Ajax 入门和应用

课堂笔记 1

目录

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。

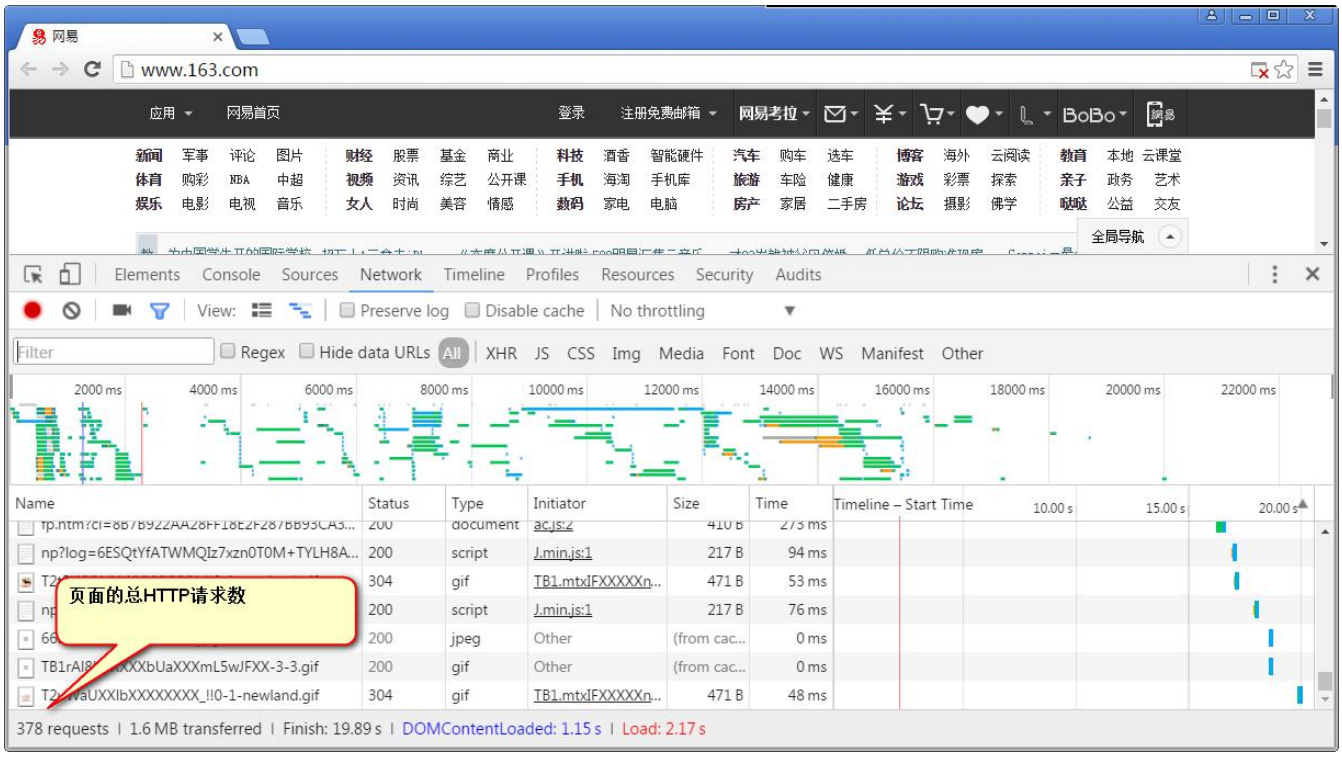
2） 当我们第一次打开网页的时候很慢，但是第二次刷新，很快。这是因为第一次已经把所需要的图片保存下来了，所以第二次就不传输了。

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 报文体）

② 传回 JSON。

③ 组建 DOM、更新页面

ajax w3c? ajax ibm? Ajax mdn ajax Wikipedia ajax tutorial

jd 注册用户 百度搜索 豆瓣电影 淘宝首页 携程酒店

比如偷偷请求一个网址：

```
http://apis.juhe.cn/mobile/get?phone=13429667914&key=ae8ec1963fd90ff8236ac6a355a69def
```

