# Ajax

第1天课堂笔记（本课程共4天）

目录

[Ajax 1](#_Toc477288584)

[一、Ajax整体感知 2](#_Toc477288585)

[1.1 从HTTP说开去 2](#_Toc477288586)

[1.2 Ajax技术欣赏 3](#_Toc477288587)

[1.3 Ajax简介 5](#_Toc477288588)

[二、Hello World 8](#_Toc477288589)

[2.1 快速演示 8](#_Toc477288590)

[2.2 XHR对象 8](#_Toc477288591)

[2.3 open方法 9](#_Toc477288592)

[2.4 到底啥是异步？ 9](#_Toc477288593)

[2.5 send()方法 11](#_Toc477288594)

[2.6 encodeURIComponent() 11](#_Toc477288595)

[2.7 post请求必须setRequestHeader一下 11](#_Toc477288596)

[2.8 readystatechange事件 12](#_Toc477288597)

[2.9 HTTP状态码 12](#_Toc477288598)

[2.10 IE6兼容 14](#_Toc477288599)

[2.11 缓存问题 14](#_Toc477288600)

[三、通用函数的封装 15](#_Toc477288601)

[四、表单序列化 16](#_Toc477288602)

[五、JSON处理 17](#_Toc477288603)

[5.1 字符串转为JSON 17](#_Toc477288604)

# 一、Ajax整体感知

## 1.1 从HTTP说开去

上网就是请求文件，就是你输入网址之后，实际上有真实物理文件从服务器上传输到你的计算机中了。

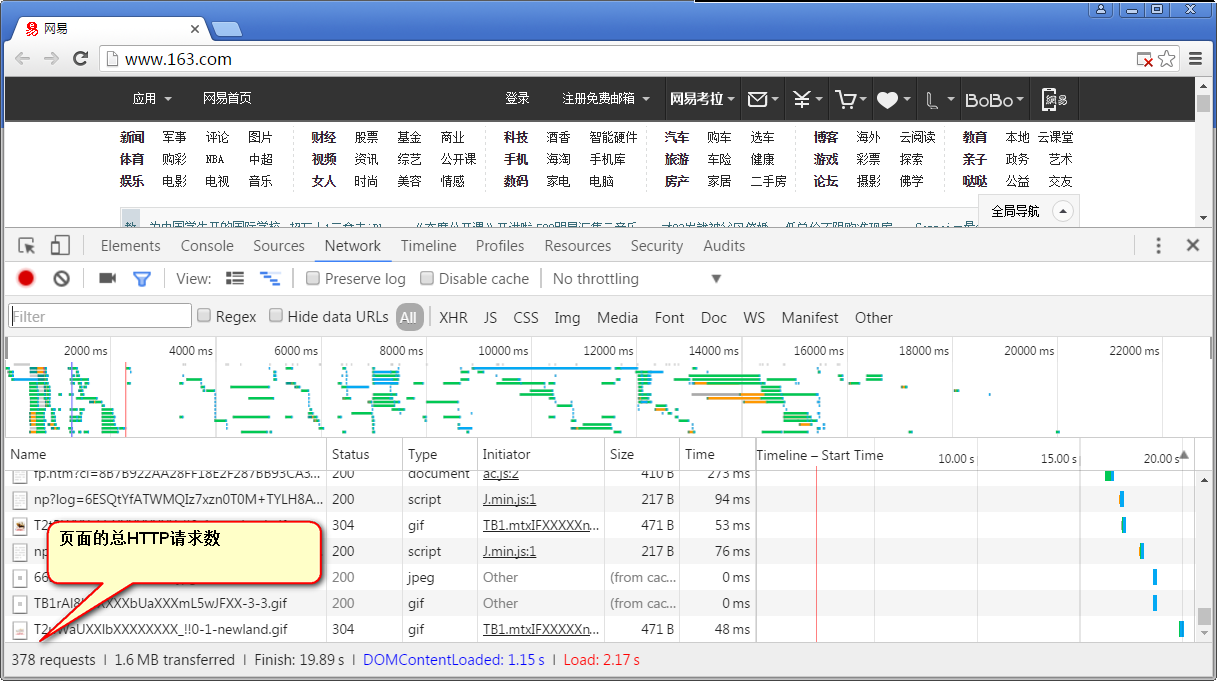
|  |
| --- |
| 所以：   1. 上完网之后，360每次清理垃圾，能清理好多。这是因为上网的时候所有的图片、html文件、css文件、js文件、swf文件都保存到本地了。 2. 当我们第一次打开网页的时候很慢，但是第二次刷新，很快。这是因为第一次已经把所需要的图片保存下来了，所以第二次就不传输了。 |

