Ajax

# Ajax

## 1.1Ajax基础概念

### 1.1.1同步和异步的概念

举个例子：普通B/S模式同步 AJAX技术(异步)

同步：提交请求->等待服务器处理->处理完毕返回 这个过程客户浏览器不能做任何事。

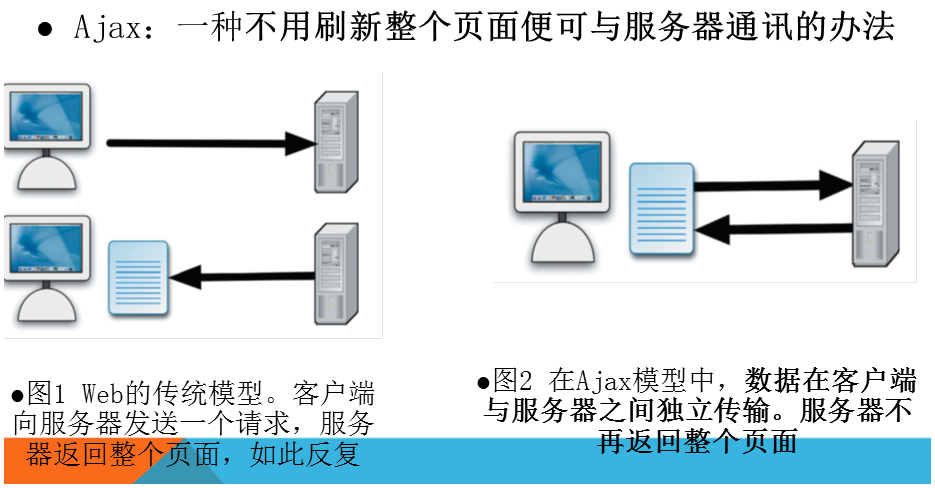
异步：请求通过事件触发->服务器处理->(这个时候浏览器可以做别的)->处理完毕。

同步是指：发送数据后，等到接收方发回响应后才发下一个数据包的通讯方式

异步是指：发送方发送数据后，不等待收方发回响应，接着发送下个数据包

### 1.1.2Ajax的简介

* 术语Ajax用来描述一组技术，它使浏览器可以为用户提供更为自然的浏览体验。
* 在Ajax之前，Web站点强制用户进入提交/等待/重新显示范例，用户的动作总是与服务器的“思考时间”同步。
* Ajax提供与服务器异步通信的能力，从而使用户从请求/响应的循环中解脱出来。借助于Ajax，可以在用户单击按钮时，使用JavaScript DHTML立即更新UI，并向服务器发出异步请求，以执行更新或查询数据库。当请求返回时，就可以使用JavaScript和CSS来相应地更新UI，而不是刷新整个页面。
* 最重要的是，用户甚至不知道浏览器正在与服务器通信：Web站点看起来是即时响应的。
* Ajax的技术的产生
  + **Ajax被认为是(Asynchronous[ei'siŋkrənəs] JavaScript and XML的缩写）。现在，允许浏览器与服务器通信而无须刷新当前页面的技术都被叫做Ajax.**
  + “Ajax”这个名字是在2005年2月，Adaptive Path的Jesse James Garrett在他的文章*Ajax:A New Approach to Web Application*中创造。
  + 而Ajax这项技术，是Google在Google Labs发布Google Maps和Google Suggest后真正为人所认识。



## 1.2Ajax的原理

* Ajax的核心是JavaScript对象XmlHttpRequest。

该对象在Internet Explorer 5中首次引入，它是一种支持异步请求的技术。简而言之，XmlHttpRequest使您可以使用JavaScript向服务器提出请求并处理响应，而不阻塞用户。

* AJAX采用异步交互过程。AJAX在用户与服务器之间引入一个中间媒介，从而消除了网络交互过程中的处理—等待—处理—等待缺点。用户的浏览器在执行任务时即装载了AJAX引擎。AJAX引擎用JavaScript语言编写，通常藏在一个隐藏的框架中。它负责编译用户界面及与服务器之间的交互。AJAX引擎允许用户与应用软件之间的交互过程异步进行，独立于用户与网络服务器间的交流。现在，可以用Javascript调用AJAX引擎来代替产生一个HTTP的用户动作，内存中的数据编辑、页面导航、数据校验这些不需要重新载入整个页面的需求可以交给AJAX来执行。
* 使用AJAX，可以为JSP、开发人员、终端用户带来可见的便捷：

## 1.3Ajax执行的过程

### 1.3.1创建XMLHttpRequst对象

* AJAX涉及到的技术中，Javascript、XMLHttpRequest、DOM、XML比较有用。  
  **A、XMLHttpRequest对象**  
  　　XMLHttpRequest是XMLHTTP组件的对象，通过这个对象，AJAX可以像桌面应用程序一样只同服务器进行数据层面的交换，而不用每次都刷新界面，也不用每次将数据处理的工作都交给服务器来做；这样既减轻了服务器负担又加快了响应速度、缩短了用户等待的时间。
* XMLHttpRequest最早是在IE5中以ActiveX组件的形式实现的。非W3C标准.
* 创建XMLHttpRequest对象（**由于非标准所以实现方法不统一**）
  + Internet Explorer把XMLHttpRequest实现为一个ActiveX对象
  + 其他浏览器（Firefox、Safari、Opera…）把它实现为一个本地的JavaScript对象。
  + **XMLHttpRequest在不同浏览器上的实现是兼容的**，所以可以用同样的方式访问XMLHttpRequest实例的属性和方法，而不论这个实例创建的方法是什么。

**function** createXMLHttpRequest(){

**var** XMLHttpReq;

**if**(window.XMLHttpRequest){ //火狐等浏览器

XMLHttpReq =**new** XMLHttpRequest();

}**else** **if**(window.ActiveXObject){//IE浏览器

**try**{

XMLHttpReq =**new** ActiveXObject("Msxml2.XMLHTTP");

}**catch**(e){

XMLHttpReq =**new** ActiveXObject("Microsoft.XMLHTTP");

}

}

**return** XMLHttpReq;

}

### 1.3.2打开和服务器的连接

//2.打开和服务器的连接

xmlReq.open("GET","AjaxServlet",**true**);

* open(method, url, asynch)
  + XMLHttpRequest 对象的 **open 方法允许程序员用一个Ajax调用向服务器发送请求**。
  + method：**请求类型**，类似 “GET”或”POST”的字符串。若只想从服务器检索一个文件，而不需要发送任何数据，使用GET(可以在GET请求里通过附加在URL上的查询字符串来发送数据，不过数据大小限制为2000个字符)。若需要向服务器发送数据，用POST。
  + 在某些情况下，有些浏览器会把多个XMLHttpRequest请求的结果缓存在同一个URL。如果对每个请求的响应不同，这就会带来不好的结果。把当前时间戳追加到URL的最后，就能确保URL的惟一性，从而避免浏览器缓存结果。
  + url：路径字符串，指向你所请求的服务器上的那个文件。可以是绝对路径或相对路径。
  + asynch：表示请求是否要异步传输，默认值为true(异步)。指定true，在读取后面的脚本之前，不需要等待服务器的相应。指定false，当脚本处理过程经过这点时，会停下来，一直等到Ajax请求执行完毕再继续执行。

### 1.3.3发送请求到服务器端

//3.发送数据

xmlReq.send(**null**);

* send(data)：
  + open 方法定义了 Ajax 请求的一些细节。**send 方法可为已经待命的请求发送指令**
  + data：将要传递给服务器的字符串。
  + **若选用的是 GET 请求，则不会发送任何数据， 给 send 方法传递 null 即可**：request.send(null);
  + 当向send()方法提供参数时，要确保open()中指定的方法是POST，如果没有数据作为请求体的一部分发送，则使用null.

