

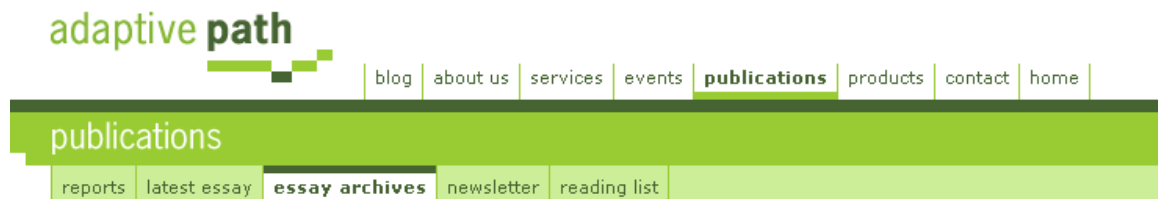
Advanced web technology

# 高级Web技术

*RIA技术 — Ajax*

# Ajax简介

- **Ajax: A New Approach to Web Applications**
  - 最早提出者 **Jesse James Garrett**
  - <http://www.adaptivepath.com/publications/essays/archives/000385.php>



## Ajax: A New Approach to Web Applications



*Jesse James Garrett is President and a founder of Adaptive Path. He is the author of the widely-referenced book [The Elements of User Experience](#). Jesse's other*

### by [Jesse James Garrett](#)

February 18, 2005

If anything about current interaction design can be called "glamorous," it's creating Web applications. After all, when was the last time you heard someone rave about the interaction design of a product that wasn't on the Web? (Okay, besides the iPod.) All the cool, innovative new projects are online.

Despite this, Web interaction designers can't help but feel a little envious of our colleagues who create desktop software. Desktop applications have a richness and responsiveness that has seemed out of reach on the Web. The same simplicity that enabled the Web's rapid proliferation also creates a gap between the experiences we can provide and the experiences users can get from a desktop application.

That gap is closing. Take a look at [Google Suggest](#). Watch the way the suggested terms update as you type, almost instantly. Now look at [Google Maps](#).

### Recent Essays

[Sarah Nelson Interviews Scott Berkun at MX San Francisco](#)

February 22, 2007

[Nine Adaptive Pathers Share Their Resolutions for 2007](#)

January 4, 2007

[Interview with Tim Brown, CEO of IDEO](#)

January 3, 2007

[Tagging vs. Cataloging: What It's All About](#)

November 30, 2006

[Organizing Your Global Corporate Intranet](#)

November 10, 2006

[Essay Archives](#) »

# AJAX的优势

---

- **直观自然的用户交互**
  - 许多时候无需点击鼠标，鼠标的移动等即可触发事件
- **页面的局部刷新**
  - 不需刷新页面就可改变页面内容，减少用户等待时间
- **数据驱动（而不是页面驱动）**
  - 按需获取数据，每次只从服务器端获取需要的数据
  - 读取外部数据，进行数据处理整合
- **异步通讯取代了同步的“请求/响应模型”**
  - 异步与服务器进行交互，在交互过程中用户无需等待，仍可继续操作
  - 显示和数据获取分离

# Ajax简介

---

- Web2.0的核心技术
- Ajax是Asynchronous JavaScript and XML的缩写
- 有确定需要从服务器读取新数据时再由Ajax引擎代为向服务器提交请求。
  - 使用XHTML和CSS标准化呈现
  - 使用DOM实现动态显示和交互
  - 使用XML和XSLT进行数据交换与处理
  - 使用XMLHttpRequest进行异步数据读取，向XMLHttpRequest注册一个回调函数
  - 最后用JavaScript整合以上技术

# Ajax简介

---

## ■ web 标准的三个关键组成

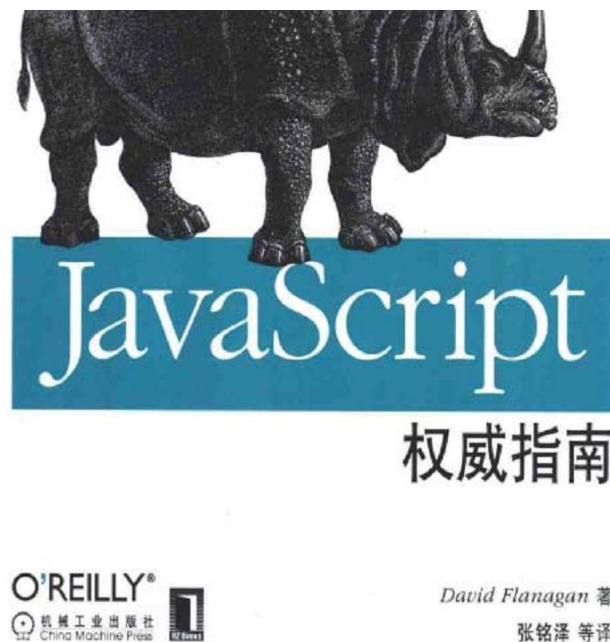
HTML→XHTML1.0→ (Transitional)→ (Strict)→XHTML2.0



