1.AJAX是什么

传统方式与服务器交互的"弊端"：如果要刷新网页中的一部分，必须整个网页重新加载！

Ajax：**局部刷新技术**：通过js的方式请求后台，获取数据，然后使用DOM技术根据后台获取的数据更新页面元素，在获取数据的过程中，浏览器不刷新。

2.AJAX快速入门（验证用户名）

163邮箱的注册页面：输入框离焦，发送ajax请求做后台查询，然后使用DOM技术弹出一个提示信息

ajax的典型应用：注册的时候，在没有提交表单的情况下，验证用户名是否存在 **编**

写步骤：

> 1.创建ajax的核心对象，ajax是以这个对象展开的 2.监听服务器的返回状态，当readyState（服务器对请求处理的步骤）和status（响应的状态码） 3.发送请求 4.获取响应

2.1.功能分析

步骤： 1.当输入用户名之后，输入框失去焦点，发送ajax请求，获取结果 2.根据结果使用DOM技术更新页面节点

2.2.前端js代码

```
<%@ page contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-
8" />
<title>Insert title here</title>
<script type="text/javascript">
    function checkName(username) {
        // 1、创建ajax的核心对象XMLHttpRequest
        var xmlhttp;
        if (window.XMLHttpRequest) {// code for IE7+, Firefox,
Chrome, Opera, Safari
            xmlhttp = new XMLHttpRequest();
        } else {// code for IE6, IE5
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        }

        // 2、监听服务器的返回状态
        xmlhttp.onreadystatechange = function() {
            // readyState=4:服务器完成对请求的处理  status=200：响应状
态码,成功
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
{
                // 4、获取来自服务器的响应结果
                var result = xmlhttp.responseText;
```

```
                if ( result == 1 ) {
                    document.getElementById("result").innerHTML = "已存在X";
                    document.getElementById("result").style.color = "red";
                } else {
                    document.getElementById("result").innerHTML = "可用√";
                    document.getElementById("result").style.color = "green";
                }
            }
        }

        // 3、发送请求
        var url = "${pageContext.request.contextPath}/checkName?username=" + username;
        xmlhttp.open("GET", url, true);
        xmlhttp.send();
    }
</script>
</head>
<body>
    <input id="username" onblur="checkName(value);" />
    <span id="result">结果</span>
</body>
</html>
```

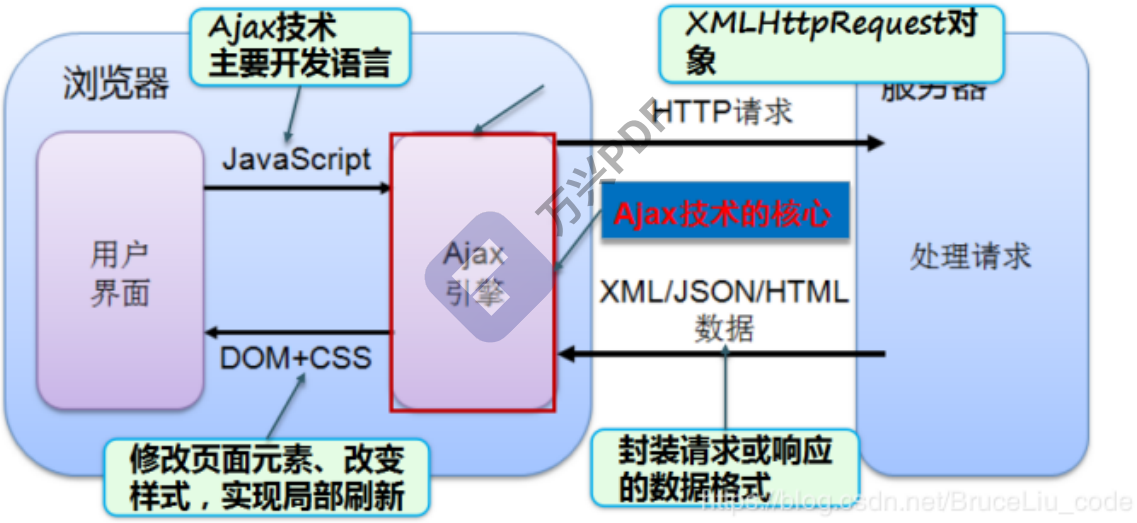
2.3.后端Servlet代码

```
package web;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@SuppressWarnings("all")
@WebServlet("/checkName")
public class CheckName extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // 获取用户名
        String username = request.getParameter("username");
```

```
17          if ( "zhangsan".equals(username) ) {
18              response.getWriter().write("1");
19          } else {
20              response.getWriter().write("0");
21  }
22      }
23
24      public void doPost(HttpServletRequest request,
    HttpServletResponse response)
25  throws ServletException, IOException {
26          doGet(request, response);
27      }
28  }
```

3.Ajax详解

3.1.Ajax工作原理



ajax无刷新技术，得益于浏览器内置的核心对象**XMLHttpRequest对象** 1.创建核心对象（需要兼容处理） 2.调用核心对象的方法发送请求 3.通过核心对象的responseText属性获取来自服务器的结果 4.根据结果做相应的处理

