自己实现AJAX

1. JS操作请求与响应

```
http报文格式
请求格式
GET /xxx HTTP/1.1                                    第一部分
HOST: jack.com:8002                                  第二部分        key:value形式
Content-Type: application/x-www-url-encoded
                                                     空行是第三部分，下面就是第四部分请求体
响应格式
HTTP/1.1 200 OK                                      第一部分
Content-Type: text/html                              第二部分        key:value形式
                                                     第三部分        空行
<!DOCTYPE html>                                      第四部分        响应体
<html>.…</html>
```

1.1 JS 可以设置任意请求 header 吗? 可以

第一部分 request.open('get', '/xxx')

第二部分 request.setHeader('content-type', 'x-www-form-urlencoded')

第四部分 request.send('a=1&b=2')

JS更改http请求的对应关系

前端文件

```
myButton.addEventListener('click', (e)=>{
    let request = new XMLHttpRequest()
    request.open('post', '/xxx') // 配置 request
    request.setRequestHeader('frank', '18'); 2
    request.send('试试设置第四部分') 4

    request.onreadystatechange = ()=>{
        if(request.readyState === 4) {
            console.log('请求响应都完毕了')
            if(request.status >= 200 && request.status < 300) {
                console.log('说明请求成功')
                console.log(typeof request.responseText)
                console.log(request.responseText)
                let string = request.responseText
                // 把符合 JSON 语法的字符串
                // 转换成 JS 对应的值
                let object = window.JSON.parse(string)
                // JSON.parse 是浏览器提供的
                // document.getElementById 是浏览器提供的
                console.log(typeof object)
                console.log(object)
                console.log('object.note')
                console.log(object.note)
            }else if(request.status >= 400) {
                console.log('说明请求失败')
            }
        }
    }
})
```

http报文

```
Request Headers      view parsed
1 POST /xxx HTTP/1.1
  Host: localhost:8001
  Connection: keep-alive
  Content-Length: 24
  Origin: http://localhost:8001
2 frank: 18
  User-Agent: Mozilla/5.0 (Windows NT 10.0
  0.3325.181 Safari/537.36
  Content-Type: text/plain;charset=UTF-8
  Accept: */*
  Referer: http://localhost:8001/
  Accept-Encoding: gzip, deflate, br
  Accept-Language: zh-CN,zh;q=0.9
  Cookie: Webstorm-ce7656cd=62c90d1d-a20a-

General
  Request URL: http://localhost:8001/xxx
  Request Method: POST
  Status Code: 200 OK
  Remote Address: [::1]:8001
  Referrer Policy: no-referrer-when-downgrade
Response Headers      view source
  Access-Control-Allow-Origin: http://frank.com:8001
  Connection: keep-alive
  Content-Type: text/json;charset=utf-8
  Date: Tue, 10 Apr 2018 01:29:47 GMT
  Transfer-Encoding: chunked
Request Headers (12)
Request Payload
  试试设置第四部分 4
```