### 1.3.4响应客户端请求

#### 1.3.4.1判断服务器的状态是否交互完毕

* onreadystatechange:
  + 该事件处理函数**由服务器触发**，而不是用户
  + 在 Ajax 执行过程中，服务器会通知客户端当前的通信状态。这依靠更新 XMLHttpRequest 对象的 readyState 来实现。**改变 readyState 属性是服务器对客户端连接操作的一种方式。每次 readyState 属性的改变都会触发** readystatechange**事件**
* readyState
  + readyState 属性**表示Ajax请求的当前状态**。它的值用数字代表。
    - 0 代表未初始化。 还没有调用 open 方法
    - 1 代表正在加载。 open 方法已被调用，但 send 方法还没有被调用
    - 2 代表已加载完毕。send 已被调用。请求已经开始
    - 3 代表交互中。服务器正在发送响应
    - **4 代表完成。响应发送完毕**

#### 1.3.4.2判断数据是否成功返回到客户端

* status
  + 服务器发送的每一个响应也都带有首部信息。三位数的状态码是服务器发送的响应中最重要的首部信息，并且属于超文本传输协议中的一部分。
  + 常用状态码及其含义：
    - 404 没找到页面(not found)
    - 403 禁止访问(forbidden)
    - 500 内部服务器出错(internal service error)
    - 200 一切正常(ok)
    - 304 没有被修改(not modified)(服务器返回304状态，表示源文件没有被修改 )
  + **在 XMLHttpRequest 对象中，服务器发送的状态码都保存在 status 属性里。通过把这个值和 200 或 304 比较，可以确保服务器是否已发送了一个成功的响应**

xmlReq.onreadystatechange=**function**(){

**if**(xmlReq.readyState==4){

**if**(xmlReq.status==200){

}

}

}

#### 1.3.4.3处理服务器端响应的数据

服务器端响应的数据分为文本text类型和xml类型

* responseText
  + XMLHttpRequest 的 responseText 属性**包含了从服务器发送的数据**。它是一个HTML,XML或普通文本，这取决于服务器发送的内容。
  + 当 readyState 属性值变成 4 时, responseText 属性才可用，表明 Ajax 请求已经结束。
* responseXML
  + 如果服务器返回的是 XML， 那么数据将储存在 responseXML 属性中。
  + 只用服务器发送了带有正确首部信息的数据时， responseXML 属性才是可用的。 **MIME 类型必须为 text/xml**

### 1.3.5完整案例

**AjaxDemo1.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"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>

</head>

<script type=*"text/javascript"* src=*"testAjax.js"*></script>

<body>

杭州归谷欢迎你:

<form >

<input type=*"button"* name=*"ok"* id=*"ok"* value=*"测试服务器"* >

</form>

</body>

</html>

**testAjax.js**

/\*\*

\* AJax建立和服务器端的连接 接受服务器的请求，处理服务器返回的数据

\* 开发步骤：

\* 1.创建XMLHttpRequest对象

\* 2.打开和服务器的连接

\* 3.发送数据到服务器端

\* 4.接收服务器端的响应

\*/

**function** createXMLHttpRequest(){

**var** XMLHttpReq;

**if**(window.XMLHttpRequest){ //火狐等浏览器

XMLHttpReq =**new** XMLHttpRequest();

}**else** **if**(window.ActiveXObject){//IE浏览器

**try**{

XMLHttpReq =**new** ActiveXObject("Msxml2.XMLHTTP");

}**catch**(e){

XMLHttpReq =**new** ActiveXObject("Microsoft.XMLHTTP");

}

}

**return** XMLHttpReq;

}

window.onload=**function**(){

document.getElementById("ok").onclick=**function**(){

//1.创建XMLHttpRequest对象

**var** xmlReq=createXMLHttpRequest();

//2.打开和服务器的连接

xmlReq.open("GET","AjaxServlet",**true**);

//3.发送数据

xmlReq.send(**null**);

//4.响应服务的端的返回的数据

xmlReq.onreadystatechange=**function**(){

**if**(xmlReq.readyState==4){//判断当前对象的状态 4代表完成

**if**(xmlReq.status==200){//代表信息已经成功返回到客户端 可以对数据进行相关的处理

//获取服务器端返回的数据 获取文本形式的数据

**var** xmlDoc=xmlReq.responseText;

alert("返回的数据是:"+xmlDoc);

}

}

}

}

}

**package** com.guigu.servlet;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/AjaxServlet")

**public** **class** AjaxServlet **extends** HttpServlet {

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

PrintWriter pw =response.getWriter();

pw.print("connection ok succee");

pw.flush();

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

doGet(request, response);

}

}

## 1.4邮箱注册

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"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>

</head>

<script type=*"text/javascript"*>

**var** XMLHttpReq;

**function** createXMLHttpRequest(){

**if**(window.XMLHttpRequest){ //火狐等浏览器

XMLHttpReq =**new** XMLHttpRequest();

}**else** **if**(window.ActiveXObject){//IE浏览器

**try**{

XMLHttpReq =**new** ActiveXObject("Msxml2.XMLHTTP");

}**catch**(e){

XMLHttpReq =**new** ActiveXObject("Microsoft.XMLHTTP");

}

}

}

**function** back(){

**if**(XMLHttpReq.readyState==4){

**if**(XMLHttpReq.status==200){

**var** xmlDoc=XMLHttpReq.responseXML;

**var** un =document.getElementById("un");

**var** result=xmlDoc.getElementsByTagName("result")[0].firstChild.nodeValue;

**if**(result==1){

**var** error="该邮箱已经被注册,请重新输入...";

un.innerHTML=error;

}**else** **if**(result==0){

un.innerHTML="恭喜可以使用";

}

}

}

}

**function** checkUserName(){

**var** name=document.getElementById("username").value;

**var** url="CheckUserNameServlet?username="+name;

createXMLHttpRequest();

XMLHttpReq.onreadystatechange=back;

XMLHttpReq.open("GET",url,**true**);

XMLHttpReq.send(**null**);

}

</script>

<body>

<form action=*""*>

用户名:<input type=*"text"* name=*"username"* id=*"username"* onblur="checkUserName()"><span id=*"un"*></span><br/>

密码：<input type=*"password"* name=*"password"*><br/>

<input type=*"submit"* value=*"注册"*>

</form>

</body>

</html>

**package** com.guigu.servlet;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** com.guigu.dao.UserDAO;

@WebServlet("/jsp/CheckUserNameServlet")

**public** **class** CheckUserNameServlet **extends** HttpServlet {

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

response.setContentType("text/xml;charset=UTF-8");

PrintWriter out =response.getWriter();

String username=request.getParameter("username");

UserDAO dao =**new** UserDAO();

**boolean** flag=dao.checkUserName(username);

//以xml的形式返回到客户端

StringBuilder sb =**new** StringBuilder(); //<response><result>1</result></response> <response><result>0</result></response>

sb.append("<response>");

sb.append("<result>");

**if**(flag) {

//用户存在

sb.append("1");

}**else** {

//用户不存在

sb.append("0");

}

sb.append("</result>");

sb.append("</response>");