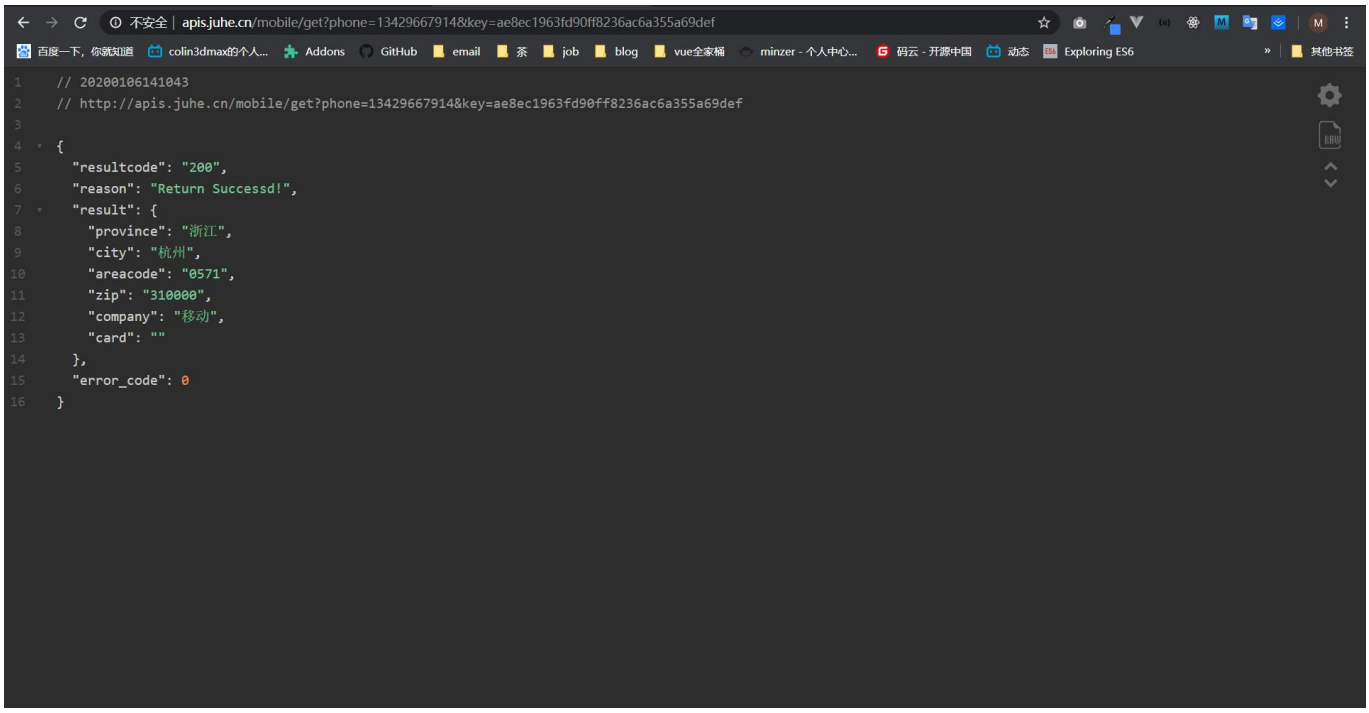
你就会发现，相当于带了 phone:“13429667914”和 key:”ae8ec1963fd90ff8236ac6a355a69def”这两个个参数上到了服务器。

服务器就给了你手机号归属地信息内容：

```
{
  "resultcode": "200",
  "reason": "Return Successd!",
  "result": {
    "province": "浙江",
```

```
      "city": "杭州",
      "areacode": "0571",
      "zip": "310000",
      "company": "移动",
      "card": ""
    },
    "error_code": 0
}
```

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。



前端开发工程师的工作：
- 书写Ajax程序，准确发送带有参数的HTTP请求
- 解析接收到的JSON数据，用DOM技术在页面上呈递

后台工程师的工作：
- 识别HTTP请求中的参数，查询数据库，发回JSON数据

1.3 Ajax 简介

Asynchronous JavaScript and XML （异步 JavaScript 和 XML）

asynchronous [a·syn·chro·nous || æ'sɪŋkrənəs]
adj. 非同期的; 异步的

synchronous ['syn·chro·nous || 'sɪŋkrənɪs]
adj. 同时的; 同步的

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　xml wikipedia

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。

```
<info>
    <neirong1>
        <biaoti></biaoti>
        <nr></nr>
        <riqi></riqi>
    </neirong1>

    <neirong2>
        <biaoti></biaoti>
        <nr></nr>
        <riqi></riqi>
    </neirong2>

    <neirong3>
        <biaoti></biaoti>
        <nr></nr>
        <riqi></riqi>
    </neirong3>
</info>
```

