logo.png

ajax

H5 web前端

[www.bufanui.com](http://www.bufanui.com)

# 一、HTTP协议

网站是基于HTTP协议的,比如我们在开发网站中经常使用的css,js,图片等等都是基于该协议进行传输的

* 组成部分
  + 从客户端(能够发送HTTP)发出的:请求Request
  + 从服务器返回的:响应Response

**Content-Type:**

application/x-www-form-urlencoded : post请求需要添加

User-Agent：判断浏览器类型，目前主要用于判断手机端还是pc端

Referrer: 当前页面从哪个页面跳转过来的

## 1.1请求/请求报文

请求有客户端发出,主要有三个组成部分:**请求行,请求头,请求主体**



* **请求行:**

请求方法:GET

请求URL

HTTP协议版本

* **请求头**
  + 这里设置的主要是一些信息,包含客户端,服务器

User-Agent：浏览器的具体类型　　如：User-Agent：Mozilla/5.0 (Windows NT 6.1; rv:17.0) Gecko/20100101 Firefox/17.0

链接：<http://www.jb51.net/article/48532.htm>

Accept：浏览器支持哪些数据类型　　如：Accept: text/html,application/xhtml+xml,application/xml;q=0.9;

Accept-Charset：浏览器采用的是哪种编码　　如：Accept-Charset: ISO-8859-1

Accept-Encoding：浏览器支持解码的数据压缩格式　　如：Accept-Encoding: gzip, deflate

Accept-Language：浏览器的语言环境　　如：Accept-Language zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3

Host：请求的主机名，允许多个域名同处一个IP地址，即虚拟主机。Host:www.baidu.com

Connection：表示是否需要持久连接。Keep-Alive/close，HTTP1.1默认是持久连接，它可以利用持久连接的优点，当页面包含多个元素时（例如Applet，图片），显著地减少下载所需要的时间。要实现这一点，Servlet需要在应答中发送一个Content-Length头，最简单的实现方法是：先把内容写入ByteArrayOutputStream，然后在正式写出内容之前计算它的大小。如：Connection: Keep-Alive

Content-Length：表示请求消息正文的长度。对于POST请求来说Content-Length必须出现。

Content-Type：WEB服务器告诉浏览器自己响应的对象的类型和字符集。例如：Content-Type: text/html; charset='gb2312'

Content-Encoding：WEB服务器表明自己使用了什么压缩方法（gzip，deflate）压缩响应中的对象。例如：Content-Encoding：gzip

Content-Language：WEB服务器告诉浏览器自己响应的对象的语言。

Cookie：最常用的请求头，浏览器每次都会将cookie发送到服务器上，允许服务器在客户端存储少量数据。

Referer：包含一个URL，用户从该URL代表的页面出发访问当前请求的页面。服务器能知道你是从哪个页面过来的。Referer: http://www.baidu.com/

* 请求体
  + 这里是提交给服务器的数据
  + 需要注意的是,如果是往服务器提交数据,需要在请求头中设置Content-Type: application/x-www-form-urlencoded(在ajax中需要手动设置)

## 1.2响应/响应报文

响应报文是服务器发回给客户端的.组成部分有状态行,响应头,响应主体



常见相应属性

Cache-Control

响应输出到客户端后，服务端通过该报文头属告诉客户端如何控制响应内容的缓存。

下面，的设置让客户端对响应内容缓存3600秒，也即在3600秒内，如果客户再次访问该资源，直接从客户端的缓存中返回内容给客户，不要再从服务端获取。

Cache-Control: max-age=3600

ETag

一个代表响应服务端资源（如页面）版本的报文头属性，如果某个服务端资源发生变化了，这个ETag就会相应发生变化。它是Cache-Control的有益补充，可以让客户端“更智能”地处理什么时候要从服务端取资源，什么时候可以直接从缓存中返回响应。

ETag: "737060cd8c284d8af7ad3082f209582d"

Location

我们在Asp.net中让页面Redirect到一个某个A页面中，其实是让客户端再发一个请求到A页面，这个需要Redirect到的A页面的URL，其实就是通过响应报文头的Location属性告知客户端的，如下的报文头属性，将使客户端redirect到iteye的首页中：

Location: http://www.google.com.hk

Set-Cookie

服务端可以设置客户端的Cookie，其原理就是通过这个响应报文头属性实现的。

Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1

HTTP响应体：如果请求的是HTML页面，那么返回的就是HTML代码。如果是JS就是JS代码。

HTTP响应头：而设置Cookie，缓存等信息就是在响应头属性设置的。

HTTP响应行：主要是设置响应状态等信息。

## 1.3常见的响应状态

**下面是常见的HTTP状态码：**

* 200 - 请求成功
* 301 - 资源（网页等）被永久转移到其它URL 通过转发出现的状态

转发 无法记录状态 需要通过session传递共享数据

* 404 - 请求的资源（网页等）不存在
* 500 - 内部服务器错误 常见的错误是模板中使用了未定义的变量，后台无法解析

**HTTP状态码分类**

|  |  |
| --- | --- |
| **HTTP状态码分类** | |
| **分类** | **分类描述** |
| 1\*\* | 信息，服务器收到请求，需要请求者继续执行操作 |
| 2\*\* | 成功，操作被成功接收并处理 |
| 3\*\* | 重定向，需要进一步的操作以完成请求 |
| 4\*\* | 客户端错误，请求包含语法错误或无法完成请求 |
| 5\*\* | 服务器错误，服务器在处理请求的过程中发生了错误 |