超文本传输协议（HTTP，HyperText Transfer Protocal ），是简单、可靠的互联网文件传输协议。

HTTP的传输，不是一个文件整体传输，而是把一个大的文件，分成一个个报文(message)，然后传输的。所以我们的浏览器渲染页面，一块一块加载的，图片是一条一条显示的。的确，一个jpg图片，也是通过多个报文回来的。每个报文都会进行校验，保证这个报文里面的内容是准确的、和服务器一致的。所以HTTP是可靠的，文件不会发生任何偏差。

HTTP是无连接的，就是你访问一个网站，此时可能产生多个HTTP请求，请求html页面、请求jpg图片、请求外链css样式表，这些请求走过的路线，可以不一致。也就是说，没有和服务器有一个持久通路，你的每次请求HTTP完成之后，请求就关闭了，一个页面要多次打开、关闭HTTP。

可以通过Chrome浏览器的Network面板查看HTTP请求，和总请求数：



文件上传协议（FTP，file Transfer Protocal）：用于本地往服务器上上传文件的。这个协议也是可靠的，这个协议是持续连接的。

“上网”这个事儿的本质，就是你输入网址之后，浏览器发出HTTP请求，请求服务器上的文件。服务器上的文件，再通过HTTP传输到本地，在浏览器中进行渲染。

HTTP是一对儿一对儿的，先请求Request，然后响应Response。每个请求都是以报文的形式，报文分为两部分：报文头、报文体。

服务器现在需要得到用户的数据，所以此时有两种方法向服务器传输数据，说白了就是通过HTTP向服务器传数据：

GET请求：实际上就是把信息写在URL里面

|  |
| --- |
| http://localhost:3000/posts?id=1 |

POST请求：就是把信息放到HTTP请求的报文体里面，不在URL中。

发出HTTP请求的方法，我们目前知道：

1. 输入网址
2. 点击链接，本质上就是重新输入了网址
3. 提交表单

一次HTTP请求，有上行request、下行response两部分。通常，浏览器产生HTTP请求，是由于用户输入了新的网址、或者点击了超级链接，使页面跳转，这将导致页面的全局刷新。而Ajax技术，可以使网页悄悄地、偷偷地发起HTTP请求，请求回来的数据在页面局部刷新呈递。

## 1.2 Ajax技术欣赏

三个要素：

① 带着数据偷偷上到服务器（GET或者POST，GET是通过URL地址?，POST报文头）

② 传回JSON。

③ 组建DOM、更新页面

[ajax w3c?](https://www.w3schools.com/xml/ajax_intro.asp) [ajax ibm?](https://www.ibm.com/developerworks/cn/xml/wa-ajaxintro1.html) [Ajax mdn](https://developer.mozilla.org/zh-CN/docs/AJAX/Getting_Started) [ajax Wikipedia](https://en.wikipedia.org/wiki/Ajax_(programming)) [ajax tutorial](https://www.xul.fr/en-xml-ajax.html)

[jd注册用户](https://reg.jd.com/reg/person?ReturnUrl=https%3A//www.jd.com/) [百度搜索](https://www.baidu.com/) [豆瓣电影](https://movie.douban.com/) [淘宝首页](https://www.taobao.com/) [携程酒店](http://hotels.ctrip.com/hotel/beijing1" \l "ctm_ref=hod_hp_sb_lst)

