1、Buffer

`Buffer` 类被引入作为 Node.js API 的一部分，使其可以在 TCP 流或文件系统操作等场景中处理二进制数据流。

```
//创建一个buffer
const buf1 = Buffer.alloc(10, 1);
console.log(buf1);

//创建一个buffer包括[1,2,3]
const buf2 = Buffer.from([1,2,3]);
console.log(buf2);
const buf3 = Buffer.from('Buffer创建方法');
console.log(buf3);

// Buffer字符编码
const buf4 = Buffer.from('hello world');
console.log(buf4.toString('base64'));
```

2、设置响应—非常重要　setHeader　writeHead　二选一

常用类型：

'Content-Type', 'text/plain' 字符文本形式响应

'Content-Type', 'text/html' html页面形式响应

'Content-Type', 'application/json' JSON数据形式响应

---------> JSON.stringify(响应的数据); 很关键当以JSON形式响应时一定要对响应的数据进行转JSON格式

如：　const user = qs.parse(postdata); //解析完就是一个字符串类型 的JSON数据

//设置响应类型

```
        res.setHeader('Content-Type', 'application/json');
        //给前台响应json数据
        res.write(JSON.stringify(user));
        res.end();
```

```
 //设置响应数据的方式
 res.setHeader('Content-Type', 'text/plain');
//设置响应码及响应数据的方式
 res.writeHead(200,{'Content-Type':'text/plain'});
```

```
postdata = Buffer.concat(postdata).toString();
console.log(postdata);    //user=125&users=258
const user = qs.parse(postdata);    //解析完就是一个字符串类型 的JSON数据
//  { user: '125', users: '258' }
console.log('解析完了的数据',user);
//设置响应头为json
res.setHeader('Content-Type', 'application/json');
//  res.setHeader('Content-Type', 'text/plain')
//  res.write(user);
//  res.end();
//给前台返回json数据
res.write(JSON.stringify(user));
res.end();
```

```
fs.readFile('./liu/3.html',function (err,data) {
    if (err){
        throw err;
    }
    //设置响应的头    响应码、响应的类型    响应的类型很重要直接决定了最终看到的结果
      res.writeHead(200,{'Content-Type':'text/plain'});
    //设置响应类型
    res.setHeader('Content-Type', 'text/html');
    res.write(data);
    res.end();
```