设置并发送头

--------------------------------------------------------------------------------

路由(router)：找路

--------------------------------------------------------------------------------

EventEmitter——事件队列

--------------------------------------------------------------------------------

单例——

--------------------------------------------------------------------------------

1.server处理大量接口

2.events自定义事件

3.模块是单例的

--------------------------------------------------------------------------------

原生Node服务器：

1.http服务

2.文件

3.流传输

4.压缩传输

5.router

--------------------------------------------------------------------------------

Child Processes 多进程

Cluster

Process

--------------------------------------------------------------------------------

Crypto 加密

--------------------------------------------------------------------------------