String xml =sb.toString();

out.println(xml);

out.flush();

out.close();

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

doGet(request, response);

}

}

**package** com.guigu.dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** com.guigu.utils.JdbcUtils;

/\*\*

\* <p>project\_name:JavaAjaxJquery</p>

\* <p>package\_name:com.guigu.dao.UserDAO</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年1月8日上午10:24:44 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** UserDAO {

**public** **boolean** checkUserName(String name) {

Connection conn = JdbcUtils.*getConnection*();

PreparedStatement ps =**null**;

ResultSet rs =**null**;

**boolean** flag =**false**;

String sql="select \* from ajaxuser where name=?";

**try** {

ps=conn.prepareStatement(sql);

ps.setString(1, name);

rs=ps.executeQuery();

**if**(rs.next()) {

flag=**true**;

}

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** flag;

}

**public** **static** **void** main(String[] args) {

UserDAO dao =**new** UserDAO();

System.***out***.println(dao.checkUserName("lisi"));

}

}

## 1.5二级联动

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"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>

</head>

<script type=*"text/javascript"*>

**var** XMLHttpReq;

**function** createXMLHttpRequest() {

**if** (window.XMLHttpRequest) { //火狐等浏览器

XMLHttpReq = **new** XMLHttpRequest();

} **else** **if** (window.ActiveXObject) {//IE浏览器

**try** {

XMLHttpReq = **new** ActiveXObject("Msxml2.XMLHTTP");

} **catch** (e) {

XMLHttpReq = **new** ActiveXObject("Microsoft.XMLHTTP");

}

}

}

**function** back(){

**if**(XMLHttpReq.readyState==4){

**if**(XMLHttpReq.status==200){

**var** xmlDoc=XMLHttpReq.responseXML;

**var** xCity=xmlDoc.getElementsByTagName('select');

**var** select\_root=document.getElementById("city");

select\_root.options.length=0;

**for**(**var** i=0;i<xCity.length;i++){

**var** xValue=xCity[i].childNodes[0].firstChild.nodeValue;

**var** xText=xCity[i].childNodes[1].firstChild.nodeValue;

**var** option =**new** Option(xText,xValue);

select\_root.add(option);

}

}

}

}

**function** getCity() {

**var** id=document.getElementById("province").value;

**var** url="${pageContext.request.contextPath}/CityServlet?id="+id;

createXMLHttpRequest();

XMLHttpReq.onreadystatechange=back;

XMLHttpReq.open("GET",url,**true**);

XMLHttpReq.send(**null**);

}

</script>

<body>

<form action=*""*>

<table width=*"70%"* border=*"1"*>

<tr>

<td align=*"center"*>二级联动</td>

</tr>

<tr>

<td>

<select name=*"province"* id=*"province"* onchange=*"getCity()"*>

<option value=*"0"*>请选择所在的省份</option>

<option value=*"1"*>浙江省</option>

<option value=*"2"*>山东省</option>

<option value=*"3"*>江苏省</option>

</select>

<select name=*"city"* id=*"city"*>

<option value=*"0"*>请选择所在的城市</option>

</select>

</td>

</tr>

</table>

</form>

</body>

</html>

**package** com.guigu.servlet;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** com.sun.prism.Presentable;

@WebServlet("/CityServlet")

**public** **class** CityServlet **extends** HttpServlet {

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

response.setContentType("text/xml;charset=UTF-8");

PrintWriter out=response.getWriter();

String tagetId=request.getParameter("id");

String xml\_start="<selects>";

String xml\_end="</selects>";

String xml="";

**if**(tagetId.equals("0")) {

xml="<select><value>0<value><text>请选择城市</text></select>";

}**else** **if**(tagetId.equals("1")) {

xml="<select><value>1</value><text>杭州市</text></select>";

xml+="<select><value>2</value><text>宁波市</text></select>";

xml+="<select><value>3</value><text>温州市</text></select>";

xml+="<select><value>4</value><text>金华市</text></select>";

}**else** **if**(tagetId.equals("2")) {

xml="<select><value>1</value><text>济南市</text></select>";

xml+="<select><value>2</value><text>青岛市</text></select>";

xml+="<select><value>3</value><text>潍坊市</text></select>";

xml+="<select><value>4</value><text>济宁市</text></select>";

}**else** **if**(tagetId.equals("3")) {

xml="<select><value>1</value><text>南京市</text></select>";

xml+="<select><value>2</value><text>苏州市</text></select>";

xml+="<select><value>3</value><text>南通市</text></select>";

xml+="<select><value>4</value><text>常州市</text></select>";

}

String last\_xml =xml\_start+xml+xml\_end;

out.println(last\_xml);

out.flush();

out.close();

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

doGet(request, response);

}

}

# 二、JQuery

## 2.1Jquery的基础知识

jQuery由美国人John Resig创建，至今已吸引了来自世界各地的众多

javascript高手加入其team。

jQuery是继prototype之后又一个优秀的Javascript框架。其宗旨是——WRITE LESS,DO MORE,写更少的代码,做更多的事情。

它是轻量级的js库(压缩后只有21k) ，这是其它的js库所不及的，它兼容CSS3，还兼容各种浏览器 （IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+）。

jQuery是一个快速的，简洁的javaScript库，使用户能更方便地处理HTML documents、events、实现动画效果，并且方便地为网站提供AJAX交互。

jQuery还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的插件可供选择。

jQuery能够使用户的html页保持代码和html内容分离，也就是说，不用再在html里面插入一堆js来调用命令了，只需定义id即可。

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<script type=*"text/javascript"*>

$(document).ready(**function**(){

$("p").click(**function**(){

$(**this**).hide();

});

});

</script>

<body>

<p>如果你点击我，我将会消失</p>

<p>你来点我啊</p>

<p>你在点一下</p>

</body>

</html>

## 2.2JQuery的语法格式

$(jsObject/页面内容)

### 2.2.1$(document).ready()

$(document).ready() 等价于document.onload事件

区别：$(document).ready()可以执行多次，而document.onload只能执行一次

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<script type=*"text/javascript"*>

$(document).ready(**function**(){

alert("hello");

});

$(document).ready(**function**(){

alert("guigu");

});

</script>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<script type=*"text/javascript"*>

window.onload=**function**(){

alert("hello");

}

window.onload=**function**(){

alert("java");

}

</script>

<body>

</body>

</html>

### 2.2.2Jquery数据的获取

document.getElementById(“username”);

document.getElementsByName(“username”);

document.getElementsByTagName(“input”);

**案例一：通过id方式获取Jquery对象**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<script type=*"text/javascript"*>

$(document).ready(**function**(){

$("#ch1").html("SpringMVC");

alert($("#ch1").length)

});

</script>

<body>

<div id=*"ch1"*>hello</div><br/>

<span id=*"ch1"*>java</span><br/>

<input type=*"text"* id=*"ch1"* name=*"username"* value=*"Kafka"*><br/>

</body>

</html>

结果：是只有第一个div标签的内容改变了，span标签和input标签内容并未改变，返回的长度也是1 。

注意这里得到的是数组集合的形式 也就是获取到的是id是ch1的数组 而默认是取第一个值

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<script type=*"text/javascript"*>

$(document).ready(**function**(){

$("a").html("杭州归谷");

alert($("a").length);

});