# Ajax简介

---

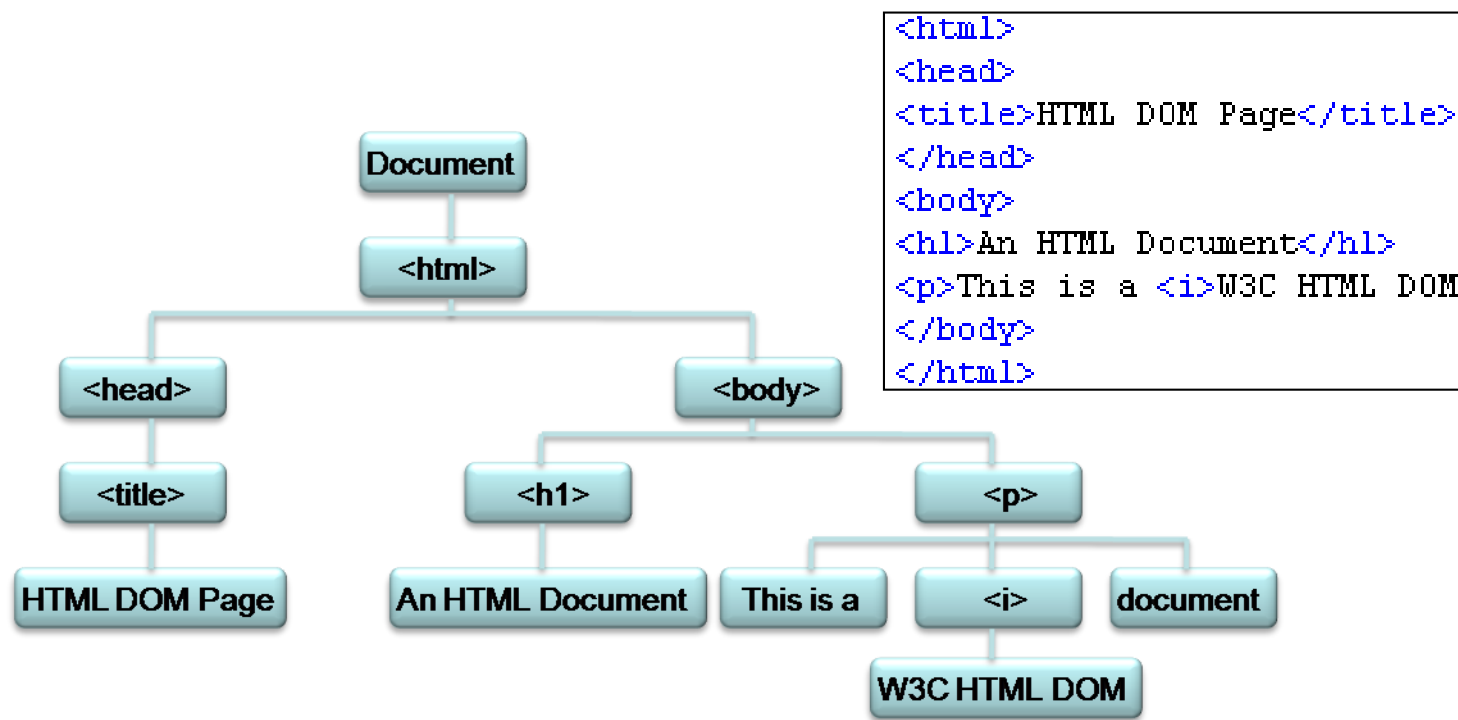
- Javascript
  - 弱类型的脚本语言
  - 页面发生事件时可以调用
  - 整个AJAX操作的胶水语言



# Ajax简介

## ■ DOM

- 文档对象模型（Document Object Model,简称DOM）
- 将XML和HTML文档结构视为对象和属性树，提供操作API



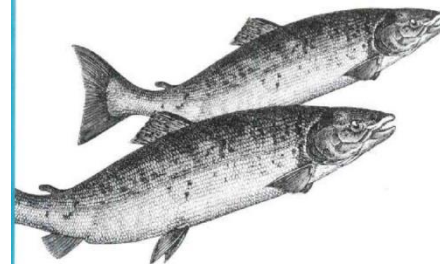
# Ajax简介

## ■ CSS

- 级联样式单 (Cascading Style Sheet, 简称CSS)
- 支持表现和内容的清晰分离; 可以通过Javascript编程修改

```
TITLE {display: block; font-size: 24pt; font-weight: bold;
       text-align: center; text-decoration: underline}
AUTHOR {display: block; font-size: 18pt; font-weight: bold;
        text-align: center}
SECTION {display: block; font-size: 16pt; font-weight: bold;
         text-align: center; font-style: italic}
P {display: block; margin-top: 10}
```

