JavaWEB-Ajax

讲师：佟刚

新浪微博：尚硅谷-佟刚

什么是Ajax

- Ajax的技术的产生
  - Ajax被认为是(Asynchronous 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。

Ajax应用实例 - Google Suggest

Ajax应用实例 - Google Maps

什么是Ajax

- Ajax：一种**不用刷新整个页面便可与服务器通讯的办法**

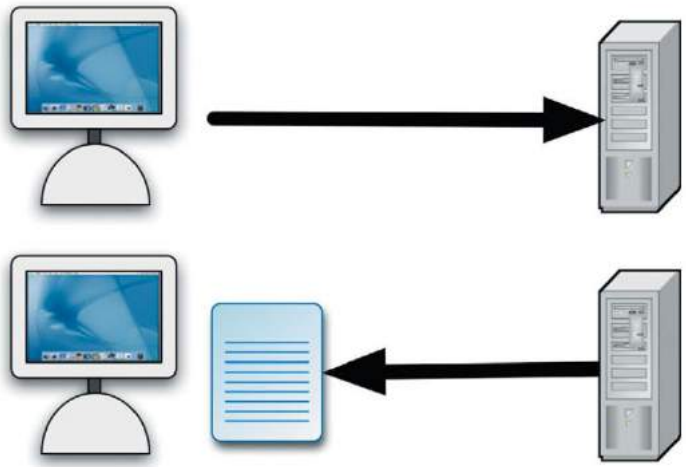


图1 Web的传统模型。客户端向服务器发送一个请求，服务器返回整个页面，如此反复

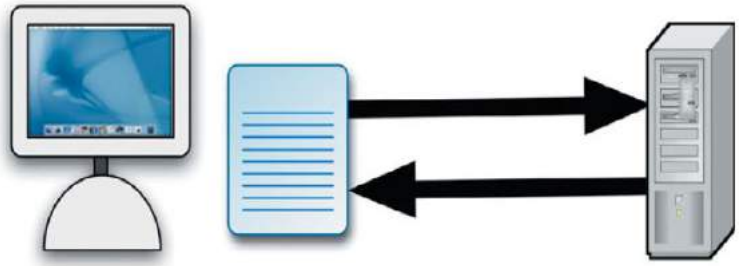
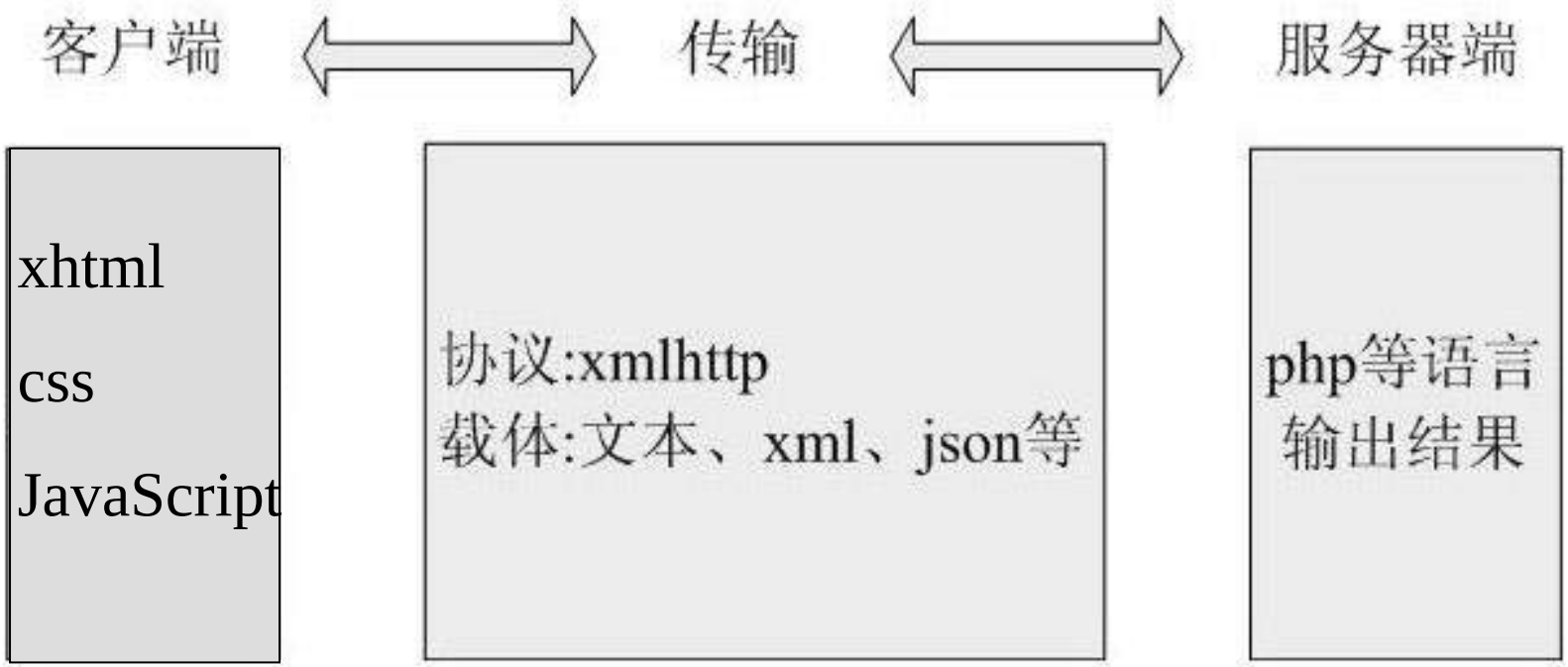