XML 没有数组，并且长，要有标签的封闭。

JSON 非常棒： JSON 规格？ JSON？

```
{
    "content" : [
        {
            "biaoti":"",
            "neirong":"",
            "images":""
        },
        {
            "biaoti":"",
            "neirong":"",
            "images":""
        },
        {
            "biaoti":"",
            "neirong":"",
            "images":""
        }
    ]
}
```

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。

```
//Ajax 的一个固定的模板：
//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
var xhr = new XMLHttpRequest();
//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。
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4){
        //接收完文件要做的事情，让 h1 的内容变为读取的东西
        biaoti.innerHTML = xhr.responseText;
    }
}

//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
xhr.open("get","a.txt",true);
//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
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。

```
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：

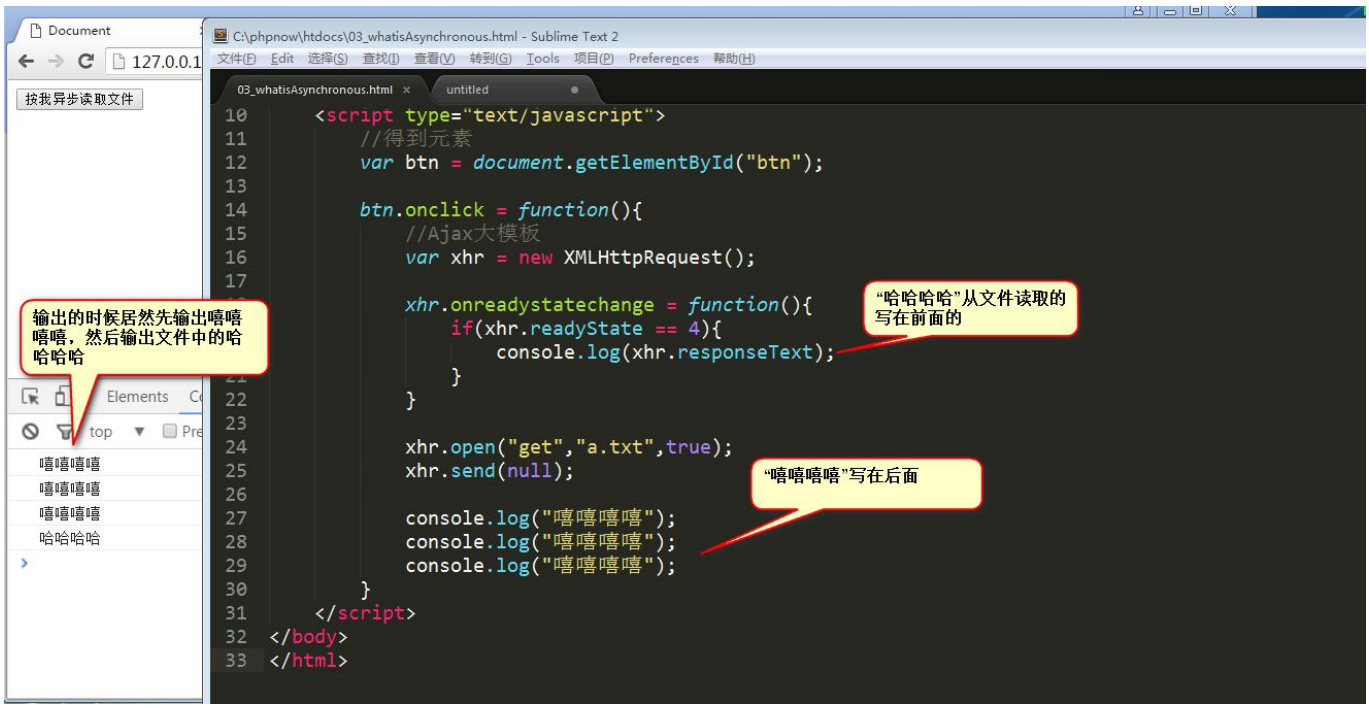
```
1    xhr.open(要发送的请求类型,路径,是否使用异步);
```

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。

那么单线程、异步是什么关系？答案是没关系。

多线程可以同步，单线程也可以异步。

计算机领域中，我们经常遇见一种情况：在执行某个请求的时候，该请求不能立即返回，而是需要一段时间，比如文件I/O。所谓的同步和异步，就是体现在对待这种“不能立即返回的请求”的不同处理方式上。