CSS权威指南





# Ajax简介

---

## ■ XMLHttpRequest

- 该对象是对 JavaScript 的一个扩展，可使浏览器与服务器通过标准的 HTTP GET/POST 进行通信。
- 在后台工作，执行和服务端异步通讯，而不打断用户操作
- 无需安装插件，在 Chrome, Firefox, Safari 和 Opera 上都有很好支持

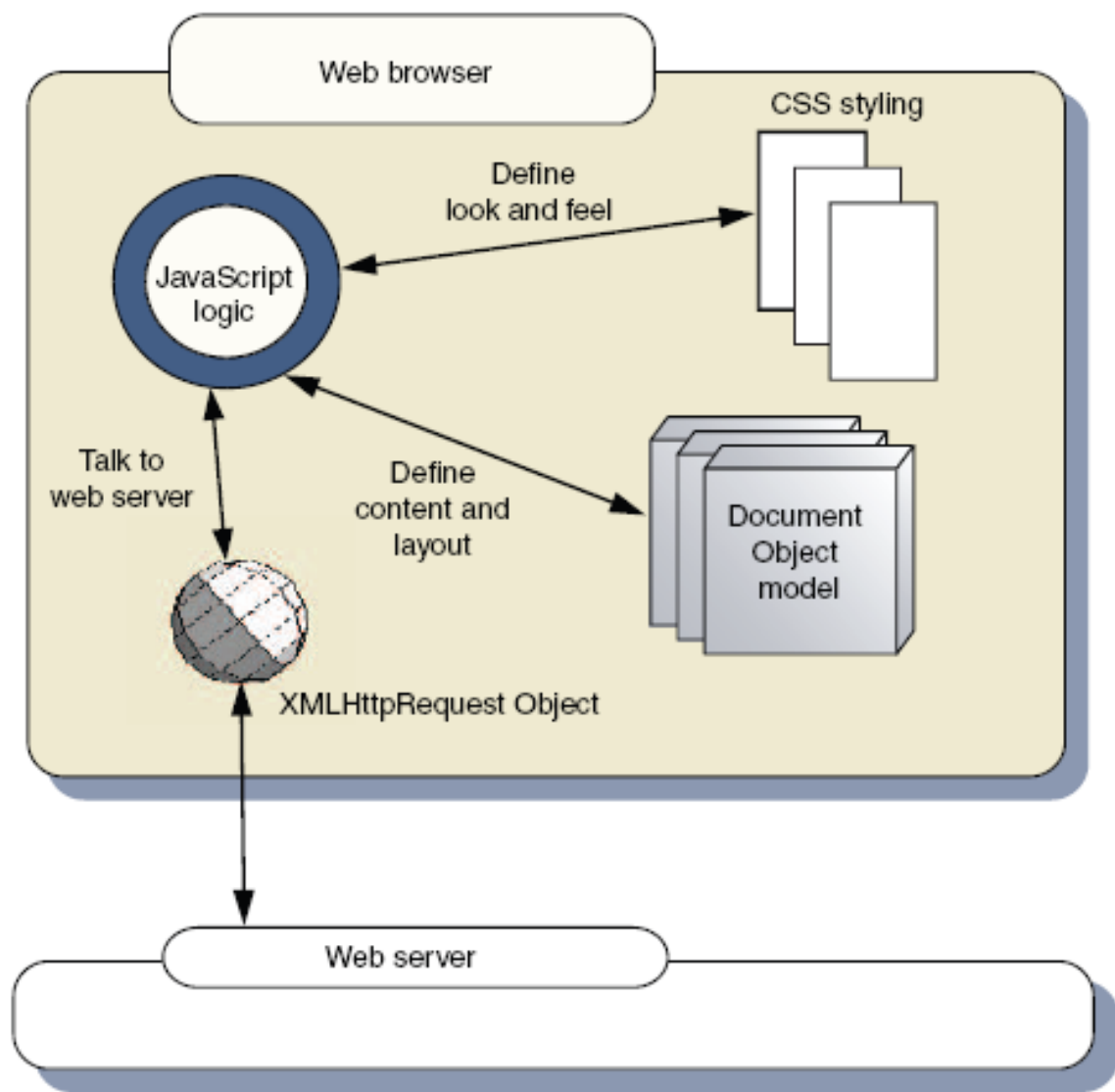
```
var xmlHttp;  
function createXMLHttpRequest() {  
    if (window.ActiveXObject) {  
        xmlHttp = new  
ActiveXObject("Microsoft.XMLHTTP");  
    } else if (window.XMLHttpRequest) {  
        xmlHttp = new XMLHttpRequest();  
    }  
}
```