</script>

<body>

<a href=*"#"*>百度</a><br/>

<a href=*"#"*>腾讯</a><br/>

<a href=*"#"*>阿里</a><br/>

<a href=*"#"*>华为</a><br/>

</body>

</html>

**通过运行结果得到 四个a标签都改变 而长度也是4**

## 2.3Jquery对象和JavaScript对象之间的转换

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<input type=*"text"* value=*"张三"* id=*"username"* name=*"username"*>

<script type=*"text/javascript"*>

//1.把dom对象转换为Jquery对象

//获取dom对象 --->利用js的方式获取

**var** usernameElement=document.getElementById("username"); //dom对象

//dom对象只能使用dom的方法 不能使用jquery的方法

alert(usernameElement.value);

//dom对象转换为jquery对象只需要加上$

**var** $usernameElement =$(usernameElement); //jquery对象

//jquery对象只能使用jquery的方法 不能使用dom对象的方法

alert($usernameElement.val());

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//2.把jquery对象转换为dom对象

//可以直接获取jquery对象 利用jquery方法获取到的都是jquery对象

**var** $uname =$("#username");//jquery对象

alert($uname.val());

//2.1把jquery利用下标的形式获取dom对象

**var** username=$uname[0];

alert(username.value)

//2.2 利用jquery本身的方法转换为dom对象

**var** usernameEle =$uname.get(0);

alert(usernameEle.value);

</script>

<body>

</body>

</html>

## 2.4 Jquery选择器

选择器是 jQuery 的根基, 在 jQuery 中, 对事件处理, 遍历 DOM 和 Ajax 操作都依赖于选择器。

### 2.4.1基本选择器

基本选择器是 jQuery 中最常用的选择器, 也是最简单的选择器, 它通过元素 id, class 和标签名来查找 DOM 元素(在网页中 id 只能使用一次, class 允许重复使用).

**1、#id 用法: *$(”#myDiv”);* 返回值 单个元素的组成的集合**

**说明: 这个就是直接选择html中的*id=”myDiv”***

**2、Element 用法: *$(”div”)* 返回值 集合元素**

**说明: element的英文翻译过来是”元素”,所以element其实就是html已经定义的标签元素,例如 div, input, a等等.**

**3、class 用法: *$(”.myClass”)* 返回值 集合元素**

**说明: 这个标签是直接选择html代码中class=”myClass”的元素或元素组(因为在同一html页面中class是可以存在多个同样值的).**

**4、\* 用法: *$(”\*”)* 返回值 集合元素**

**说明: 匹配所有元素,多用于结合上下文来搜索**

**5、selector1, selector2, selectorN 用法: *$(”div,span,p.myClass”)* 返回值 集合元素**

**说明: 将每一个选择器匹配到的元素合并后一起返回.你可以指定任意多个选择器, 并将匹配到的元素合并到一个结果内.其中p.myClass是表示匹配元素**

**p class=”myClass”**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<html>

<head>

<title>4.html</title>

<meta http-equiv=*"keywords"* content=*"keyword1,keyword2,keyword3"*>

<meta http-equiv=*"description"* content=*"this is my page"*>

<meta http-equiv=*"content-type"* content=*"text/html; charset=UTF-8"*>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<style type=*"text/css"*>

**div,** **span** {

width: *140px*;

height: *140px*;

margin: *20px*;

background: *#9999cc*;

border: *#000 1px solid*;

float: *left*;

font-size: *17px*;

font-family: *Roman*;

}

**div***.mini* {

width: *30px*;

height: *30px*;

background: *#cc66ff*;

border: *#000 1px solid*;

font-size: *12px*;

font-family: *Roman*;

}

</style>

</head>

<body>

<input type=*"button"* value=*"保存"* class=*"mini"* name=*"ok"* />

<input type=*"button"* value=*"改变id 为 one 的背景颜色 "* id=*"b1"* />

<input type=*"button"* value=*"改变元素名为div的所有元素的背景颜色为#00FFF"* id=*"b2"* />

<input type=*"button"* value=*"改变class为mini的所有属性的背景颜色为#FF0033"* id=*"b3"* />

<input type=*"button"* value=*"改变所有元素的背景颜色#00FF33"* id=*"b4"* />

<input type=*"button"* value=*"改变所有<span>元素和id为two的元素 的背景颜色为#3399ff"*

id=*"b5"* />

<input type=*"text"* value=*"zhangsan"* id=*"tid"* />

<h1>天气冷了</h1>

<h2>天气 又冷了</h2>

<div id=*"one"*>id 为one</div>

<div id=*"two"* class=*"mini"*>

id为two class为mini

<div class=*"mini"*>class 为mini</div>

</div>

<div class=*"one"*>

class为one

<div class=*"mini"*>class是mini</div>

<div class=*"mini"*>class是mini</div>

</div>

<div class=*"one1"*>

class为one

<div class=*"mini1"*>class是mini1</div>

<div class=*"mini"*>class是mini</div>

</div>

<br />

<div id=*"mover"*>动画</div>

<br />

<span class=*"spanone"*> span</span>

<span class=*"mini"*> span</span>

<script type=*"text/javascript"*>

//<input type="button" value="改变id 为 one 的背景颜色 " id="b1"/>

$("#b1").click(**function**(){

$("#one").css("background","red");

});

//<input type="button" value="改变元素名为div的所有元素的背景颜色为#00FFF" id="b2"/>

$("#b2").click(**function**(){

$("div").css("background","#000FFF");

});

//<input type="button" value="改变class为mini的所有属性的背景颜色为#FF0033" id="b3"/>

$("#b3").click(**function**(){

$(".mini").css("background","#FF0033");

});

//<input type="button" value="改变所有元素的背景颜色#00FF33" id="b4"/>

$("#b4").click(**function**(){

$("\*").css("background","#FF0033");

});

//<input type="button" value="改变所有<span>元素和id为two的元素 的背景颜色为#3399ff" id="b5"/>

$("#b5").click(**function**(){

$("span,#two").css("background","#3399ff");

});

</script>

</body>

</html>

### 2.4.2过滤选择器

**过滤选择器**主要是通过特定的过滤规则来筛选出所需的 DOM 元素, 该选择器**都以 “:” 开头**

按照不同的过滤规则, 过滤选择器可以分为基本过滤, 内容过滤, 可见性过滤, 属性过滤, 子元素过滤和表单对象属性过滤选择器.

**1、:first  
用法: $(”tr:first”) ; 返回值 单个元素的组成的集合**

**说明: 匹配找到的第一个元素**

**2、:last  
用法: $(”tr:last”) 返回值 集合元素**

**说明: 匹配找到的最后一个元素.与 :first 相对应.**

**3、:not(selector)  
用法: $(”input:not(:checked)”)返回值 集合元素**

**说明: 去除所有与给定选择器匹配的元素.有点类似于”非”,意思是没有被选中的input(当input的type=”checkbox”).**

**4、:even  
用法: $(”tr:even”) 返回值 集合元素**

**说明: 匹配所有索引值为偶数的元素，从 0 开始计数.js的数组都是从0开始计数的.例如要选择table中的行,因为是从0开始计数,所以table中的第一个tr就为偶数0.**

**5、: odd  
用法: $(”tr:odd”) 返回值 集合元素**

**说明: 匹配所有索引值为奇数的元素,和:even对应,从 0 开始计数.**

**6、:eq(index)  
用法: $(”tr:eq(0)”) 返回值 集合元素**

**说明: 匹配一个给定索引值的元素.eq(0)就是获取第一个tr元素.括号里面的是索引值,不是元素排列数.**

**7、:gt(index)  
用法: $(”tr:gt(0)”) 返回值 集合元素**

**说明: 匹配所有大于给定索引值的元素.**

**8、:lt(index)  
用法: $(”tr:lt(2)”) 返回值 集合元素**

**说明: 匹配所有小于给定索引值的元素.**

**9、:header(固定写法)  
用法: $(”:header”).css(”background”, “#EEE”) 返回值 集合元素**

**说明: 匹配如 h1, h2, h3之类的标题元素.这个是专门用来获取h1,h2这样的标题元素.**

**10、:animated(固定写法) 返回值 集合元素**

**说明: 匹配所有正在执行动画效果的元素**

<script type=*"text/javascript"*>

//<input type="button" value="改变第一个 div 元素的背景色为 #0000FF" id="b1"/>

$("#b1").click(**function**(){

$("div:first").css("background","#0000FF");