- 同步：等待这个请求完成，进程被阻塞，此时，请求返回后，再执行后续语句。

- 异步：不等待这个请求完成，立即执行后续语句，请求返回后，执行回调函数里面的语句。

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：

```
//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。
$result = Mysql_query("SELECT * FROM xuesheng");
mysql_fetch_array($result);
```

node.js 不是这样，是异步的：

```
db.read({"student":"all"},function(data){
    console.log(data.xuehao);
});

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：

```
xhr.send(null);
```

post 请求，写的也是类似于 get 请求的参数字符串：

```
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：

```
encodeURIComponent("我是一个文本");
```

```
encodeURIComponent("我是一个文本")
"%E6%88%91%E6%98%AF%E4%B8%80%E4%B8%AA%E6%96%87%E6%9C%AC"
```

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：

```
xhr.open("post","do2.php",true);
//如果用 post 发送请求，那么必须写一句话，模拟成 form 表单提交：
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
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：

Value	State	Description
0	UNSENT	open() has not been called yet.
1	OPENED	send() has been called.
2	HEADERS_RECEIVED	send() has been called, and headers and status are available.
3	LOADING	Downloading; responseText holds partial data.
4	DONE	The operation is complete.

0: UNSENT open()还没有被调用

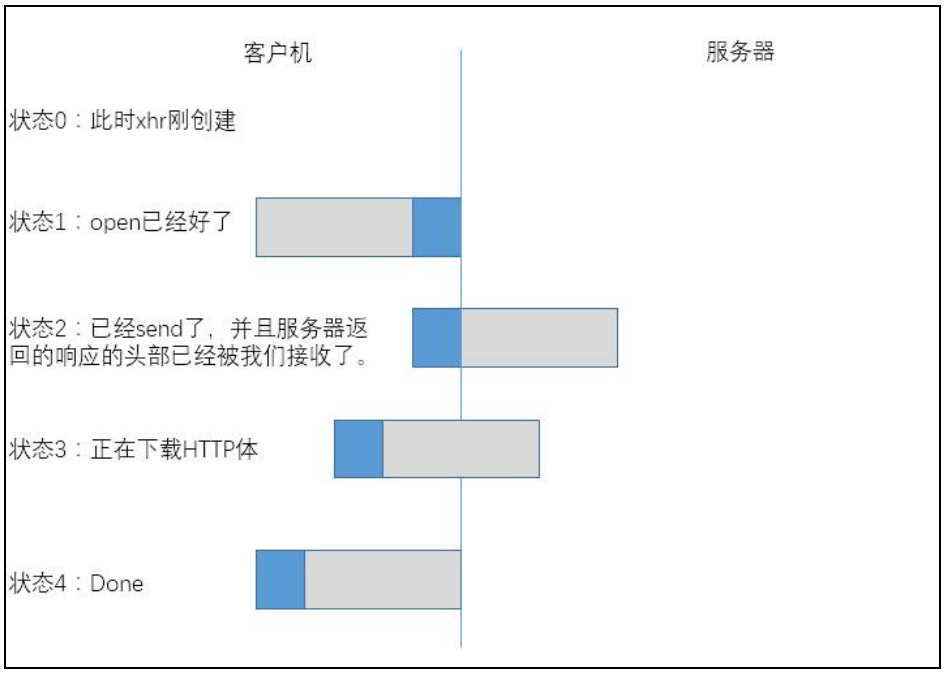
1: OPENED 调用 send()

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

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

```
new XMLHttpRequest();
```

IE6 是：

```
new ActiveXObject("Microsoft.XMLHTTP");
```

能力检测：

```
var httpRequest;
if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
    httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 6 and older
    httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

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

```
myajax.get(“check.php”,{“xingming”:”考拉”,”age”:18},function(err,data){

});
```

函数在 JS 中是一等公民，变量能出现的地方，函数就能出现。

```
//唯一向外暴露的顶层变量
var myajax = window.myajax = {};
myajax.get = function(){
}
myajax.post = function(){
}
```

比较巧妙的东西就是 queryJSON → querySring

```
myajax._queryjson2querystring = function(json){
    var arr = [];  //结果数组
    for(var k in json){
        arr.push(k + "=" + encodeURIComponent(json[k]));
    }
    return arr.join("&");
};
```

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：

```
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。

```
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`

```
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          });
```

```
string
```

首先的工作就是把字符串转为 JSON：

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