# Ajax简介

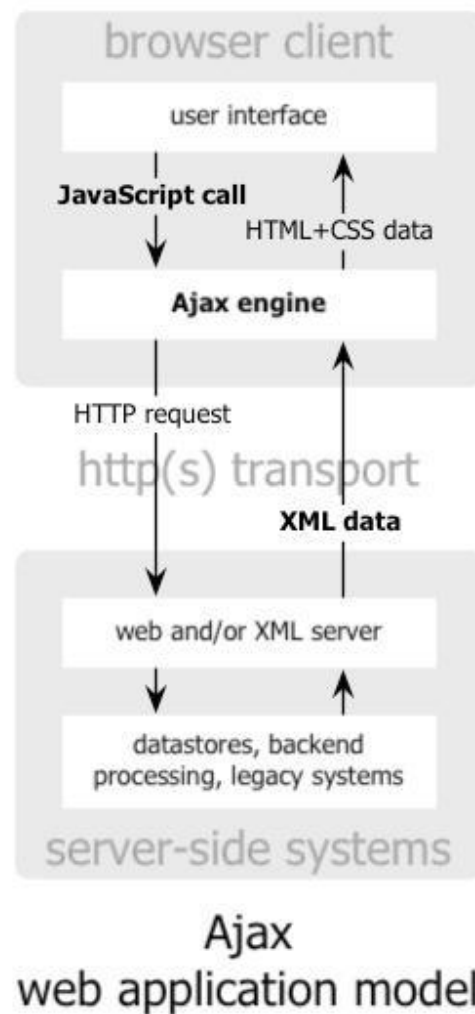
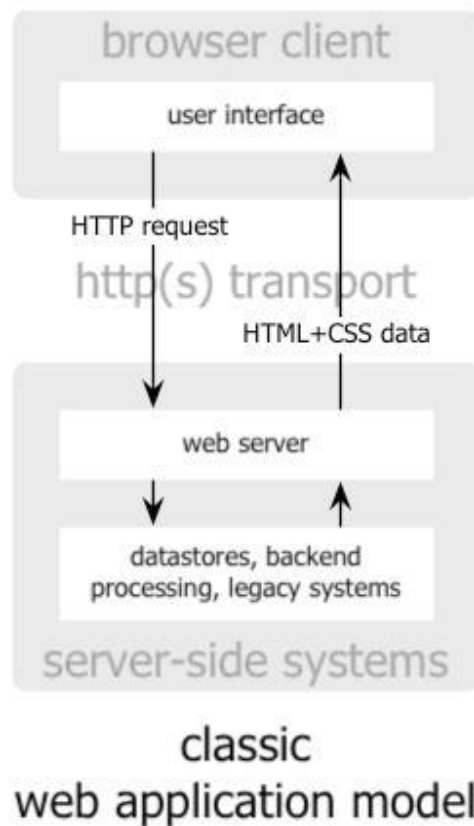
- JavaScript编写的Ajax引擎允许用户与页面的交互和页面与服务器的交互异步的进行