1 js更改http请求第一部分
method与路径

2 js更改http请求第二部分

4 js更改http请求第四部分

1.2 JS 可以获取任意响应 header 吗？ 可以

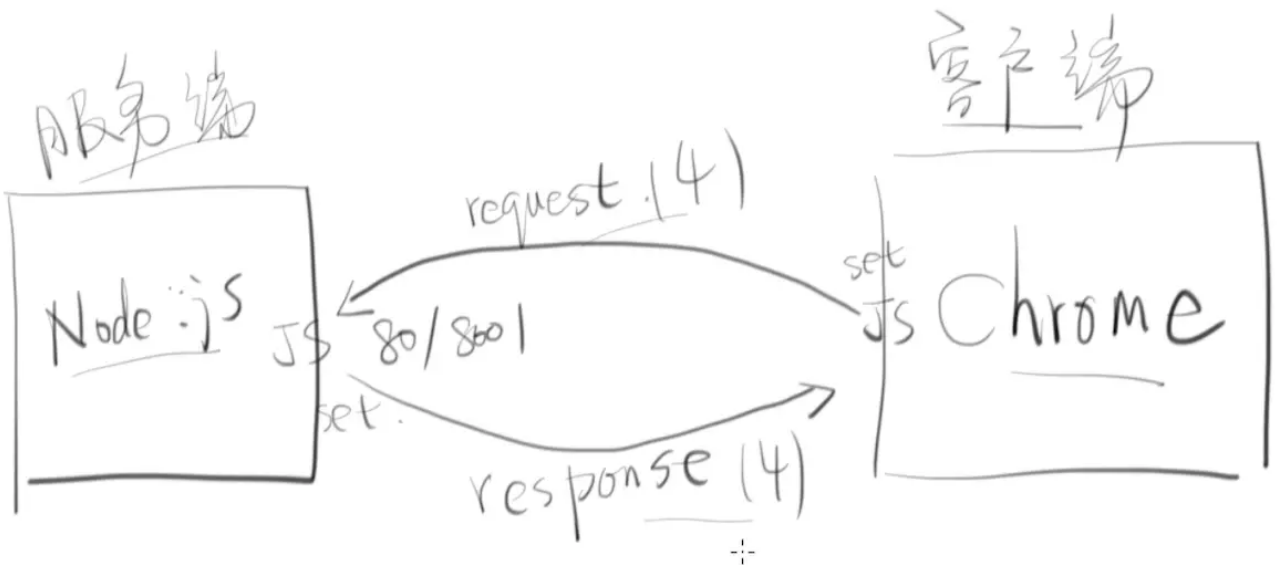
第一部分 request.status / request.statusText

第二部分 request.getResponseHeader() / request.getAllResponseHeaders()