图2 在Ajax模型中，**数据在客户端与服务器之间独立传输。服务器不再返回整个页面**

什么是Ajax

- 不用刷新整个页面便可与服务器通讯的办法：
  - Flash
  - Java applet
  - 框架：如果使用一组框架构造了一个网页，可以只更新其中一个框架，而不必惊动整个页面
  - 隐藏的iframe
  - **XMLHttpRequest**：该对象是对 JavaScript 的一个扩展，可使网页与服务器进行通信。是创建 Ajax 应用的最佳选择。**实际上通常把 Ajax 当成 XMLHttpRequest 对象的代名词**

Ajax的工作原理图



Ajax工具包

- Ajax**并不是一项新技术**，它实际上是几种技术，每种技术各尽其职，以一种全新的方式聚合在一起
  - **服务器端语言**：服务器需要具备向浏览器发送特定信息的能力。**Ajax与服务器端语言无关**。
  - **XML**（eXtensible Markup Language，可扩展标记语言）是一种描述数据的格式。**Aajx 程序需要某种格式化的格式来在服务器和客户端之间传递信息，XML 是其中的一种选择**
  - **XHTML**（eXtended Hypertext Markup Language,使用扩展超媒体标记语言）和 **CSS**（Cascading Style Sheet,级联样式单）**标准化呈现；**
  - **DOM**（Document Object Model,文档对象模型）**实现动态显示和交互；**
  - 使用XMLHTTP组件**XMLHttpRequest对象**进行**异步数据读取**
  - 使用**JavaScript绑定和处理所有数据**

Ajax的缺陷

- AJAX不是完美的技术。使用AJAX，它的一些缺陷不得不权衡一下：
  - 由 Javascript 和 AJAX 引擎导致的浏览器的**兼容**
  - 页面局部刷新，导致**后退等功能失效**。
  - 对流媒体的支持没有FLASH、Java Applet好。
  - 一些手持设备（如手机、PDA等）支持性差。

XMLHttpRequest的概述

- 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。

创建XMLHttpRequest对象

- 为了每次写Ajax的时候都节省一点时间，可以把对象检测的内容打包成一个可复用的函数：

```
function getHTTPObject(){
    var xhr = false;
    if(window.XMLHttpRequest){
        xhr = new XMLHttpRequest();
    }else if(window.ActiveXObject){
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
    }
    return xhr;
}
```

说明：对window.XMLHttpRequest的调用会返回一个对象或null，if语句会把调用返回的结果看作是true或false（如果返回对象则为true，返回null则为false）。如果XMLHttpRequest对象存在，则把 xhr 的值设为该对象的新实例。如果不存在，就去检测 ActiveObject 的实例是否存在，如果答案是肯定的，则把微软 XMLHTTP 的新实例赋给 xhr

XMLHttpRequest的方法

方法	描述
abort()	停止当前请求
getAllResponseHeaders()	把HTTP请求的所有响应首部作为键/值对返回
getResponseHeader(“header”)	返回指定首部的串值
open(“method”, “url”)	建立对服务器的调用。Method参数可以是GET、POST或PUT.url参数可以是相对URL或绝对URL。
send(content)	向服务器发送请求
setRequestHeader(“header”, “value”)	把指定首部设置为所提供的值。在设置任何首部之前必须先调用open()

XMLHttpRequest的属性

属性	描述
onreadystatechange	每个状态改变是都会触发这个事件处理器，通常会调用一个javaScript函数
readyState	请求的状态，有5个可取值：0=未初始化、1=正在加载、2=已经加载、3=交互中、4=完成。
responseText	服务器的响应，表示为一个串。
responseXML	服务器的响应，表示为XML。这个对象可以解析为DOM对象。
status	服务器的HTTP状态码（200对应OK、404对应NotFount、等）
statusText	HTTP状态码的相应文本（OK或NotFount等）