# Ajax简介



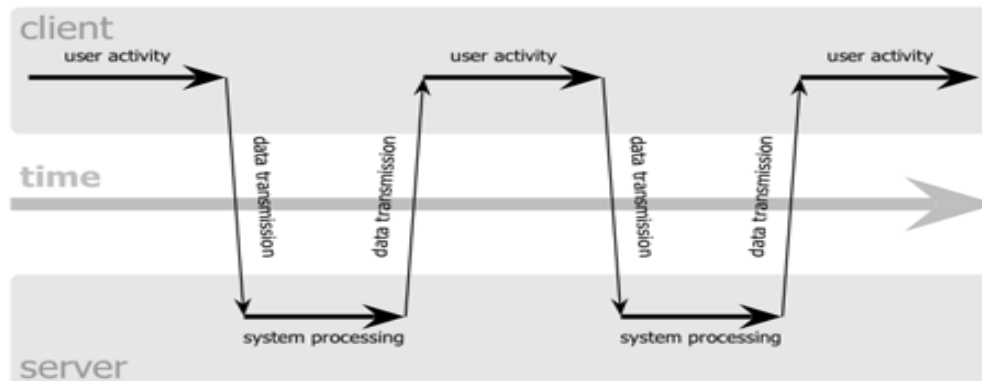
# Ajax简介



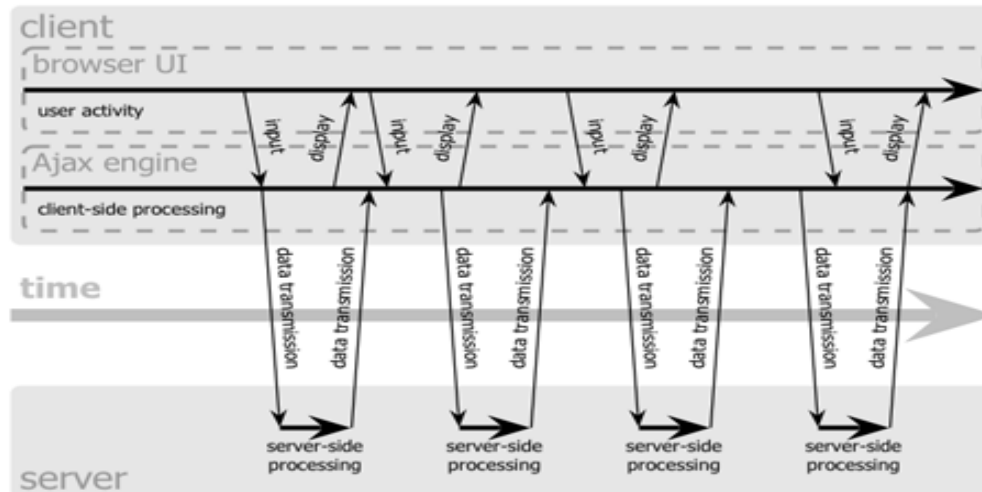
# Ajax简介

## ■ 异步调用模型

classic web application model (synchronous)

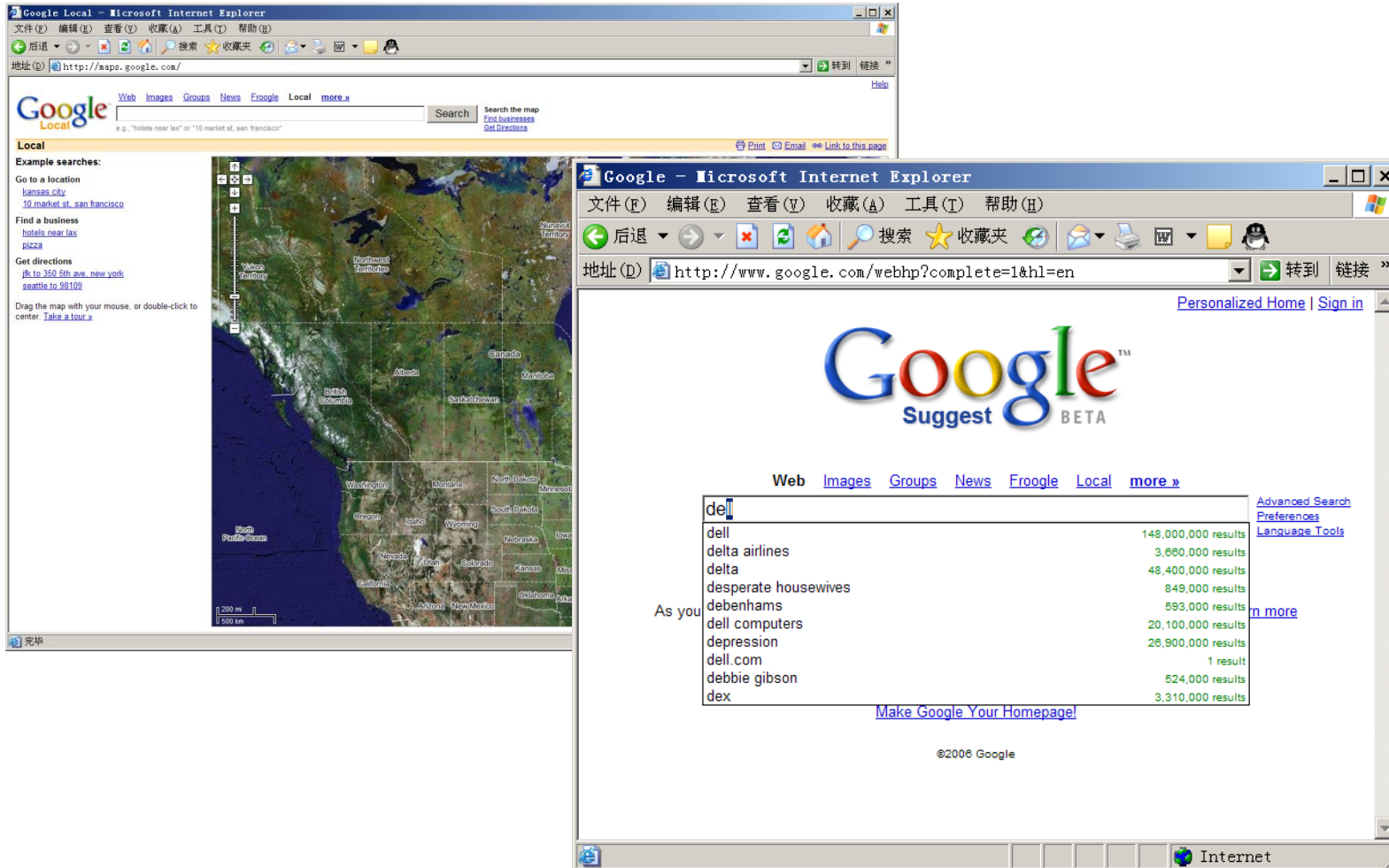


Ajax web application model (asynchronous)