第四部分 request.responseText

所以通过ajax，我们可以获取请求中的4各部分的所有内容（不安全的会不让设置），也可以获取响应中的4个部分。

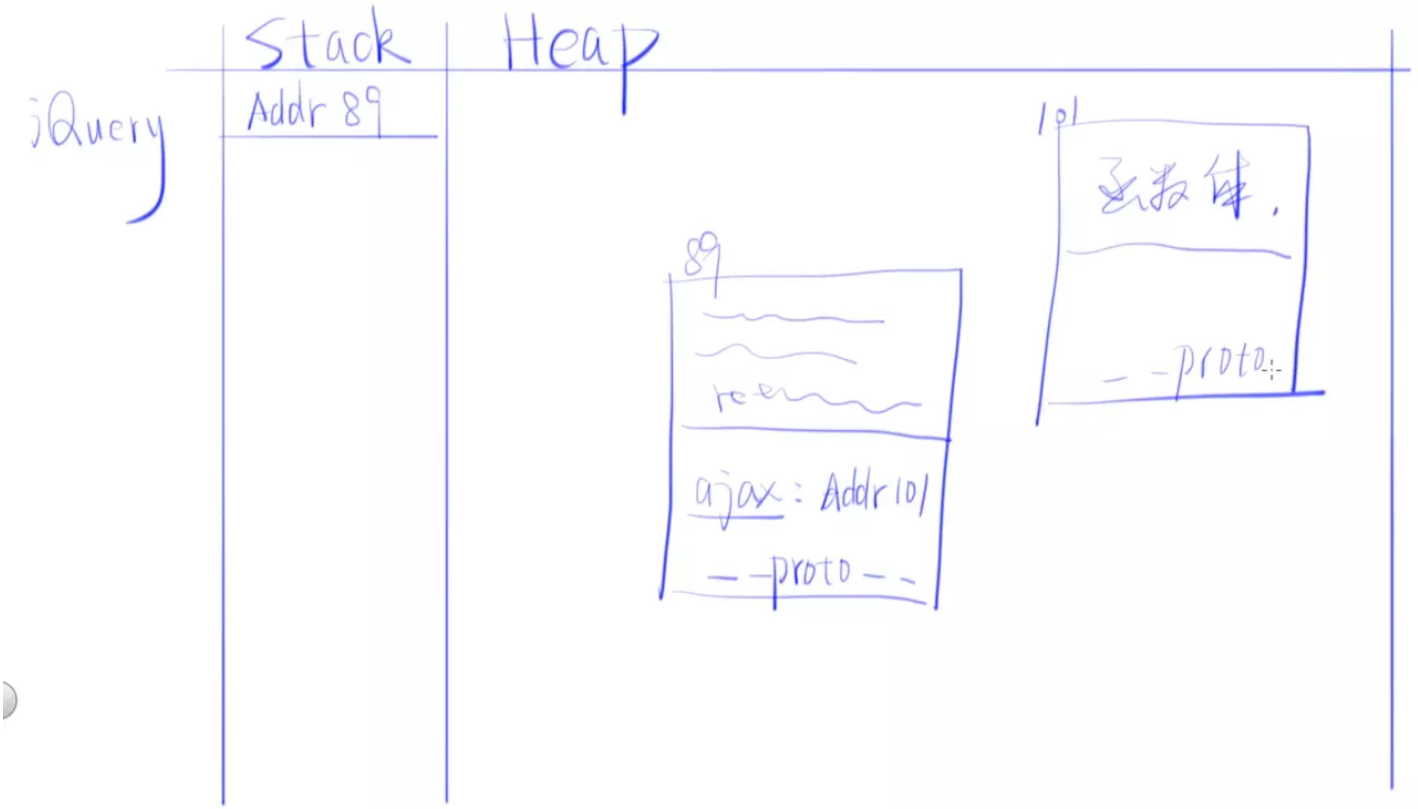
1.3 宏观上看一眼



2. 写window.jQuery.ajax （封装）

2.1 从内存图的角度看我们在做什么

jQuery作为对象，我们假设它在栈内存中地址是Addr89，对应的堆内存中的一个对象89。 现在我们需要给89添加一个新的key，也就是ajax，它的值对应着101这个函数体，我们要写的就是这个函数体。



代码形式如下：

```
window.jQuery.ajax = function(options){
  // 代码    （这就是我们要写的部分）
}

```

2.2 自己封装一个jQuery.ajax 的api

满足条件 jQuery.ajax(url,method,body,success, fail)

代码如下：

```
window.jQuery.ajax = function(url, method, body, success, fail) {
    let request = new XMLHttpRequest()
    request.open(method, url)
    request.onreadystatechange = ()=>{
        if(request.readyState === 4) {
            if(request.status >= 200 && request.status < 300) {
                success.call(undefined, request.responseText)
            } else if (request.status >= 400) {
                fail.call(undefined, request)
            }
        }
```

```
    }
    request.send(body)
}
```

3. 升级 jQuery.ajax 满足 Promise 规则

promise的好处
1.完全不需要记是传 success还是成功或是error还是fail，只需要then(这里放成功，这里放失败)，标准化操作

1. 可以对同一个状态进行多次处理

3.1 升级 jQuery.ajax 满足 Promise 规则

```
window.jQuery.ajax = function({url, method}) { // 传参是 es6解构赋值
        return new Promise ( function(resolve, reject){ // 这一行很关键
                let request = new XMLHttpRequest()
                request.open(method, url)
                request.onreadystatechange = ()=>{
                        if(request.readyState === 4) {
                                if(request.status >= 200 && request.status < 300) {
                                        resolve.call(undefined, request.responseText)
                                } else if (request.status >= 400) {
                                        reject.call(undefined, request)
                                }
                        }
                }
                request.send()
        })
}
```

https://github.com/FrankFang/nodejs-test-cors/blob/6b7c53c0eccb712213b7ed9640e8ca7d21620166/main.js

https://github.com/FrankFang/nodejs-test-cors/blob/e86d62069b517d35374eecd6b8a3d4b81402d48a/main.js