});

//<input type="button" value="改变最后一个 div 元素的背景色为 #0000FF" id="b2"/>

$("#b2").click(**function**(){

$("div:last").css("background","#0000FF");

});

//<input type="button" value="改变class不为 one 的所有 div 元素的背景色为 #0000FF" id="b3"/>

$("#b3").click(**function**(){

$("div:not(.one)").css("background","#0000FF");

});

//<input type="button" value="改变索引值为偶数的 div 元素的背景色为 #0000FF" id="b4"/>

$("#b4").click(**function**(){

$("div:even").css("background","#0000FF");

});

//<input type="button" value="改变索引值为奇数的 div 元素的背景色为 #0000FF" id="b5"/>

$("#b5").click(**function**(){

$("div:odd").css("background","#0000FF");

});

//<input type="button" value="改变索引值为大于 3 的 div 元素的背景色为 #0000FF" id="b6"/>

$("#b6").click(**function**(){

$("div:gt(3)").css("background","#0000FF");

});

//<input type="button" value="改变索引值为等于 3 的 div 元素的背景色为 #0000FF" id="b7"/>

$("#b7").click(**function**(){

$("div:eq(3)").css("background","#0000FF");

});

//<input type="button" value="改变索引值为小于 3 的 div 元素的背景色为 #0000FF" id="b8"/>

$("#b8").click(**function**(){

$("div:lt(3)").css("background","#0000FF");

});

//<input type="button" value="改变所有的标题元素的背景色为 #0000FF" id="b9"/>

$("#b9").click(**function**(){

$(":header").css("background","#0000FF");

});

//<input type="button" value="改变当前正在执行动画的所有元素的背景色为 #0000FF" id="b10"/>

//$("#move").slideToggle("fast");

**function** move(){

$("#move").slideToggle("2220",move);

}

move();

$("#b10").click(**function**(){

$(":animated").css("background","red");

});

</script>

案例

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<body>

<ul>

<li>aaaaaaaaaaaaaaaaa</li>

<li>bbbbbbbbbbbbbbbbb</li>

<li>ccccccccccccccccc</li>

<li>ddddddddddddddddd</li>

<li>eeeeeeeeeeeeeeeee</li>

<li>fffffffffffffffff</li>

</ul>

<script type=*"text/javascript"*>

$(**function**(){

$("li").mouseover(setColor).mouseout(setColor1);

**function** setColor(){

$(**this**).css("color","red");

}

**function** setColor1(){

$(**this**).css("color","#000");

}

});

</script>

</body>

</html>

### 2.4.3内容过滤选择器

内容过滤选择器的过滤规则主要**体现在它所包含的子元素和文本内容上**

**1、:contains(text)  
用法: $(”div:contains(’John’)”) 返回值 集合元素**

**说明: 匹配包含给定文本的元素.这个选择器比较有用，当我们要选择的不是dom标签元素时,它就派上了用场了,它的作用是查找被标签”围”起来的文本内容是否符合指定的内容的.**

**2、:empty  
用法: $(”td:empty”) 返回值 集合元素**

**说明: 匹配所有不包含子元素或者文本的空元素**

**3、:has(selector)**

**用法: $(”div:has(p)”).addClass(”test”) 返回值 集合元素**

**说明: 匹配含有选择器所匹配的元素的元素.这个解释需要好好琢磨,但是一旦看了使用的例子就完全清楚了:给所有包含p元素的div标签加上class=”test”.**

**4、:parent  
用法: $(”td:parent”) 返回值 集合元素**

**说明: 匹配含有子元素或者文本的元素.注意:这里是”:parent”,可不是”.parent”哦!感觉与上面讲的”:empty”形成反义词.**

<script type=*"text/javascript"*>

//<input type="button" value="改变含有文本 ‘di’ 的 div 元素的背景色为 #0000FF" id="b1"/>

$("#b1").click(**function**(){

$("div:contains('di')").css("background","red");

});

//<input type="button" value="改变不包含子元素(或者文本元素) 的 div 空元素的背景色为 #0000FF" id="b2"/>

$("#b2").click(**function**(){

$("div:empty").css("background","red");

});

//<input type="button" value="改变含有 class 为 mini 元素的 div 元素的背景色为 #0000FF" id="b3"/>

$("#b3").click(**function**(){

$("div:has(.mini)").css("background","red");

});//\*\*\*

//<input type="button" value="改变含有子元素(或者文本元素)的div元素的背景色为 #0000FF" id="b4"/>

$("#b4").click(**function**(){

$("div:parent").css("background","red");

});

//<input type="button" value="改变不含有文本 di的 div 元素的背景色" id="b5"/>

$("#b5").click(**function**(){

$("div:not(:contains('di'))").css("background","red");

});

</script>

### 2.4.4可见性过滤选择器

可见度过滤选择器是**根据元素的可见和不可见状态**来选择相应的元素

**1、:hidden  
用法: $(”tr:hidden”) 返回值 集合元素**

说明: 匹配所有的不可见元素，input 元素的 type 属性为 “hidden” 的话也会被匹配到.意思是css中display:none和input type=”hidden”的都会被匹配到.同样,要在脑海中彻底分清楚冒号”:”, 点号”.”和逗号”,”的区别.

**2、:visible  
用法: $(”tr:visible”) 返回值 集合元素**

说明: 匹配所有的可见元素.

<script type=*"text/javascript"*>

//<input type="button" value="改变所有可见的div元素的背景色为 #0000FF" id="b1"/>

$("#b1").click(**function**(){

$("div:visible").css("background","red");

});

//<input type="button" value="选取所有不可见的元素, 利用 jQuery 中的 show() 方法将它们显示出来, 并设置其背景色为 #0000FF" id="b2"/>

$("#b2").click(**function**(){

$("div:hidden").show().css("background","red");

});

// <input type="button" value="选取所有的文本隐藏域, 并打印它们的值" id="b3"/>

$("#b3").click(**function**(){

**var** $hidinput=$("input:hidden");

$hidinput.each(**function**(){

alert($(**this**).val());//Jquery对象只能使用Jquery方法

});

});