# Ajax简介

应用: Gmail, Google Suggest, 以及Google Maps



## 服务器端请求处理

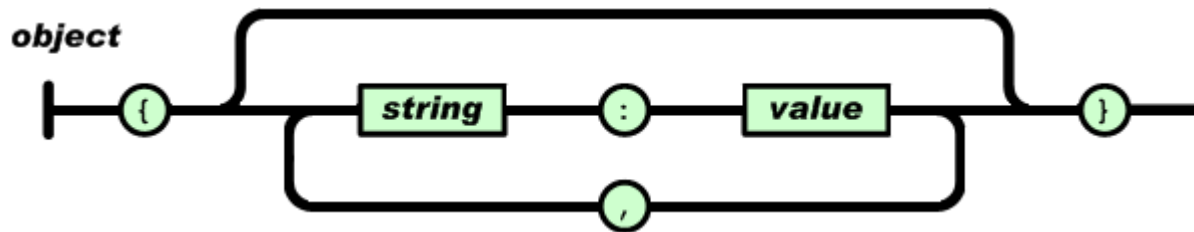
---

- 服务器端编程模型保持不变
  - 接受标准的HTTP GETs/POSTs
  - 可以使用Servlet, JSP, JSF, ...
- 客户端更频繁和细粒度的请求
  - 相应的内容类型可以是
    - `text/xml`
    - `text/plain`
    - `text/json`
    - `text/javascript`

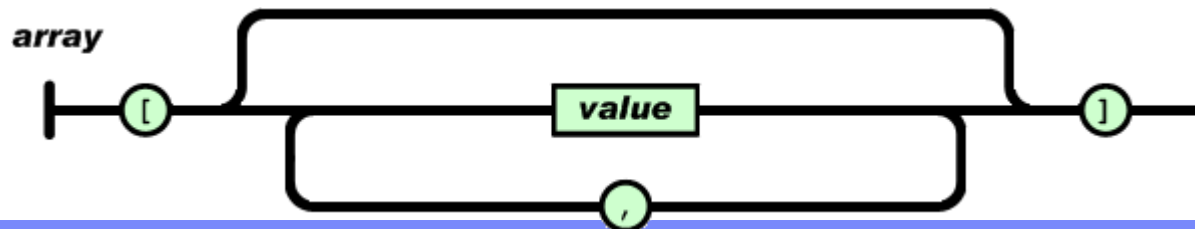
# 服务器端请求处理

## ■ Json说明

- JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式
- JSON 语法是 JavaScript 对象表示语法的子集。有对象和数组两种结构。
- 对象：对象是一个无序的“‘名称/值’对”集合。对象在js中表示为“{}”括起来的内容，数据结构为 {key: value,key: value,...}的键值对的结构。这个属性值的类型可以是 数字、字符串、数组、对象几种。



- 数组：数组在js中是中括号“[]”括起来的内容，数据结构为 ["java","javascript","vb",...]



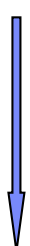


# 服务器端请求处理

## ■ Json说明

```
<?xml version="1.0" encoding="utf-8"?>
<details>
  <name>Richard Rutter</name>
  <website>http://clagnut.com/</website>
  <email>richard@clearleft.com</email>
</details>
```

```
{"person":{
  "name":"Richard Rutter",
  "website":"http://clagnut.com/",
  "email":"richard@clearleft.com"
}}
```



```
var person = function() {
  this.name = "Richard Rutter";
  this.website = "http://clagnut.com/";
  this.email = "richard@clearleft.com";
};
```

## XMLHttpRequest对象上的方法

方法	描述
<b>open</b> <b>(String method,String url,boolean asynch,String username,String password)</b>	建立对服务器的调用。 其中method表示HTTP调用方法。一般使用“GET”，“POST” url表示调用的服务器的地址 asynch表示是否采用异步方式，true表示异步 后两个参数可以不指定，username和password分别表示用户名和密码，提供http认证机制需要的用户名和密码
<b>send(content)</b>	向服务器发出请求，如果采用异步方式，该方法会立即返回。 Content可以不指定，其内容可以是DOM对象，输入流或是字符串。
<b>setRequestHeader(String header,String value)</b>	设置HTTP请求中的指定首部header的值为value。 此方法需在open方法以后调用。

## XMLHttpRequest对象上的方法

- 使用**GET**和**POST**的区别:

- “GET” 方式传递给服务器的信息一般以后缀参数方式存在于URL地址中，而URL的长度通常都有限制

```
xmlhttp.open("GET","AJAXServer?name=" +  
encodeURIComponent(encodeURIComponent(name)),true);  
xmlhttp.send(null);
```