# 二、Ajax

## 2.1问题

**注册登录案例存在的问题？**

用户名无法验证，如果注册用户名验证失败，需要重新填写整个表单。

**如何解决？**

如果可以“异步检测”用户名合法性，则可以解决问题

ajax可以帮我们解决这样的问题。

## 2.2 什么是ajax

同步(Synchronous)和异步(Asynchronous)

AJAX的全称是Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）。

**ajax**不是新的编程语言，而是一种使用现有标准的新方法。ajax是与服务器交换数据并更新部分网页的技术，在不重新加载整个页面的情况下。

　　ajax是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

ajax是一种用于创建快速动态网页的技术。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。而传统的网页（不使用ajax）如果需要更新内容，必须重载整个网页面。

**ajax的优点：**

　　1、最大的一点是页面无刷新，用户的体验非常好。

　　2、使用异步方式与服务器通信，具有更加迅速的响应能力。。

　　3、可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。并且减轻服务器的负担，ajax的原则是“按需取数据”，可以最大程度的减少冗余请求，和响应对服务器造成的负担。

　　4、基于标准化的并被广泛支持的技术，不需要下载插件或者小程序。

　　5、ajax可使因特网应用程序更小、更快，更友好。

**ajax的缺点：**

　　1、ajax不支持浏览器back按钮。

　　2、安全问题 AJAX暴露了与服务器交互的细节。

　　3、对搜索引擎的支持比较弱。

**使用场景**

在浏览器中,我们也能够不刷新页面,通过ajax的方式去获取一些新的内容,类似网页有微博,朋友圈,邮箱等。

## 2.3 XMLHttpRequest

ajax的核心是XMLHttpRequest对象(简称 XHR)，这是由微软最新引入的一个特性，其他浏览器提供商后来都提供了相同的实现。但是2005年Google在它著名的交互应用程序中使用了异步通讯，如Google、Google地图、Google搜索建议、Gmail等。Ajax这个词由《Ajax: A New Approach to Web Applications》一文所创，该文的迅速流传提高了人们使用该项技术的意识。另外，对Mozilla/Gecko的支持使得该技术走向成熟，变得更为易用。

所有现代浏览器均支持 **XMLHttpRequest** 对象（~~IE5 和 IE6 使用 ActiveXObjec~~t）。XMLHttpRequest 用于在后台与服务器交换数据。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

### 2.3.1 试例：

1. 建立XMLHTTPRequest对象
2. 注册回调函数
   * 当服务器回应我们了,我们想要执行什么逻辑
3. 使用open方法设置和服务器端交互的基本信息
   * 设置提交的网址,数据,post提交的一些额外内容
4. 设置发送的数据，开始和服务器端交互
   * 发送数据
5. 更新界面
   * 在注册的回调函数中,获取返回的数据,更新界面

示例：

|  |
| --- |
| var xhr = new XMLHttpRequest();;  xhr.open('get','/data.json’);  xhr.send(null);  xhr.onreadystatechange = function(){  if (xhr.readyState == 4 && xhr.status == 200) {  console.log(xhr.responseText)  }  } |

|  |
| --- |
| var xhr = new XMLHttpRequest();  xhr.open('post','/test/testPost');  // 如果想要使用post提交数据,必须添加  xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");  ~~xhr.send(JSON.stringify({"name":"李四"}));~~  // key=value&key=value&key=value....  xhr.send('username='+username+'&password='+password);  xhr.onreadystatechange = function(){  if (xhr.readyState == 4 && xhr.status == 200) {  console.log(xhr.responseText)  }  } |
|  |

## ****2.4 XMLHttpRequest\_API****

### ****2.4.1 创建****XMLHttpRequest****对象(兼容性写法)****

* **新版本浏览器:**

var xml=new XMLHttpRequest();

* **（IE5 和 IE6）**

var xml=new ActiveXObject("Microsoft.XMLHTTP");

* **考虑兼容性创建Ajax对象**

|  |
| --- |
| var request ;  if(XMLHttpRequest){  // 新式浏览器写法  request = new XMLHttpRequest();  }else{  //IE5,IE6写法  new ActiveXObject("Microsoft.XMLHTTP");  } |