</script>

### 2.4.5表单过滤选择器

**此选择器主要对所选择的表单元素进行过滤**

**1、:enabled  
用法: $(”input:enabled”) 返回值 集合元素**

**说明: 匹配所有可用元素.意思是查找所有input中不带有disabled=”disabled”的input.不为disabled,当然就为enabled啦.**

**2、:disabled  
用法: $(”input:disabled”) 返回值 集合元素**

**说明: 匹配所有不可用元素.与上面的那个是相对应的**.

**3、:checked  
用法: $(”input:checked”) 返回值 集合元素**

**说明: 匹配所有选中的被选中元素(复选框、单选框等，不包括select中的option).这话说起来有些绕口.**

**4、:selected  
用法: $(”select option:selected”) 返回值 集合元素**

**说明: 匹配所有选中的option元素.**

**1、:input  
用法: $(”:input”) ; 返回值 集合元素**

**说明:匹配所有 input, textarea, select 和 button 元素**

**2、:text  
用法: $(”:text”) ; 返回值 集合元素**

**说明: 匹配所有的单行文本框.**

**3、:password  
用法: $(”:password”) ; 返回值 集合元素**

**说明: 匹配所有密码框.**

**4、:radio  
用法: $(”:radio”) ; 返回值 集合元素**

**说明: 匹配所有单选按钮.**

**5、:checkbox  
用法: $(”:checkbox”) ; 返回值 集合元素**

**说明: 匹配所有复选框**

**6、:submit  
用法: $(”:submit”) ; 返回值 集合元素**

**说明: 匹配所有提交按钮.**

**7、:image  
用法: $(”:image”) 返回值 集合元素**

**说明: 匹配所有图像域.**

**8、:reset  
用法: $(”:reset”) ; 返回值 集合元素**

**说明: 匹配所有重置按钮.**

**9、:button  
用法: $(”:button”) ; 返回值 集合元素**

**说明: 匹配所有按钮.这个包括直接写的元素button.**

**10、:file  
用法: $(”:file”) ; 返回值 集合元素**

**说明: 匹配所有文件域.**

**11、:hidden  
用法: $(”input:hidden”) ; 返回值 集合元素**

**说明: 匹配所有不可见元素，或者type为hidden的元素.这个选择器就不仅限于表单了,除了匹配input中的hidden外,那些style为hidden的也会被匹配.**

**注意: 要选取input中为hidden值的方法就是上面例子的用法,但是直接使用 “:hidden” 的话就是匹配页面中所有的不可见元素,包括宽度或高度为0的,**

<script type=*"text/javascript"*>

//<input type="button" value="利用 jQuery 对象的 val() 方法改变表单内可用 <input> 元素的值" id="b1"/>

$("#b1").click(**function**(){

$("input[type=text]:enabled").each(**function**(){

$(**this**).val("guigu");

});

});

//<input type="button" value="利用 jQuery 对象的 val() 方法改变表单内不可用 <input> 元素的值" id="b2"/>

$("#b2").click(**function**(){

$("input[type=text]:disabled").each(**function**(){

$(**this**).val("hellojava");

});

});

//<input type="button" value="利用 jQuery 对象的 length 属性获取多选框选中的个数" id="b3"/>

$("#b3").click(**function**(){

/\* $("input[type=checkbox]:checked").each(function(){

alert($(this).val());

}); \*/

//alert($("input[type=checkbox]:checked").length);

//获取单选按钮

$("input[type=radio]:checked").each(**function**(){

alert($(**this**).val());

});

});

//<input type="button" value="利用 jQuery 对象的 text() 方法获取下拉框选中的内容" id="b4"/>

$("#b4").click(**function**(){

$("select option:selected").each(**function**(){

alert($(**this**).text());

});

});

</script>

## 2.5 Jquery操作DOM对象

HTML常见的DOM操作

查找节点 增加节点 删除节点 复制节点 替换节点

### 2.5.1添加节点

**\* append(content) :向每个匹配的元素的内部的结尾处追加内容**

**\* appendTo(content) :将每个匹配的元素追加到指定的元素中的内部结尾处**

**\* prepend(content):向每个匹配的元素的内部的开始处插入内容**

**\* prependTo(content) :将每个匹配的元素插入到指定的元素内部的开始处**

**\* after(content) :在每个匹配的元素之后插入内容**

**\* before(content):在每个匹配的元素之前插入内容**

**\* insertAfter(content):把所有匹配的元素插入到另一个、指定的元素元素集合的后面**

**\* insertBefore(content) :把所有匹配的元素插入到另一个、指定的元素元素集合的前面 。**

**以上方法不但能将新创建的 DOM 元素插入到文档中, 也能对原有的 DOM 元素进行移动.**

<script type=*"text/javascript"*>

// $("#city").append($("#fk"));

// $("#city").appendTo($("#fk"));

//$("#city").prepend($("#fk"));

//$("#city").prependTo($("#fk"));

//$("#city").after($("#fk"));

//$("#city").before($("#fk"));

</script>

### 2.5.2创建节点

<script type=*"text/javascript"*>

//创建节点信息

**var** $sh=$("<li></li>");//<li></li>

$sh.attr("id","sh");//<li id="sh" name="shanghai">上海</li>

$sh.attr("name","shanghai");

$sh.text("上海");

$("#city").append($sh);

</script>

### 2.5.3删除节点

<script type=*"text/javascript"*>

//移除指定的节点

//var $cq=$("#cq").remove();

//清空节点信息

$("#city").empty();

</script>

### 2.5.4替换节点

<script type=*"text/javascript"*>

$("<h1>hello</h1>").replaceAll("span");

</script>

### 2.5.5案例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<html>

<head>

<title>17.html</title>

<meta http-equiv=*"keywords"* content=*"keyword1,keyword2,keyword3"*>

<meta http-equiv=*"description"* content=*"this is my page"*>

<meta http-equiv=*"content-type"* content=*"text/html; charset=UTF-8"*>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

</head>

<body>

<form method=*"post"* action=*""* >

请选择你的爱好！

<br/>

<input type=*"checkbox"* id=*"checkAll\_2"*/>全选/全不选<br/>

<input type=*"checkbox"* name=*"items"* value=*"足球"*/>足球

<input type=*"checkbox"* name=*"items"* value=*"篮球"*/>篮球

<input type=*"checkbox"* name=*"items"* value=*"游泳"*/>游泳

<input type=*"checkbox"* name=*"items"* value=*"唱歌"*/>唱歌

<br/>

<input type=*"button"* id=*"checkAll"* value=*"全选"*/>

<input type=*"button"* id=*"checkNo"* value=*"全不选"*/>

<input type=*"button"* id=*"checkRev"* value=*"反选"*/>

<input type=*"button"* id=*"send"* value=*"提交"*/>

</form>

<script type=*"text/javascript"*>

//全选

$("#checkAll").click(**function**(){

$("input[type=checkbox][name=items]").attr("checked","checked");

});

//全不选

$("#checkNo").click(**function**(){

$("input[type=checkbox][name=items]").attr("checked",**null**);

});

//反选

$("#checkRev").click(**function**(){

$("input[type=checkbox][name=items]").each(**function**(){

**if**(**this**.checked){

$(**this**).attr("checked",**null**);

}**else**{

$(**this**).attr("checked","checked");

}

});

});

//全选全不选

$("#checkAll\_2").click(**function**(){

**if**(**this**.checked){

$("input[type=checkbox][name=items]").attr("checked","checked");

}**else**{

$("input[type=checkbox][name=items]").attr("checked",**null**);

}

});

</script>

</body>

</html>

## 2.6Jquery操作CSS样式

获取 class 和设置 class : class 是元素的一个属性, 所以获取 class 和设置 class 都可以使用 attr() 方法来完成.