- “POST” 方式传递给服务器的信息一般位于http协议的第三部分

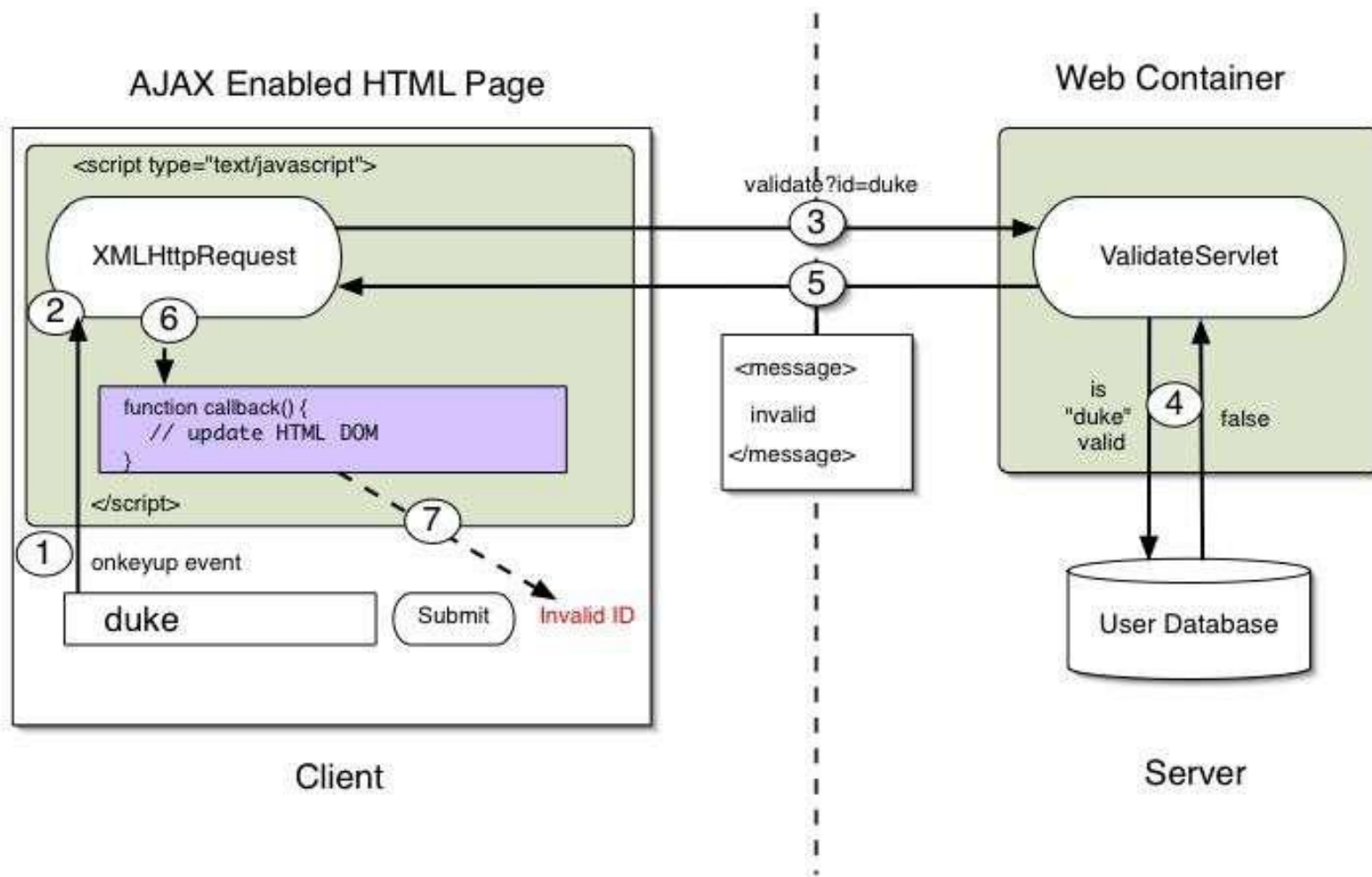
```
xmlhttp.open("POST","AJAXServer",true);  
xmlHttp.setRequestHeader("Content-Type","application/x-www-form-  
urlencoded");  
xmlhttp.send("name=" + encodeURIComponent(encodeURIComponent(name)));
```

## XMLHttpRequest对象上的方法

---

方法	描述
<b>abort()</b>	停止当前请求
<b>getAllResponseHeaders()</b>	<p>返回包含HTTP请求的所有响应头信息，其中响应头包括Content-Length,Date,URI等内容。</p> <p>返回值是一个字符串，包含所有头信息，其中每一个键名和键值用冒号分开，每一组键之间用CR和LF（回车加换行符）来分隔</p>
<b>getResponseHeader(String header)</b>	返回HTTP请求的响应头中指定的键名header对应的值

# 数据验证示例



## AJAX操作步骤

---

1. 一个客户端事件发生
2. 创建一个XMLHttpRequest对象
3. 配置XMLHttpRequest对象
4. XMLHttpRequest对象发起一个异步请求
5. ValidateServlet返回一个包含结果数据的XML文档
6. XMLHttpRequest对象调用callback() 功能并处理结果
7. 更新HTML DOM

# AJAX操作步骤

---

## ■ 1. 发生一个客户端事件

- 客户端的事件引发一个JavaScript函数调用。
- 例如: `validateUserId()` JavaScript函数作为一个事件处理器, 响应一个id为“userid” 的输入域上的onkeyup事件。

```
<input type="text"  
      size="20"  
      id="userid"  
      name="id"  
      onkeyup="validateUserId();">
```