发送请求

- 利用XMLHttpRequest 实例与服务器进行通信包含以下3个关键部分：
  - onreadystatechange 事件处理函数
  - open 方法
  - send 方法

发送请求

- 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**

发送请求

- 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，就会带来不好的结果。**在此将时间戳追加到URL的最后，就能确保URL的唯一性，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。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。

发送请求

- 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.
  - 完整的 Ajax 的 GET 请求示例：

```
var request = getHTTPObject();
if(request){
    request.onreadystatechange = doSomeThing;
    request.open("GET","file.txt",true);
    request.send(null);
}
```

发送请求

- setRequestHeader(header,value)
  - 当浏览器向服务器请求页面时，它会伴随这个请求发送一组首部信息。这些首部信息是一系列描述请求的元数据(metadata)。**首部信息用来声明一个请求是 GET 还是 POST**。
  - Ajax 请求中，发送首部信息的工作可以由 setRequestHeader该完成
  - 参数header： 首部的名字; 参数value：首部的值。
  - **如果用 POST 请求向服务器发送数据，需要将 “Content-type” 的首部设置为 “application/x-www-form-urlencoded”.它会告知服务器正在发送数据，并且数据已经符合URL编码了。**
  - **该方法必须在open()之后才能调用**
  - 完整的 Ajax 的 POST 请求示例：

```
var url = "../jsp/forumServlet";
var nameValue = trim(document.forumAddForm.name.value);
xhr.open("POST",url);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.send("method=name_isExist" + "&name=" + nameValue);
```

接收相应

- 用 XMLHttpRequest 的方法可向服务器发送请求。在 Ajax 处理过程中，XMLHttpRequest 的如下属性可被服务器更改：
  - readyState
  - status
  - responseText
  - responseXML

接收相应

- 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**
  - **每次 readyState 值的改变，都会触发 readystatechange 事件**。如果把 onreadystatechange 事件处理函数赋给一个函数，那么每次 readyState 值的改变都会引发该函数的执行。
  - readyState 值的变化会因浏览器的不同而有所差异。**但是，当请求结束的时候，每个浏览器都会把 readyState 的值统一设为 4**

接收相应

- 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)
  - **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**

接收相应

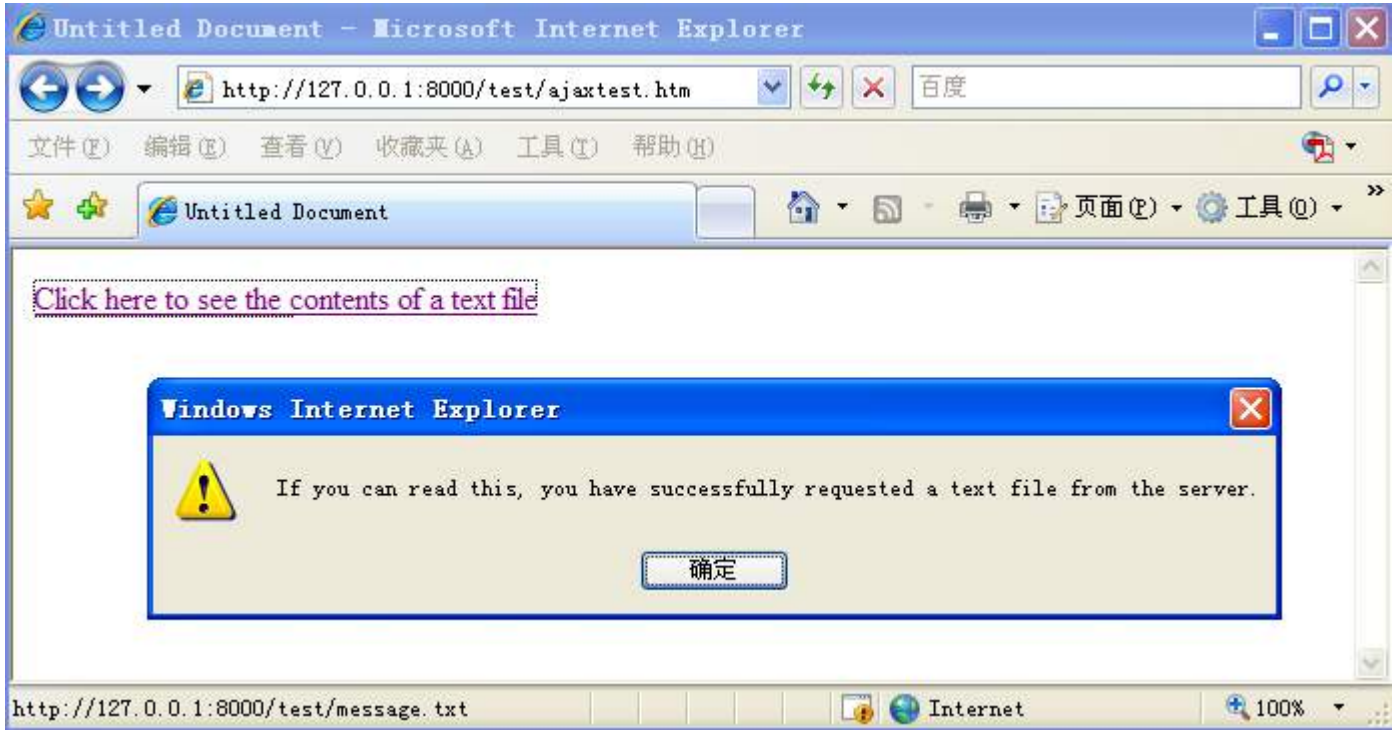
- responseText
  - XMLHttpRequest 的 responseText 属性**包含了从服务器发送的数据**。它是一个HTML,XML或普通文本，这取决于服务器发送的内容。
  - 当 readyState 属性值变成 4 时, responseText 属性才可用，表明 Ajax 请求已经结束。

```
function doSomeThing(){
    if(request.readyState == 4){
        if(request.status == 200 || request.status == 304){
            alert(request.responseText);
        }
    }
}
```