**追加样式: addClass()**

移除样式: removeClass() --- 从匹配的元素中删除全部或指定的 class

**切换样式: toggleClass() --- 控制样式上的重复切换.如果类名存在则删除它, 如果类名不存在则添加它.**

判断是否含有某个样式: hasClass() --- 判断元素中是否含有某个 class, 如果有, 则返回 true; 否则返回 false

### 2.6.1操作CSS样式案例一

<script type=*"text/javascript"*>

// <input type="button" value="采用属性增加样式" id="b1"/>

$("#b1").click(**function**(){

$("#one").attr("class","mini");

});

//<input type="button" value="addclass" id="b2"/>

$("#b2").click(**function**(){

$("#one").addClass("mini");

});

//<input type="button" value="remove class" id="b3"/>

$("#b3").click(**function**(){

$("#one").removeClass("mini");

});

// <input type="button" value="切换样式" id="b4"/>

$("#b4").click(**function**(){

$("#one").toggleClass("mini");

});

//<input type="button" value="has class" id="b5"/>

$("#b5").click(**function**(){

alert($("#one").hasClass("mini"));

});

</script>

### 2.6.2焦点事件

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<html>

<head>

<title>19.html</title>

<meta http-equiv=*"keywords"* content=*"keyword1,keyword2,keyword3"*>

<meta http-equiv=*"description"* content=*"this is my page"*>

<meta http-equiv=*"content-type"* content=*"text/html; charset=UTF-8"*>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

</head>

<body>

<input type=*"text"* value=*"用户邮箱/手机号/用户名"* id=*"b1"*/><br/>

<input type=*"password"* value=*""* id=*"b2"*/><br/>

<input type=*"button"* value=*"登陆"* id=*"b2"*/><br/>

<script type=*"text/javascript"*>

//获得焦点

$("#b1").focus(**function**(){

//获取当前文本框的值

**var** curValue=$(**this**).val();

//如果当前文本框的值和默认值相等

**if**(curValue==**this**.defaultValue){

//设置当前文本的值为空

$(**this**).val("");

}

});

//失去焦点

$("#b1").blur(**function**(){

//获取当前文本框的值

**var** curValue=$(**this**).val();

//

**if**($.trim(curValue)==""){

//用户没有输入数据 设置为默认值即可

$(**this**).val(**this**.defaultValue);

}

});

</script>

</body>

</html>

### 2.6.3下拉列表和复选按钮

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<html>

<head>

<title>20.html</title>

<meta http-equiv=*"keywords"* content=*"keyword1,keyword2,keyword3"*>

<meta http-equiv=*"description"* content=*"this is my page"*>

<meta http-equiv=*"content-type"* content=*"text/html; charset=UTF-8"*>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

</head>

<body>

<input type=*"button"* value=*"使单选下拉框的2号被选中"*><br/>

<input type=*"button"* value=*"使多选下拉框中的2号和5号被选中"*><br/>

<input type=*"button"* value=*"使复选框的2和4被选中"*><br/>

<input type=*"button"* value=*"使单选框单选2被选中"*><br/>

<input type=*"button"* value=*"打印已经被选中的值"*><br/>

<select id=*"one"*>

<option>1号</option>

<option>2号</option>

<option>3号</option>

<option>4号</option>

</select>

<select id=*"many"* multiple=*"multiple"* style="height: *120px*;" >

<option selected=*"selected"*> 1号</option>

<option>2号</option>

<option>3号</option>

<option>4号</option>

<option>5号</option>

<option selected=*"selected"*>6号</option>

</select>

<br/>

<br/>

<input type=*"checkbox"* name=*"c"* value=*"check1"*/>复选1

<input type=*"checkbox"* name=*"c"* value=*"check2"*/>复选2

<input type=*"checkbox"* name=*"c"* value=*"check3"*/>复选3

<input type=*"checkbox"* name=*"c"* value=*"check4"*/>复选4

<br/>

<input type=*"radio"* name=*"r"* value=*"radio1"*/>单选1

<input type=*"radio"* name=*"r"* value=*"radio2"*/>单选2

<input type=*"radio"* name=*"r"* value=*"radio3"*/>单选3

<input type=*"radio"* name=*"r"* value=*"radio4"*/>单选4

<script type=*"text/javascript"*>

//<input type="button" value="使单选下拉框的2号被选中"><br/>

$("input[type=button]:eq(0)").click(**function**(){

$("#one").val("2号");

});

//<input type="button" value="使多选下拉框中的2号和5号被选中"><br/>

$("input[type=button]:eq(1)").click(**function**(){

$("#many").val(["2号","5号"]);

});

//<input type="button" value="使复选框的2和4被选中"><br/>

$("input[type=button]:eq(2)").click(**function**(){

$("input[type=checkbox][name=c]").val(["check2","check4"]);

});

//<input type="button" value="使单选框单选2被选中"><br/>

$("input[type=button]:eq(3)").click(**function**(){

$("input[type=radio][name=r]").val(["radio2"]);

});

//<input type="button" value="打印已经被选中的值"><br/>

$("input[type=button]:eq(4)").click(**function**(){

//打印下拉列表

$("select option:selected").each(**function**(){

alert($(**this**).text());

});

//打印 复选按钮 和单选按钮

$("input:checked").each(**function**(){

alert($(**this**).val());

});

});

</script>

</body>

</html>

### 2.6.4收缩展开效果

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

<html>

<head>

<title>21.html</title>

<meta http-equiv=*"keywords"* content=*"keyword1,keyword2,keyword3"*>

<meta http-equiv=*"description"* content=*"this is my page"*>

<meta http-equiv=*"content-type"* content=*"text/html; charset=UTF-8"*>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

</head>

<body>

<script type=*"text/javascript"*>

</script>

<div class=*"box"*>

<h1> 收缩展开效果</h1>

<div id=*"menu1"*>

1<br/>

2<br/>

3<br/>

4<br/>

5<br/>

</div>

</div>

<br/>

<div id=*"menu2"*>

<h1> 收缩展开效果2</h1>

<div class=*"text"*>

1<br/>

2<br/>

3<br/>

</div>

</div>

<div id=*"menu3"*>

<h1> 收缩展开效果3</h1>

<div class=*"text"*>

1<br/>

2<br/>

3<br/>

</div>

</div>

<script type=*"text/javascript"*>

$(":header").each(**function**(){

$(**this**).click(**function**(){

$(**this**).next().slideToggle(1000);

});

});

</script>

</body>

</html>

## 2.7使用Jquery操作Ajax实现异步调用

通过 HTTP 请求加载远程数据。

jQuery 底层 AJAX 实现。简单易用的高层实现见 $.get, $.post 等。$.ajax() 返回其创建的 XMLHttpRequest 对象。大多数情况下你无需直接操作该函数，除非你需要操作不常用的选项，以获得更多的灵活性。

最简单的情况下，$.ajax()可以不带任何参数直接使用。

**数据类型**

$.ajax()函数依赖服务器提供的信息来处理返回的数据。如果服务器报告说返回的数据是XML，那么返回的结果就可以用普通的XML方法或者jQuery的选择器来遍历。如果见得到其他类型，比如HTML，则数据就以文本形式来对待。

通过dataType选项还可以指定其他不同数据处理方式。除了单纯的XML，还可以指定 html、json、jsonp、script或者text。

其中，text和xml类型返回的数据不会经过处理。数据仅仅简单的将XMLHttpRequest的responseText或responseHTML属性传递给success回调函数，

'''注意'''，我们必须确保网页服务器报告的MIME类型与我们选择的dataType所匹配。比如说，XML的话，服务器端就必须声明 text/xml 或者 application/xml 来获得一致的结果。