比如偷偷请求一个网址：

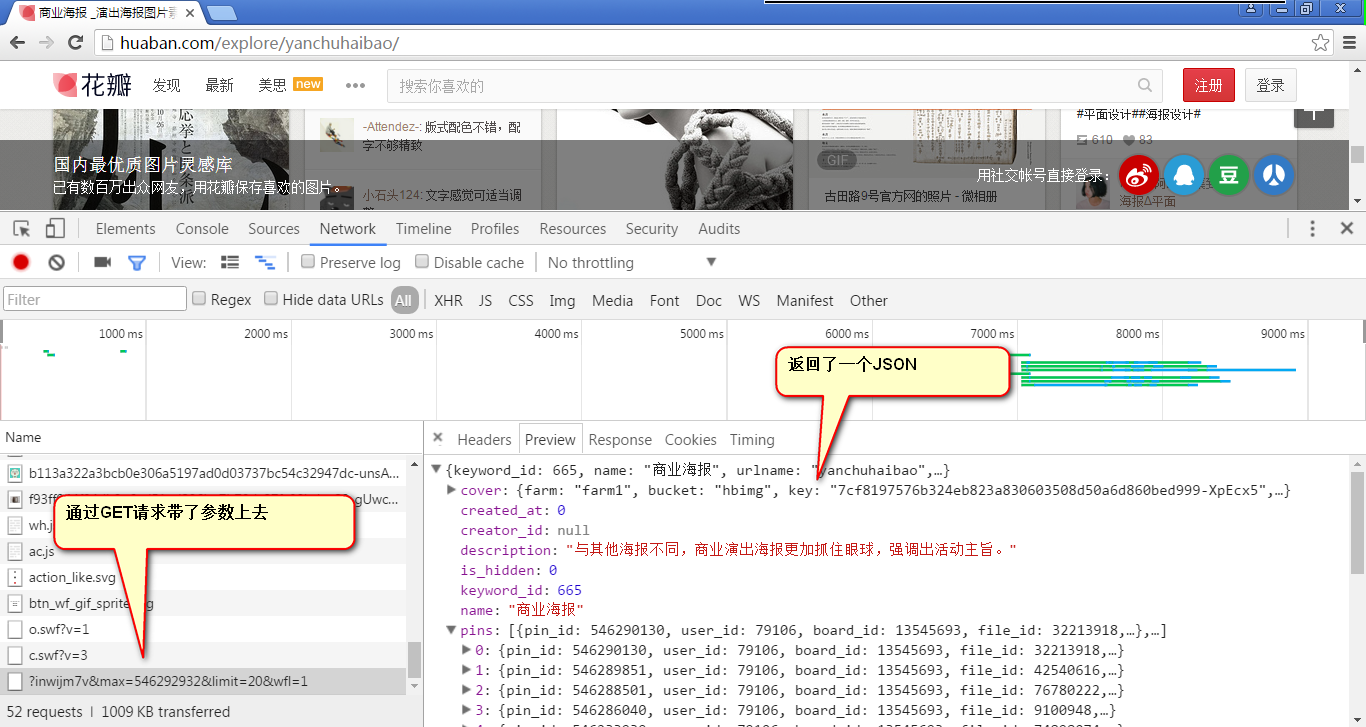
|  |
| --- |
| 1. www.huaban.com/page/r.php?page=3 |

你就会发现，相当于带了page:3这个参数上到了服务器。

服务器就给了你第三页的瀑布流内容：

|  |
| --- |
| 1. { 2. "content" : [ 3. { 4. "biaoti":"", 5. "neirong":"", 6. "images":"" 7. }, 8. { 9. "biaoti":"", 10. "neirong":"", 11. "images":"" 12. }, 13. { 14. "biaoti":"", 15. "neirong":"", 16. "images":"" 17. } 18. ] 19. } |

然后前端开发工程师利用这个JSON创建更多的DOM，改变页面的内容。



## 1.3 Ajax简介

Asynchronous JavaScript and XML （异步JavaScript和XML）



实际上现在工作没有一个公司使用XML当做后台、前台的中介文件，都是使用JSON。所以Ajax现在应该改名为Ajaj（Asynchronous JavaScript and JSON），但是大家还是约定俗成起名为Ajax。

[Xml w3c](https://www.w3schools.com/xml/xml_whatis.asp) [xml wikipedia](https://en.wikipedia.org/wiki/XML)

下面的这个就是XML，和HTML很像，唯一的不同就是标签可以自定义，也是表达语义的，但是不用被浏览器呈递，就是负责交换信息的，现在已经被JSON替代。因为XML后台难以生成，前台难以解析。

|  |
| --- |
| 1. <info> 2. <neirong1> 3. <biaoti></biaoti> 4. <nr></nr> 5. <riqi></riqi> 6. </neirong1> 8. <neirong2> 9. <biaoti></biaoti> 10. <nr></nr> 11. <riqi></riqi> 12. </neirong2> 13. <neirong3> 14. <biaoti></biaoti> 15. <nr></nr> 16. <riqi></riqi> 17. </neirong3> 18. </info> |