接收相应

- 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**

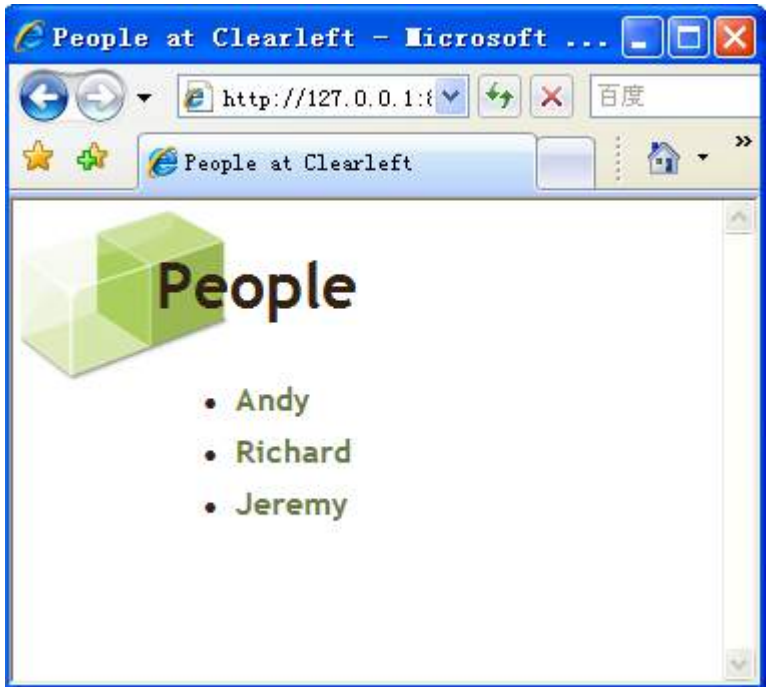
汇总


Untitled Document - Microsoft Internet Explorer
http://127.0.0.1:8000/test/ajaxtest.htm
百度
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
Untitled Document
页面(P) 工具(O)
Click here to see the contents of a text file
Windows Internet Explorer
If you can read this, you have successfully requested a text file from the server.
确定
http://127.0.0.1:8000/test/message.txt
Internet
100%