### 2.7.1$.ajax

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<body>

<div id=*"div1"*></div>

<input type=*"button"* value=*"test"* id=*"btn"*>

<script type=*"text/javascript"*>

$("#btn").click(**function**(){

**var** param={username:"guigu",password:"hz123"};

$.ajax({

type:"POST",//请求方式

url:"TestAjaxJqueryServlet",//请求路径信息

data:param,

success:**function**(returnData){

$("#div1").html(returnData);

}

});

});

</script>

</body>

</html>

**package** com.guigu.servlet;

**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;

@WebServlet("/ajaxjquery/TestAjaxJqueryServlet")

**public** **class** TestAjaxJqueryServlet **extends** HttpServlet {

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

String p1=request.getParameter("username");

String p2=request.getParameter("password");

response.getWriter().println("用户名:"+p1+",密码:"+p2+",Ajax is so easy");

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

doGet(request, response);

}

}

### 2.7.2$.post

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<body>

<div id=*"div1"*></div>

<input type=*"button"* value=*"test"* id=*"btn"*>

<script type=*"text/javascript"*>

$("#btn").click(**function**(){

**var** param={username:"zhangsan",password:"zs123"};

$.post("TestAjaxJqueryServlet",param,**function**(returnData){

$("#div1").html(returnData);

},"text");

});

</script>

</body>

</html>

### 2.7.3$.get

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<body>

<div id=*"div1"*></div>

<input type=*"button"* value=*"test"* id=*"btn"*>

<script type=*"text/javascript"*>

$("#btn").click(**function**(){

//var param={username:"zhangsan",password:"zs123"};

$.get("TestAjaxJqueryServlet",**function**(returnData){

$("#div1").html(returnData);

},"text");

});

</script>

</body>

</html>

# 三、JSON

## 3.1JSON的简介

JSON（JavaScript Object Notation）一种简单的数据格式，比xml更轻巧。JSON是JavaScript**原生格式**，这意味着在JavaScript中处理JSON数据不需要任何特殊的API或工具包。

JSON的规则很简单：对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号）结束。每个“名称”后跟一个“:”（冒号）；“‘名称/值’对”之间使用“,”（逗号）分隔。

**规则如下:**

1） 并列的数据之间用逗号（“，”）分隔。名称1:值1,名称2:值2

2） 映射用冒号（“：”）表示。名称:值

3） 并列数据的集合（数组）用方括号(“[]”)表示。

[

{名称1:值,名称2:值2},

{名称1:值,名称2:值2}

]

4） 映射的集合（对象）用大括号（“{}”）表示。{名称1:值,名称2:值2}

5。元素值可具有的类型：string, number, object, array, true, false, null

**JSON 用冒号(而不是等号)来赋值。每一条赋值语句用逗号分开。整个对象用大括号封装起来。可用大括号分级嵌套数据。**

**对象描述中存储的数据可以是字符串，数字或者布尔值。对象描述也可存储函数，那就是对象的方法。**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<script type=*"text/javascript"*>

**var** jsonObject ={

username:"guigu",

password:"hzgg1234",

age:45,

tel:"15064067894"

};

**function** writeFunction(){

document.getElementById("jname").innerHTML=jsonObject.username;

document.getElementById("jpwd").innerHTML=jsonObject.password;

document.getElementById("jage").innerHTML=jsonObject.age;

document.getElementById("jtel").innerHTML=jsonObject.tel;

}

</script>

<p id=*"jname"*>a</p>

<p id=*"jpwd"*>b</p>

<p id=*"jage"*>c</p>

<p id=*"jtel"*>d</p>

<input type=*"button"* onclick="writeFunction()" value=*"ok"*>

</body>

</html>

1. **JSON和XML有什么区别？**

**<person>**

**<id>1</id>**

**<name>guigu</name>**

**<age>34</age>**

**</person>**

**var personJson={id:1,name:guigu,age:34};**

**相同之处:**

JSON的纯文本、JSON具有自我描述性、JSON具有层次结构、JSON可以在js中进行解析

**不同之处:**

JSON是没有结束标签、更短、读写速度快、使用js的内置方法进行相关解析。

**2）为什么使用JSON**

对于程序来说比如Ajax的调用，使用json 比xml更加容易，并且更加方便解析和操作。、

**<persons>**

**<person>**

**<id>1</id>**

**<name>guigu</name>**

**<age>34</age>**

**</person>**

**<person>**

**<id>2</id>**

**<name>zhangsan </name>**

**<age>35</age>**

**</person>**

**<person>**

**<id>3</id>**

**<name>lisi </name>**

**<age>36</age>**

**</person>**

**</persons>**

**var personJson=[{id:1,name:guigu,age:34} , {id:2,name:zhangsan,age:35}];**

## 3.2JSON的使用

**案例一**

**package** com.guigu.json;

**import** java.util.ArrayList;

**import** java.util.List;

**import** net.sf.json.JSONArray;

**import** net.sf.json.JsonConfig;

/\*\*

\* <p>project\_name:JavaAjaxJquery</p>

\* <p>package\_name:com.guigu.json.JsonTest</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年1月9日下午3:38:39 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** JsonTest {

**public** **static** **void** main(String[] args) {

//模拟从数据库查询出指定的数据 List<Province>

List<Province> list =**new** ArrayList<>();

Province p1 =**new** Province(1,"浙江省");

Province p2 =**new** Province(2,"山东省");

Province p3= **new** Province(3,"江苏省");

list.add(p1);

list.add(p2);

list.add(p3);

//把查询到的数据传递到页面中显示

//假如通过ajax查询 ajax需要的返回值类型是json轻量级数据

// JSONArray jsonArray =JSONArray.fromObject(list);

//假如Province 有几百个属性信息 想过滤属性进行传递

JsonConfig config =**new** JsonConfig();

config.setExcludes(**new** String[] {"pid"});

JSONArray jsonArray =JSONArray.*fromObject*(list,config);

System.***out***.println(jsonArray.toString());

}

}

**案例二**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"* src=*"../js/jquery-1.8.2.min.js"*></script>

<script type=*"text/javascript"*>

$(**function**(){

$("#button1").click(**function**(){

$.ajax({

type:"POST",

url:"TestJqueryJSONServlet",

data:"id="+document.getElementById("userid"),

dataType:"json",

success:**function**(res){

alert("id:"+res[0].id+",name"+res[0].name);

}

});

});

});

</script>

<body>

<table>

<tr>

<td>id:</td>

<td>

<input type=*"text"* id=*"userid"*>

<input type=*"button"* value=*"ok"* id=*"button1"* name=*"button1"*>

</td>

</tr>

</table>

</body>

</html>

**package** com.guigu.servlet;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** com.guigu.json.User;

**import** net.sf.json.JSONArray;

@WebServlet("/json/TestJqueryJSONServlet")

**public** **class** TestJqueryJSONServlet **extends** HttpServlet {

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

response.setContentType("text/json;charset=UTF-8");

PrintWriter pw =response.getWriter();

List<User> list =**new** ArrayList<>();

User p1 =**new** User(1,"浙江省");

User p2 =**new** User(2,"山东省");

User p3= **new** User(3,"江苏省");

list.add(p1);

list.add(p2);

list.add(p3);

JSONArray jsonArray =JSONArray.*fromObject*(list);

pw.println(jsonArray.toString());

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

doGet(request, response);

}

}