3.2.AJAX的核心对象XMLHttpRequest

获取核心对象，不同的浏览器，获取的方式不同，因此获取该对象，需要判断浏览器的类型，然后使用相应的方式去获取，一般是将兼容代码抽成一个函数，以后直接调用函数获取该对象 每次发送ajax请求的时候，都需要创建一个全新的XMLHttpRequest对象 抽取单独的js文件：xhr.js

```
function getXHR() {
    var xmlhttp;
    if (window.XMLHttpRequest) {
        // code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp = new XMLHttpRequest();
    } else {
        // code for IE6, IE5
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    return xmlhttp;
}
```

在需要使用ajax的页面中，引入该js文件，在获取该对象的时候，直接调用getXHR()方法去获取

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Insert title here</title>
<script type="text/javascript" src="js/xhr.js"></script>
<script type="text/javascript">
    function checkName(username) {
        //alert(username);
        // 1. 创建ajax的核心对象
        var xmlhttp = getXHR();
```


3.3.AJAX的监听函数onreadystatechange

不要手动写！

onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

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

因此，必须等核心对象的readyState属性为4并且status属性的值为200的时候，才能获取到来自服务器响应的结果，因此获取结果的代码，通常写成如下形式：

当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
```



```
1  监听服务器的返回状态
2  xmlhttp.onreadystatechange = function() {
3      // readyState=4:服务器完成对请求的处理  status=200：响应状态码,成功
4      if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
5          //获取来自服务器的响应结果
6          var result = xmlhttp.responseText;
7          // do something...
8      }
9  }
```

3.4.发送AJAX请求

向服务器发送请求

如需将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open() 和 send() 方法：

```
xmlhttp.open("GET","test1.txt",true);
xmlhttp.send();
```

方法	描述
open(*method,url,async*) 相当于在请求之前，对请求进行某些设置	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 如果是get请求，那么请求的参数跟在这个URL之后 • *async*：true（异步）或 false（同步）
send(*string*) 真正发送请求的方法	将请求发送到服务器。 • *string*：仅用于 POST 请求 string表示发送请求的时候，传递的参数：a=1&b=2

发送POST请求的时候还需要设置请求头字段Content-type的值为application/x-www-form-urlencoded

> 发送请求必须先调用open方法对象请求进行设置，然后再调用send方法才会发送请求！

3.5.获取AJAX响应

服务器响应

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。

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。

获取服务器的响应有2个属性：

responseText：获取文本

responseXML：获取服务器响应的xml格式的数据，后台需要使用DOM4j将java对象转换成xml格式的字符串，然后前端js还需要使用DOM技术去解析xml数据，和使用DOM操作HTML元素类似，相对来说比较麻烦，现在已经淘汰！

4.GET还是POST?

4.1.get和post的使用场景

get请求一般用于查询(列表)数据，而post请求用于提交数据（增、删、改） ajax中具体使用哪种请求方式，和传统方式一致！

4.2.GET请求和POST请求的区别

1.get请求的参数在地址栏中，因此相对post不安全，而post请求的参数在请求体中，相对get较安全 2.get请求发送的数据量有大小限制，而post理论上没有大小限制。3.get请求支持缓存，而post不支持缓存。（最重要的区别！）当浏览器发送一个get请求时，会将请求的资源加载本地的缓存中（硬盘中的某个位置），当再次请求该资源的时候，浏览器会优先从本地缓存中获取数据，如果缓存中没有数据，则请求服务器！这样可以减轻服务器的压力！而发送一个post请求时，浏览器根本就不会将请求的资源缓存到本地来，这是浏览器给我们提供一种机制！

缓存是根据请求地址做判断的，因此，如果请求的资源不希望从本地缓存中获取，可以在请求的get请求的地址栏的后面追加随机请求参数，以保证每次请求的地址不一样，浏览器从本地就找不到缓存数据，因此每次都会将请求发送给服务器！

缓存： 前端缓存：css、js、图片等静态资源 后台缓存：从数据库中查询的数据缓存

当发送get请求时，如果不希望服务器返回缓存的数据，可以在地址栏后随机拼接参数，因此走不走缓存，浏览器和服务器是根据请求的地址是否一致来进行判断的！

4.post请求需要在请求头中添加Content-Type

Ajax中发送get和post请求的方式：

为了避免这种情况，请向 URL 添加一个唯一的 ID：

```
xmlhttp.open("GET","demo_get.asp?t=" + Math.random(),true);
xmlhttp.send();
```

```
xmlhttp.open("POST","ajax_test.asp",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Bill&lname=Gates");
```

4.同步还是异步?

同步请求和异步请求的区别:

同步:单线程 请求发出之后,在服务器没有返回结果之前,客户端处理等待状态,必须等服务器响应结果之后,才能进行后续的操作。在服务器处理请求的这个过程中,客户端的js代码是出于阻塞状态。 **异步**:多线程 请求由另外一个线程来完成,完成之后处理响应的结果即可,客户端可以进行其他的操作,js代码也不会产生阻塞。

> ajax异步请求:open("get", "xx", true) ajax同步请求:open("get", "xx", false)

大多数情况下都是使用异步请求,几乎没有理由要写一个同步请求,因为不管是同步还是异步,都必须等服务器返回结果之后,才能根据结果做响应的处理,处理的代码可以在下方标注的地方处理!