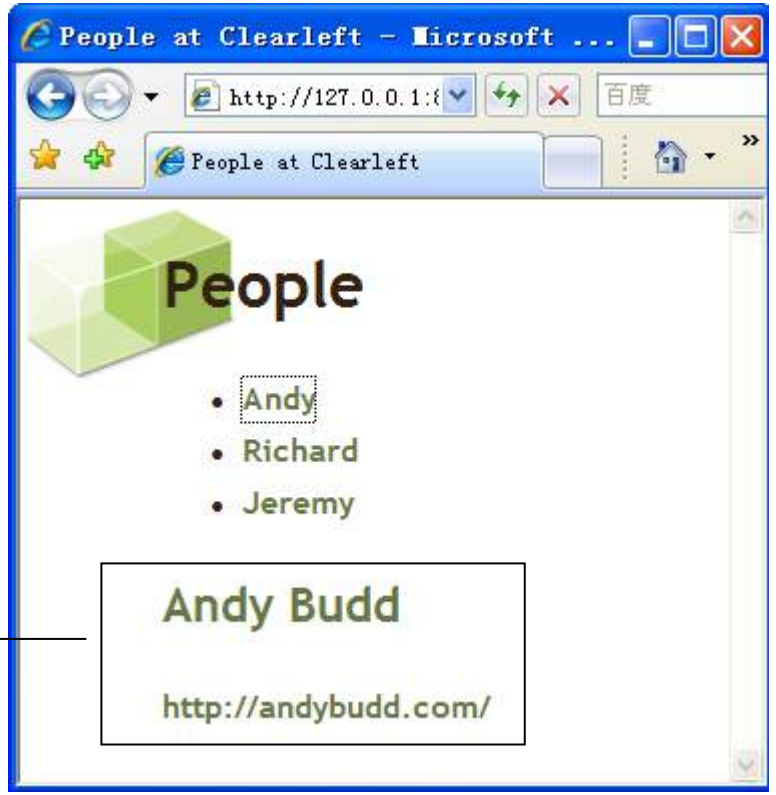
数据格式提要

- 在服务器端 AJAX 是一门与语言无关的技术。在业务逻辑层使用何种服务器端语言都可以。
- 从服务器端接收数据的时候，那些数据必须以浏览器能够理解的格式来发送。服务器端的编程语言只能以如下 3 种格式返回数据：
  - XML
  - JSON
  - HTML

任务



网页中的人员列表



点击连接就会显示个人的信息

```
<div id="details">
  <h2>
      <a href="mailto:andy@clearleft.com">Andy Budd</a>
  </h2>
  <a href="http://andybudd.com/">http://andybudd.com/</a>
</div>
```

XML

- 优点：
  - XML 是一种通用的数据格式。
  - 不必把数据强加到已定义好的格式中，而是要为数据自定义合适的标记。
  - 利用 DOM 可以完全掌控文档。
- 缺点：
  - 如果文档来自于服务器，就必须得保证文档含有正确的首部信息。若文档类型不正确，那么 responseXML 的值将是空的。
  - 当浏览器接收到长的 XML 文件后， DOM 解析可能会很复杂

JSON

- 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。

JSON 示例

```
var user =
{
    "username":"andy",
    "age":20,
    "info": {"tel":"123456","cellphone":"98765"},
    "address":
        [
            {"city":"beijing","postcode":"222333"},
            {"city":"newyork","postcode":"555666"}
        ]
}

alert(user.username);
alert(user.age);
alert(user.info.cellphone);
alert(user.address[0].city);
alert(user.address[0].postcode);
```

- 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。

解析 JSON

- JSON 只是一种文本字符串。它被存储在 responseText 属性中
- 为了读取存储在 responseText 属性中的 JSON 数据，需要根据 JavaScript 的 eval 语句。**函数 eval 会把一个字符串当作它的参数。然后这个字符串会被当作 JavaScript 代码来执行。因为 JSON 的字符串就是由 JavaScript 代码构成的，所以它本身是可执行的**
- 代码实例：

```
var jsonResponse = xhr.responseText;
var personObject = eval("(" + jsonResponse + ")");
var name = personObject.person.name;
var website = personObject.person.website;
var email = personObject.person.email;
```

- JSON提供了json.js包，下载http://www.json.org/json.js后，使用parseJSON()方法将字符串解析成 JS 对象

```
var jsonResponse = xhr.responseText;
var personObject = jsonResponse.parseJSON();
var name = personObject.person.name;
var website = personObject.person.website;
var email = personObject.person.email;
```

JSON 小结

- 优点：
  - 作为一种数据传输格式，JSON 与 XML 很相似，但是它更加灵巧。
  - JSON 不需要从服务器端发送含有特定内容类型的首部信息。
- 缺点：
  - 语法过于严谨
  - 代码不易读
  - eval 函数存在风险

解析 HTML

- HTML 由一些普通文本组成。如果服务器通过 XMLHttpRequest 发送 HTML， 文本将存储在 responseText 属性中。
- 不必从 responseText 属性中读取数据。它已经是希望的格式，可以直接将它插入到页面中。
- 插入 HTML 代码最简单的方法是更新这个元素的 innerHTML 属性。

HTML 小结

- 优点：
  - 从服务器端发送的 HTML 代码在浏览器端不需要用 JavaScript 进行解析。
  - HTML 的可读性好。
  - HTML 代码块与 innerHTML 属性搭配，效率高。
- 缺点：
  - 若需要通过 AJAX 更新一篇文档的多个部分，HTML 不合适
  - innerHTML 并非 DOM 标准。

