**1，fs模块**

**（1）创建目录**

**mkdir(path,callback)**

**mkdirSync(path)**

**（2）读取目录**

**readdir(path,callback)**

**readdirSync(path**)

**（3）移除目录,只能移除空目录**

**rmdir(path,callback)**

**rmdirSync(path)**

**（4）创建文件/写入文件**

**writeFile(path,data,callback)**

**writeFileSync(path,data)**

**path 文件路径**

**data 要写入的数据**

**如果文件不存在，会先创建然后再写入，**

**如果文件存在，会覆盖原文件内容写入**

**（5）追加写入/创建文件**

**appendFile(path,data,callback)**

**appendFileSync(path,data)**

**如果文件不存在，会先创建然后再写入数据。**

**如果文件已经存在，会追加（在文件末尾）写入数据。**

**（6）读取文件**

**readFile(path,callback)**

**readFileSync(path)**

**callback 回调函数**

**参数**

**err 可能产生的错误**

**data 读取到的数据，格式为buffer，需要转字符串**

**（7）删除文件**

**unlink(path,callback)**

**unlinkSync(path)**

**（8）检测文件是否存在**

**existsSync(path)**

**如果存在返回true 不存在返回false**

**（9）文件流**

**流是为了解决大文件的操作，如果一次性读取占用的内存空间很大；如果使用流只需要很小空间就可以读取大文件，形式是分段的，每次读取一段。**

**createReadStream(path) 创建可读取的文件流**

**createWriteStream(path) 创建可写入的文件流(相当于空文件，等待写入)**

|  |
| --- |
| const fs=require('fs');  let rs=fs.createReadStream('a.zip');  //获取读取的数据  //事件：一旦数据流入到内存，自动触发回调函数；  //on用于添加事件，data是数据流入的事件，固定的写法  let count=0;  rs.on('data',(chunk)=>{  //chunk 分块，一段一段的数据  //每读取1端加1  count++;  });  //事件：当读取结束自动触发回调函数  //end结束的事件，固定写法  rs.on('end',()=>{  console.log(count);  }); |
| //通过读取流和写入流完成大文件的拷贝  //创建可读取的流  let rs=fs.createReadStream('a.zip');  //创建可写入的流(相当于创建一个空文件)  let ws=fs.createWriteStream('b.zip');  //把读取到的流通过管道添加到写入流  //管道 pipe  rs.pipe(ws); |

**2，HTTP协议**

**是浏览器和web服务器之间的通信协议**

**（1）通用头信息 general**

**包含了部分响应和请求信息**

**Request URL: 请求的URL,表示浏览器要请求的资源**

**Request Method: 请求的方法，常见的有get / post**

**get：获取 post：给与**

**Status Code: 响应的状态码**

1\*\*：正在响应，还没有结束

2\*\*：成功的响应

3\*\*：响应的重定向，会跳转到另一个URL

4\*\*：客户端错误

5\*\*：服务器端错误

**（2）响应的头信息 response**

**web服务器作出的叫做响应**

Content-Type：响应的文件类型

Content-Length：响应的文件长度，单位是字节

Location：跳转的URL，往往结合着状态码302使用

**（3）请求的头信息 request**

**浏览器发出的叫做请求**

**（4）请求主体**

**传递的数据，例如注册，登录传递数据**

**可有可无，没有传递数据，这项不显示。**

**3，HTTP模块**

**创建web服务器**

|  |
| --- |
| **const http=require('http');**  **const app=http.createServer(); //创建web服务器**  **app.listen(8800); //设置端口号**  **//通过事件来接收浏览器的请求，一旦请求自动执行回调函数，request固定写法**  **app.on('request', (req,res)=>{**  **req 请求的对象**  **req.url 请求的URL 所对应的是端口号后面的部分**  **req.method 请求的方法**  **res 响应的对象**  **res.writeHead(状态码, 头信息对象) 设置状态码和头信息，头信息可以为空**  **res.write() 设置响应的内容**  **res.end() 结束并发送响应**  **});** |

课后任务

(1)复习今天内容，整理思维导图

(2)练习

创建web服务器，设置端口为8080，根据浏览器请求的URL作出响应

/index 响应 '这是首页'

/login 响应 this is login page

/list 响应文件 list.html （同步读取文件，然后把读取到的响应过去）

/study 跳转 'http://www.tmooc.cn'

其它 响应 404 Not found

(3)预习express第三方模块