XML没有数组，并且长，要有标签的封闭。

JSON非常棒： [JSON规格？](http://www.ietf.org/rfc/rfc4627.txt) [JSON？](http://www.json.org/)

|  |
| --- |
| 1. { 2. "content" : [ 3. { 4. "biaoti":"", 5. "neirong":"", 6. "images":"" 7. }, 8. { 9. "biaoti":"", 10. "neirong":"", 11. "images":"" 12. }, 13. { 14. "biaoti":"", 15. "neirong":"", 16. "images":"" 17. } 18. ] 19. } |

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页页面。

我们说说历史：

Google在2004年的Gmail和Google Map都大量使用了Ajax技术，一下子就让人们认识到Ajax对提高用户体验的重要性。

Jesse James Garrett Ajax布道者



有趣的信息：

● AJAX 不是新的编程语言，而是一种原来被人放到角落里，突然被这个哥们推广，它是使用已有标准的新概念。  
● 2005年由美国人Jesse James Garrett推广，并取名。神奇的是，这哥们并不是搞程序的，而是搞设计的，是交互设计大师、用户体验大师。甚至是个优秀建筑设计师。  
● 在 2005 年，Google 通过其 Google Suggest 使 AJAX 变得流行起来。  
● 在今天，很少有哪个网站不使用Ajax技术。  
● Ajax技术对智能手机支持非常好。

# 二、Hello World

## 2.1 快速演示

Ajax必须运行在服务器端，不能随便一个页面往浏览器里面拖。

|  |
| --- |
| 1. //Ajax的一个固定的模板： 2. //第1步创建一个xhr对象，使用new关键来调用一个内置构造函数 3. var xhr = new XMLHttpRequest(); 4. //第2步指定接收回来的内容，怎么处理。监听xhr对象的onreadystatechange事件，这个事件在xhr对象的“就绪状态”改变的时候触发。我们只关心就绪状态为4的时候的事情。 5. xhr.onreadystatechange = function(){ 6. if(xhr.readyState == 4){ 7. //接收完文件要做的事情，让h1的内容变为读取的东西 8. biaoti.innerHTML = xhr.responseText; 9. } 10. } 11. //第3步创建一个请求，第一个参数是请求的类型get或者post，第二个参数就是请求的路径，第三个参数叫做是否使用异步机制 12. xhr.open("get","a.txt",true); 13. //第4步发送请求，圆括号里面是请求头内容，get请求没有报文头写null 14. xhr.send(null); |

## 2.2 XHR对象

没啥好讲的，Ajax完全依赖XMLHttpRequest对象，字面意思就是“XML文件的HTTP请求”对象。大家一般把new出来的变量叫做xhr。

|  |
| --- |
| 1. var xhr = new XMLHttpRequest(); |

驼峰命名法

XML : 就是一个文件格式，

HTTP： 传输协议

Request :请求

所以这个对象顾名思义，它就是发出一个HTTP请求的对象，请求的是XML文件。

实例的名字，我们取字头，叫做xhr。注意，面试会考笔试，会让你写这个对象，记住“小黄人” xhr。

IE6不兼容，请见2.10 IE6兼容；

## 2.3 open方法

open方法表示让xhr对象配置一个请求，open字面意思是打开，就是打开一个请求。open之后并没有真正的发送请求，而是要用send()方法

open方法有三个参数：

|  |
| --- |
| 1. xhr.open(要发送的请求类型,路径,是否使用异步); |

第一个参数，要么是"get" 要么是"post"