对比小结

- 若应用程序**不需要与其他应用程序共享数据的时候**, 使用 HTML 片段来返回数据时最简单的
- **如果数据需要重用**, JSON 文件是个不错的选择, 其在性能和文件大小方面有优势
- **当远程应用程序未知时**, XML 文档是首选, 因为 XML 是 web 服务领域的“世界语”

jQuery 中的 Ajax

- jQuery 对 Ajax 操作进行了封装, 在 jQuery 中最底层的方法时 $.ajax(), 第二层是 **load(), $.get() 和 $.post(),** 第三层是 $.getScript() 和 $.getJSON()

load() 方法

- load() 方法是 jQuery 中最为简单和常用的 Ajax 方法, **能载入远程的 HTML 代码并插入到 DOM 中**. 它的结构是: **load(url[, data][,callback])**

参数名称	类型	说明
url	String	请求 HTML 页面的 URL 地址
data(可选)	Object	发送到服务器的 key/value 数据
callback(可选)	Function	请求完成时的回调函数, 无论请求成功或失败

- 程序员只需要使用 jQuery 选择器为 HTML 片段指定目标位置, 然后将要加载的文件的 url 做为参数传递给 load() 方法即可

load() 方法 ---- 细节

- 如果只需要加载目标 HTML 页面内的某些元素, 则可以通过 load() 方法的 URL 参数来达到目的. **通过 URL 参数指定选择符**, 就可以方便的从加载过来的 HTML 文档中选出所需要的内容. load() 方法的 URL 参数的语法结构为 **“url selector”**(注意: url 和 选择器之间有一个空格)
- 传递方式: load() 方法的传递参数根据参数 data 来自动自定. **如果没有参数传递, 采用 GET 方式传递, 否则采用 POST 方式**
- 对于必须在加载完才能继续的操作, load() 方法提供了回调函数, 该函数有三个参数: **代表请求返回内容的 data; 代表请求状态的 textStatus 对象和 XMLHttpRequest 对象**