### ****2.4.2 发送请求:****

|  |  |
| --- | --- |
| **方法** | **描述** |
| open(*method*,*url*,*async*) | 规定请求的类型、URL 以及是否异步处理请求。   * *method*：请求的类型；GET 或 POST * *url*：文件在服务器上的位置 * ***async*：true（异步）或 false（同步）**   当该boolean值为true时，服务器请求是异步进行的，也就是脚本执行send（）方法后不等待  服务器的执行结果，而是继续执行脚本代码；  当该boolean值为false时，服务器请求是同步进行的，也就是脚本执行send（）方法后等待  服务器的执行结果的返回，若在等待过程中超时，则不再等待，继续执行后面的脚本代码！  这里面有2个概念。 一个是Ajax请求分异步和同步2种模式。如果请求是同步的，在请求返回之前线程会一直阻塞，如果请求是在主线程中发起的，那就会造成整个浏览器阻塞。 另外一个就是主线程。这段话应该是针对HTML5说的，因为在HTML5以前，JavaScript是完全的单线程方式，主线程之外不存在其他线程。但在HTML5中增加了Worker对象，每个Worker运行在一个独立的线程中，Worker线程被阻塞一般是不会影响主线程和浏览器的。因此，如果非要使用同步的Ajax（这种情况应该很少见），那就放到Worker线程中吧，千万千万不要放到主线程里。 |
| send(*string*) | 将请求发送到服务器。   * *string*：仅用于 POST 请求 |

### ****2.4.3 POST请求注意点:****

如果想要像form表单提交数据那样使用POST请求,需要使用XMLHttpRequest对象的setRequestHeader()方法 来添加 HTTP 头。然后在 send() 方法中添加想要发送的数据：

xmlhttp.open("POST","ajax\_test.php",true);

xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");

xmlhttp.send("fname=Bill&lname=Gates");

### ****2.4.4 onreadystatechange事件****

当服务器给予我们反馈时,我们需要实现一些逻辑，我们需要监听请求状态

|  |  |
| --- | --- |
| **属性** | **描述** |
| onreadystatechange | 存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。 |
| readyState | 存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。   * 0: 请求未初始化 * 1: 服务器连接已建立 * 2: 请求已接收 * 3: 请求处理中 * 4: 请求已完成，且响应已就绪 |
| status | 200: "OK"  404: 未找到页面 |

### ****2.4.5 服务器响应内容****

如果响应的是普通字符串,使用responseText,如果响应的是XML,使用responseXML

|  |  |
| --- | --- |
| **属性** | **描述** |
| responseText | 获得字符串形式的响应数据。 |
| responseXML | 获得 XML 形式的响应数据。 |

## 2.5 XML

### 2.5.1 什么是xml

XML 指可扩展标记语言EXtensible Markup Language,他设计的时候是用来传递数据的,虽然格式跟HTML类似.

* 下面是一个XML示例

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <student>  <name>张三</name>  <age>20</age>  <skill>撸代码</skill>  </ student > |

* **xml是纯文本** XML是纯文本,这点跟HTML很像,所以我们可以用任何的文本编辑软件去打开编辑它

### 2.5.2 XML语法

* **XML声明** 第一行是XML的声明,指定XML版本(1.0)以及使用的编码(UTF-8万国码)

<?xml version="1.0" encoding="UTF-8"?>

* **自定义标签** XML中没有默认的标签,所有的标签都是我们定义者自定义的

<!-- 下列标签都是被允许的 -->

<fox></fox>

<name></name>

* **双标签** XML中没有单标签,都是双标签

<haha>标签内</haha>

* **根节点** XML中必须有一个根节点,所有的子节点都放置在根节点下

<root>

<name></name>

</root>

|  |
| --- |
| ~~<name>~~  ~~张三~~  ~~</name>~~  ~~<age>20</age>~~   * **注意：必须要有根节点** |

* **XML属性** 跟HTML一样,XML的标签里面也能够添加属性type = 'text',但是不建议这样用,而是使用标签的方式来表述内容(下半部分代码)

|  |
| --- |
| <!-- 使用属性配合标签表述信息 -->  <person sex="female">  <firstname>Anna</firstname>  <lastname>Smith</lastname>  </person>  <!-- 使用标签来表述信息 -->  <person>  <sex>female</sex>  <firstname>Anna</firstname>  <lastname>Smith</lastname>  </person> |

### ****2.5.3 XML解析****

因为XML就是标签,所以直接用解析Dom元素的方法解析即可

* **html代码**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Document</title>  </head>  <body>  <person id='personXML'>  <name>小明</name>  <age>18</age>  <skill>撸代码</skill>  </person>  </body>  </html> |

获取方式

|  |
| --- |
| <script type="text/javascript">  var xmlObj = document.getElementById("personXML");  var name = xmlObj.getElementsByTagName('name')[0].innerHTML;  console.log(name);  </script> |

获取本地服务XML并解析

|  |
| --- |
| var xhr = new XMLHttpRequest();  xhr.open('get','/data.xml');  xhr.send(null);  xhr.onreadystatechange = function(){  if (xhr.readyState == 4 && xhr.status == 200) {  console.log(xhr.responseText)  var xml=xhr.responseXML;  var name = xml.documentElement.getElementsByTagName("name");  console.log(name[0].innerHTML)  } |

案例：腾讯新闻rss

### 2.5.4 JSON

* JSON.parse()方法:将JSON字符串转化为JavaScript对象
* JSON.stringify()方法:将JavaScript对象,转化为JSON字符串

# 三、函数封装

1.定义方法参数

2.参数优化

3.参数格式化

# 四、案例

用户注册登陆