第二个参数，就是处理这个请求的PHP、java、.net路径

的三个参数，表示是否是异步处理，没有任何理由不用true，必须写true

## 2.4 到底啥是异步？



现在目前知道的异步：

1） setInterval、setTimeout

2） 所有的事件监听可以看做异步

3） Ajax（实际上也是onreadystatechange事件）

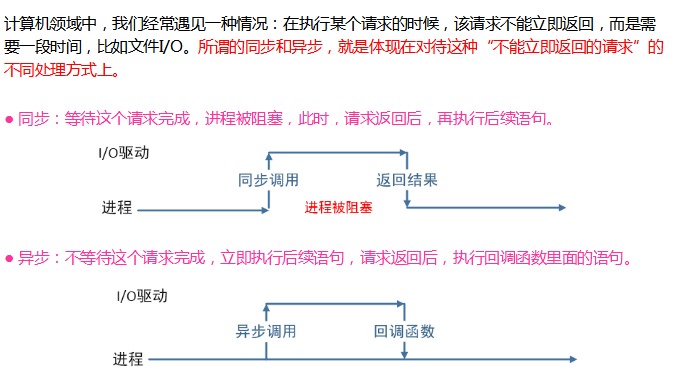
**异步：不死等耗时较长的事情完成，先同时干别的事情，耗时较长的事情完成了，控制权交给回调函数**

**同步：就是异步的反面，死等那个耗时较长的事情完成，然后做别的事情**

说白了，就是死等不死等的区别。

node.js中我们深入探讨单线程、异步的关系。答案是没关系。

多线程可以同步，单线程也可以异步。



多说一句，Node.js世界全是异步！！

PHP读取数据库，是同步的，死等读取完成：

|  |
| --- |
| 1. //php是同步读取数据库，I/O时间非常长，仍然死等，I/O驱动程序开始工作，CPU计算进程被阻塞。 2. $result = Mysql\_query("SELECT \* FROM xuesheng"); 3. mysql\_fetch\_array($result); |

node.js不是这样，是异步的：

|  |
| --- |
| 1. db.read({"student":"all"},function(data){ 2. console.log(data.xuehao); 3. }); 4. console.log(1+2+3); |

Ajax异步的细节：

● 浏览器执行到Ajax代码这行语句，发出了一个HTTP请求，欲请求服务器上的数据a.txt。服务器的此时开始I/O，所谓的I/O就是磁盘读取，需要花一些时间，所以不会立即产生下行HTTP报文。

● 由于Ajax是异步的，所以本地的JavaScript程序不会停止运行，页面不会假死，不会傻等下行HTTP报文的出现。后面的JavaScript语句将继续运行。进程不阻塞。

● 服务器I/O结束，将下行HTTP报文发送到本地。此时，回调函数将执行。回调函数中，将使用DOM更改页面内容。

所以，没有任何理由将open函数的第三个参数设置为false，一定是true的。

## 2.5 send()方法

send方法就是发送请求，里面的参数表示http requset报头里面携带内容。

get请求报头里面没有内容，post请求有内容。

get请求：

|  |
| --- |
| 1. xhr.send(null); |

post请求，写的也是类似于get请求的参数字符串：

|  |
| --- |
| 1. xhr.send("name=kaola&age=18"); |

## 2.6 encodeURIComponent()

请求尽量不要有中文，如果要传输中文为了防止服务器上错乱，我们前端一般要进行encodeURIComponent处理。

据我所知，高级后台程序语言，都能够自动处理转译。

get请求、post请求如果要用中文，一般要将中文转为URI标准字符。

语法：

|  |
| --- |
| 1. encodeURIComponent("我是一个文本"); |



世界上所有的文字都有URI编码，根据你的HTML页面的字符集不同，URI编码也不同。

## 2.7 post请求必须setRequestHeader一下

POST请求在服务器端比较难处理，需要用服务器写对应“流处理”程序。为什么？POST请求参数的尺寸可以无限大，所以post请求也是一段一段上去的。（node.js中我们将遇见这个datachuck）。  
比如$\_POST[] 接收会报错。