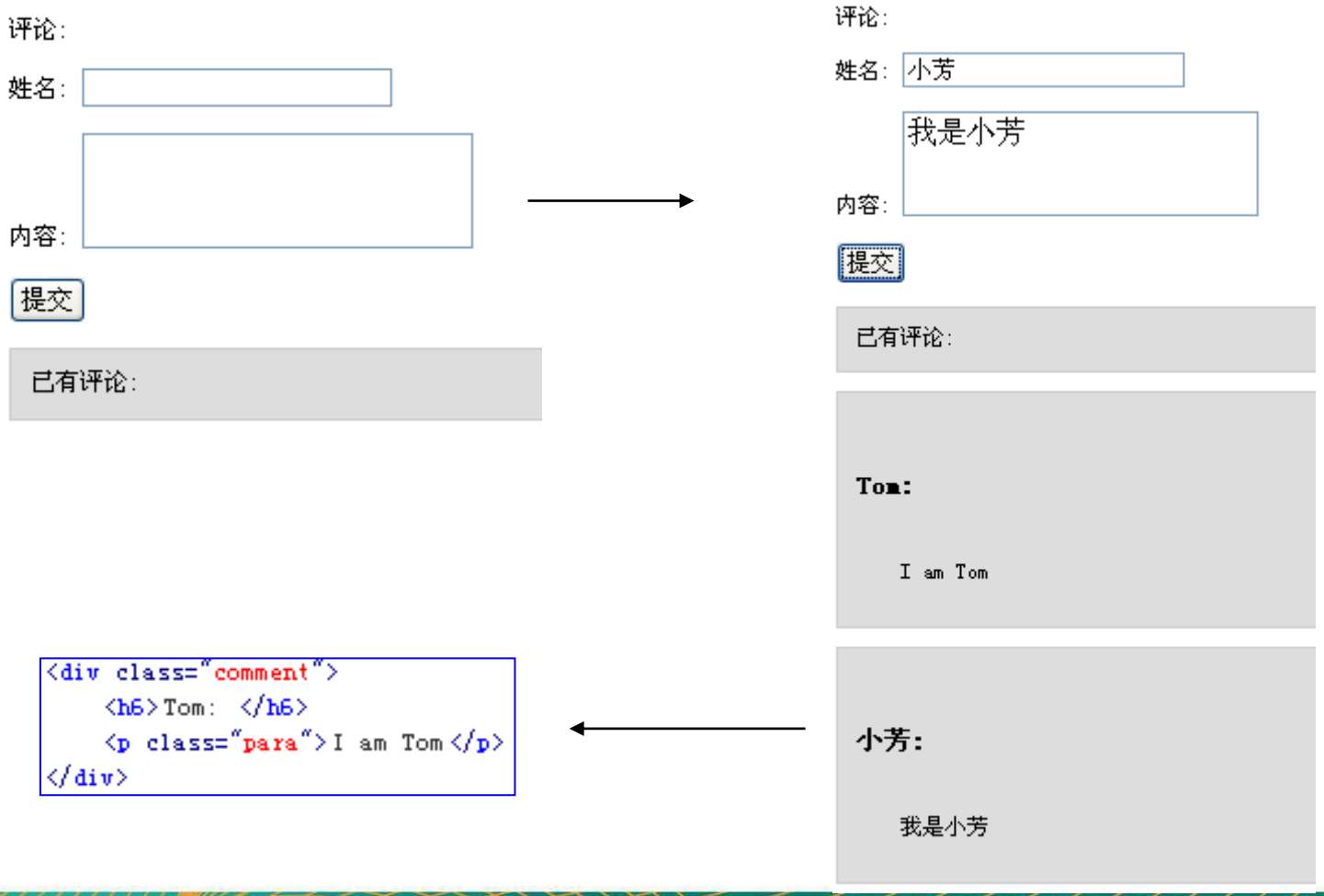
$.get() (或$.post()) 方法

- $.get() 方法使用 GET 方式来进行异步请求. 它的结构是: $.get(url[, data][, callback][, type]);

参数名称	类型	说明
url	String	请求 HTML 页面的 URL 地址
data(可选)	Object	发送到服务器的 key/value 数据会作为 QueryString 附加到请求 URL 中
callback(可选)	Function	载入成功时回调函数(只有当 Response 的返回状态时 success 才调用该方法)自动将请求结果和状态传递给该方法
type(可选)	String	服务器返回内容的格式, 包括 xml, html, script, json, text 和 _default

- $.get() 方法的回调函数只有两个参数: data 代表返回的内容, 可以是 XML 文档, JSON 文件, HTML 片段等; textstatus 代表请求状态, 其值可能为: succuss, error, notmodify, timeout 4 种.
- $.get() 和 $.post() 方法时 jQuery 中的全局函数, 而 find() 等方法都是对 jQuery 对象进行操作的方法

$.get() (或$.post()) 方法应用



序列化元素

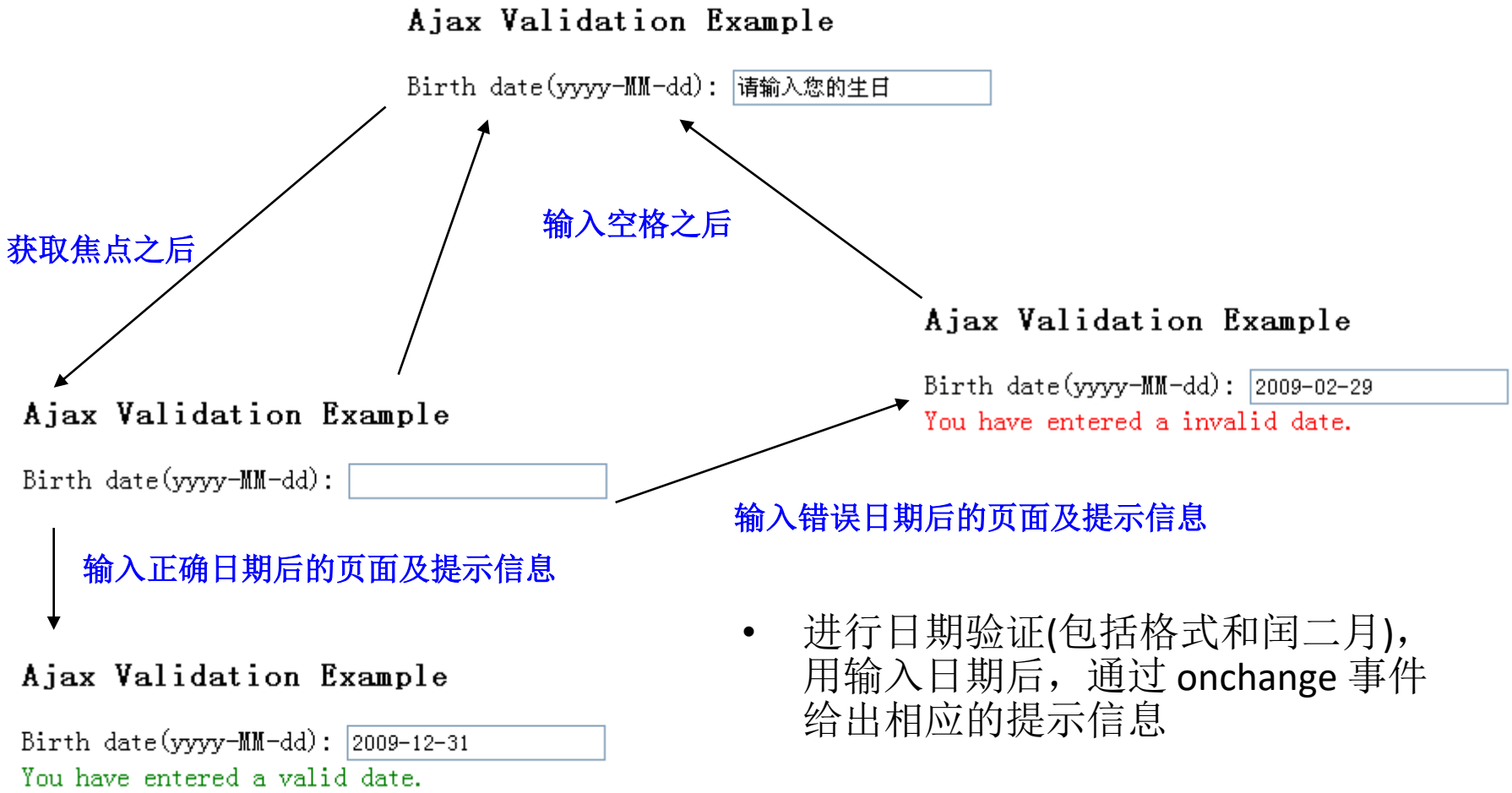
- jQuery 为准备 “发送到服务器的 key/value 数据” 提供了一个简化的方法: serialize(). 该方法作用于一个 jQuery 对象, 能将 DOM 元素内容序列化为字符串, 用于 Ajax 请求.

```
$("#send").click(function(){
    $.get("/ajax/messageservlet", {
        username: $("#username").val(),
        content: $("#content").val()
    }, function(data, textStatus){
        alert(data);
        var username = data.username;
        var content = data.content;
```

```
$("#send").click(function(){
    $.get("/ajax/messageservlet", $("#form1").serialize(),
        function(data, textStatus){
            var username = data.username;
            var content = data.content;
```

- 使用 serialize() 方法可以自动完成对参数的 url 编码
- 因为该方法作用于 jQuery 对象, 所以不光只要表单能使用, 其它选择器选取的元素也能使用它.

验证日期



- 进行日期验证(包括格式和闰二月)，用输入日期后，通过 onchange 事件给出相应的提示信息

验证日期

- 日期验证的关键：
  - Java 正则表达式类： java.util.regex.Pattern
  - JAVA日期验证 正则表达式,包括润二月：

```
Pattern p = Pattern.compile("^((\\d{2}(([02468][048])|([13579][26]))[\\-\\-\\s]?((((0?" +
        "[13578])|(1[02]))[\\-\\-\\s]?((0?[1-9])|([1-2][0-9])|(3[01])))" +
        "|(((0?[469])|(11))[\\-\\-\\s]?((0?[1-9])|([1-2][0-9])|(30)))|" +
        "(0?2[\\-\\-\\s]?((0?[1-9])|([1-2][0-9])))))|(\\d{2}(([02468][12" +
        "35679])|([13579][01345789]))[\\-\\-\\s]?((((0?[13578])|(1[02]))" +
        "[\\-\\-\\s]?((0?[1-9])|([1-2][0-9])|(3[01])))|(((0?[469])|(11))" +
        "[\\-\\-\\s]?((0?[1-9])|([1-2][0-9])|(30)))|(0?2[\\-\\-\\s]?((0?[" +
        "1-9])|(1[0-9])|(2[0-8]))))))");
```

验证 Email

- 练习：把 email 作为注册用户名。需验证其格式的正确和在数据库中的唯一性。
  - Java email验证的正则表达式：

```
String regex = "\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*";
```

  - 正确信息：You have entered a valid email.
  - 格式错误信息：You have entered an invalid email.
  - Email 重复信息： Your email has been saved.

动态加载列表框


My JSP 'employee.jsp' starting page - Microsoft Int...
http://localhost:8
百度
键入以搜索网站(Ctrl+E)
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
My JSP 'employee.jsp' starting page
城市: South San Francisco 部门: Shipping 员工名: Davies
ID: 142
LastName: Davies
FirstName: Curtis
Email: CDAVIES
完成
Internet
100%


尚硅谷
www.atguigu.com