但是，如果是表单提交，那么PHP内置了相应处理程序。Ajax如果需要模拟表单提交，那么需要在send前设置：

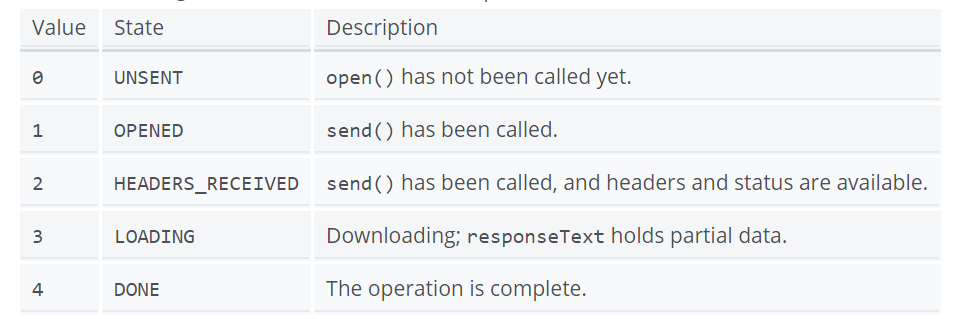
|  |
| --- |
| 1. xhr.open("post","do2.php",true); 2. //如果用post发送请求，那么必须写一句话，模拟成form表单提交： 3. **xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");** 4. xhr.send("xingming="+encodeURIComponent("考拉")+"&age=18"); |



## 2.8 readystatechange事件

xhr对象一旦开始open，就有了readyState属性，readyState属性一旦发生改变，就能够触发onreadystatechange事件。

xhr对象的readyState属性的值有0、1、2、3、4，一共五个值：



0: UNSENT open()还没有被调用

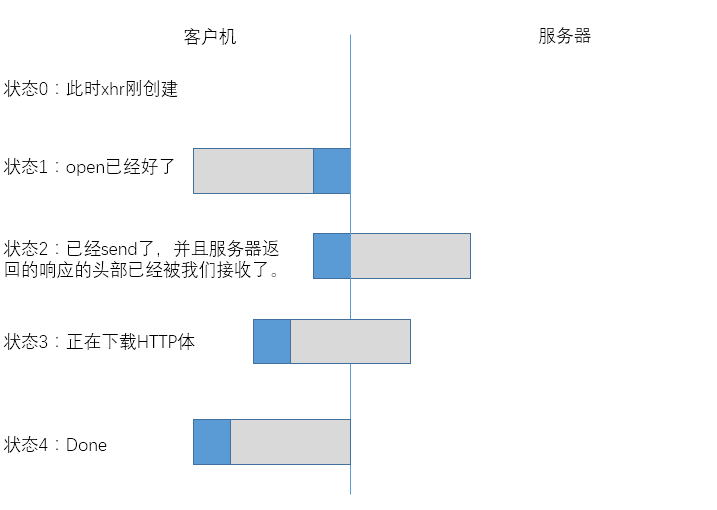
1: OPENED 调用send()

2: HEADERS\_RECEIVED 头部已经被服务器接收

3: LOADING 开始接收服务器的返回的东西，还没有接收完全

4: DONE 完成

工程上，我们只关心4这个状态。



xhr.responseText就表示接收回来的文本。是一个string类型。

## 2.9 HTTP状态码

每一次http请求，会根据请求是否成功，有不同的状态码。

所以应该用if语句验证：

|  |
| --- |
| if(xhr.status >= 200 && xhr.status < 300 || xhr.status == 304){  //继续执行  } |

这样做的好处就是当请求的内容不存在的时候，就不把错误信息显示了。

查看每个HTTP请求的状态码Status

常见状态码：

200 ok，成功

302 文件临时移动

304 not modified ，文件没有改变。浏览器会比对你请求的文件，和已经在缓存文件夹中的文件，如果相同，不再请求。这就是为啥第二次访问网站，速度更快的原因。

400 错误的请求

401 没有权限

403 禁止访问

404 not found ，没有找到文件

500 服务器故障

502 错误的访问

503 服务器不可用

学习node.js之后，你就知道服务器可以指定状态码返回。

大类型要记住：



## 2.10 IE6兼容

高级浏览器是

|  |
| --- |
| 1. new XMLHttpRequest(); |

IE6是：

|  |
| --- |
| 1. new ActiveXObject("Microsoft.XMLHTTP"); |

能力检测：



## 2.11 缓存问题

就是当你用Ajax请求一个文件的时候，我们改变文件内容，再次请求这个文件，文件有严重的缓存，内容不会更改。解决问题的思路就是让每次请求都发往不同的地址，即使是同一个文件，也要有不同的参数，此时浏览器就没有缓存了！

方法1：随机数法：

|  |
| --- |
| xhr.open("GET","test.txt?" **+ Math.random()**,true); |

方法2：时间戳：

时间戳就是Date.parse(new Date()) 到1970年1月1日00:00的毫秒数，精确到秒。

比如：

Date.parse("2016-10-1 10:10:11");

|  |
| --- |
| xhr.open("GET","test.txt?" **+ Date.parse(new Date())**,true); |

最靠谱的方法是服务器的配置。

# 三、通用函数的封装

使用的API

|  |
| --- |
| 1. myajax.get(“check.php”,{“xingming”:”考拉”,”age”:18},function(err,data){ 3. }); |

函数在JS中是一等公民，变量能出现的地方，函数就能出现。

|  |
| --- |
| 1. //唯一向外暴露的顶层变量 2. var myajax = window.myajax = {}; 3. myajax.get = function(){ 4. } 5. myajax.post = function(){ 6. } |

比较巧妙的东西就是queryJSON → querySring

|  |
| --- |
| 1. myajax.\_queryjson2querystring = function(json){ 2. var arr = []; //结果数组 3. for(var k in json){ 4. arr.push(k + "=" + encodeURIComponent(json[k])); 5. } 6. return arr.join("&"); 7. }; |

# 四、表单序列化

表单中的每个控件，都有name属性，值却是千差万别的，radio、checkbox、select甚至需要迭代才能得到他们的值。不方便，并且用ajax提交表单的时候，要手工序列化。

你填的表单，最终要变为：

|  |
| --- |
| 1. name=考拉&age=18&sex=男&hobby=篮球&hooby=足球 |

这个就叫做表单序列化。

经典面试题，写一个表单序列化函数。

DOM提供了一个非常简单的一个属性，所有的表单元素都可以用elements来获得里面的所有控件。

|  |
| --- |
| 1. var elems = biaodan.elements; |

元素的类型，input元素的type属性就是自己的type，而select的type属性是select-one

|  |
| --- |
| 1. elems[0].type |

# 五、JSON处理

对于前端开发工程师来说，主要的工作：

①准确发出请求，之前讲的事情都是发请求

②根据收到的JSON进行处理

## 5.1 字符串转为JSON

接收到的东西，永远是string，无论它长得多么像JSON。

|  |
| --- |
| 1. myajax.get("a.json",{},function(err,data){ 2. if(err) return; 3. console.log(typeof data); 4. }); |



首先的工作就是把字符串转为JSON：

方法1：eval语句

eval语句很神奇，能够识别字符串，并把字符串当做JS语句执行。

先来说eval是什么：

|  |
| --- |
| 1. var a = "alert(1+2+3)"; 2. eval(a); |

一个字符串被eval之后，能够把字符串当做语句来执行。

eval语句一般用于奇淫技巧。

现在回到Ajax，data是一个string，可以用eval转为json，需要注意的是，json先通过圆括号的拼接，变为表达式：

|  |
| --- |
| 1. var json = eval(**"("** + data + **")"**); 2. console.log(json); |

方法2：new Function()

说白了我们之前定义函数的时候

|  |
| --- |
| 1. function fun(){ 2. } |

是函数的字面量，实际上能通过new Function()来定义一个函数。但是纯属抽风，没人这么玩儿。

|  |
| --- |
| 1. var fun = new Function("var sum = 0;for(var i = 1 ; i <= 100 ; i++){sum+=i;}alert(sum);"); 2. fun(); |

弹出1~100的和

new Function()也暗含了一个功能，把字符串→语句

回到Ajax中，data现在是string，既然new Function有把字符串→语句的功能，就能retrun 这个object的字面量，然后别忘了执行。

|  |
| --- |
| 1. var data = new Function("return " + data)(); 2. console.log(data); |

方法3：JSON.parse()内置函数

JSON对象注意，不是一个构造函数，就是一个内置对象。这个对象有两个方法

JSON.parse : 字符串 → JSON

JSON.stringify ： JSON → 字符串

|  |
| --- |
| 1. var obj = JSON.parse(data); 2. alert(obj.result[0].name); |

IE6、7不兼容

“老道”JSON作者帮我们写了一个shim。shim就是桌角垫，计算机世界中就是指本身没有这个功能，利用现有功能模拟一个这个功能。IE8、9、高级浏览器有内置JSON对象，而IE6、7没有，就可以用现有功能去模拟JSON对象。

<https://github.com/douglascrockford/JSON-js>

三种方法要熟练记忆。