# AJAX操作步骤

## ■ 2. 创建一个XMLHttpRequest对象

```
var req;  
function initRequest() {  
    if (window.XMLHttpRequest) {  
        req = new XMLHttpRequest();  
    } else if (window.ActiveXObject) {  
        isIE = true;  
        req = new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}  
  
function validateUserId() {  
    initRequest();  
    req.onreadystatechange = processRequest;  
    if (!target) target = document.getElementById("userid");  
    var url = "validate?id=" + escape(target.value);  
    req.open("GET", url, true);  
    req.send(null);  
}
```



## AJAX操作步骤

- 3. 一个XMLHttpRequest对象被配置具有一个回调函数

```
var req;  
function initRequest() {  
    if (window.XMLHttpRequest) {  
        req = new XMLHttpRequest();  
    } else if (window.ActiveXObject) {  
        isIE = true;  
        req = new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}  
  
function validateUserId() {  
    initRequest();  
    req.onreadystatechange = processRequest; // callback function  
    if (!target) target = document.getElementById("userid");  
    var url = "validate?id=" + escape(target.value);  
    req.open("GET", url, true);  
    req.send(null);  
}
```

## Steps of AJAX Operation

- 4. XMLHttpRequest对象发起一个异步请求

```
function initRequest() {  
    if (window.XMLHttpRequest) {  
        req = new XMLHttpRequest();  
    } else if (window.ActiveXObject) {  
        isIE = true;  
        req = new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}  
  
function validateUserId() {  
    initRequest();  
    req.onreadystatechange = processRequest;  
    if (!target) target = document.getElementById("userid");  
    var url = "validate?id=" + escape(target.value);  
    req.open("GET", url, true);  
    req.send(null);  
}
```

URL设为validate?id=greg

## Steps of AJAX Operation

- 5. 服务器端的ValidateServlet返回一个包含结果的XML文档

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {

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

    if ((targetId != null) && !accounts.containsKey(targetId.trim())) {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>true</valid>");
    } else {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>false</valid>");
    }
}
```

## Steps of AJAX Operation

- 6. XMLHttpRequest对象调用回调函数并处理结果
  - XMLHttpRequest 对象配置为当状态改变为readyState时，调用processRequest() 函数.

```
function processRequest() {  
    if (req.readyState == 4) {  
        if (req.status == 200) {  
            var message = ...;  
        }  
    }  
}
```

```
var message =  
req.responseXML.getElementsByTagName("valid")[0].childNodes[0].nodeValue;
```

# Steps of AJAX Operation

---

## ■ 7. 更新HTML DOM

- JavaScript使用DOM API 得到页面中元素的引用。

```
document.getElementById("userIdMessage")
```

“userIdMessage” 为HTML 文档中的元素ID。

- 应用JavaScript修改元素的属性，样式属性，添加、删除或者修改子元素。

```
1. <script type="text/javascript">
2. function setMessageUsingDOM(message) {
3.     var userMessageElement = document.getElementById("userIdMessage");
4.     var messageText;
5.     if (message == "false") {
6.         userMessageElement.style.color = "red";
7.         messageText = "Invalid User Id";
8.     } else {
9.         userMessageElement.style.color = "green";
10.        messageText = "Valid User Id";
11.    }
12.    var messageBody = document.createTextNode(messageText);
13.    // if the messageBody element has been created simple replace it otherwise
14.    // append the new element
15.    if (userMessageElement.childNodes[0]) {
16.        userMessageElement.replaceChild(messageBody,
17.                                         userMessageElement.childNodes[0]);
18.    } else {
19.        userMessageElement.appendChild(messageBody);
20.    }
21.}
22.</script>
23.<body>
24.  <div id="userIdMessage"></div>
25.</body>
```

## AJAX: DOM API和InnerHTML

---

- DOM APIs 提供了使用JavaScript代码来浏览/修改页面内容。

```
function setMessageUsingDOM(message) {  
    var userMessageElement = document.getElementById("userIdMessage");  
    var messageText;  
    if (message == "false") {  
        userMessageElement.style.color = "red";  
        messageText = "Invalid User Id";  
    } else {  
        userMessageElement.style.color = "green";  
        messageText = "Valid User Id";  
    }  
    var messageBody = document.createTextNode(messageText);  
    if (userMessageElement.childNodes[0]) {  
        userMessageElement.replaceChild(messageBody,  
            userMessageElement.childNodes[0]);  
    } else {  
        userMessageElement.appendChild(messageBody);  
    }  
}
```

## AJAX: DOM API & InnerHTML

---

- 使用innerHTML更方便

```
function setMessageUsingDOM(message) {  
    var userMessageElement = document.getElementById("userIdMessage");  
    var messageText;  
    if (message == "false") {  
        userMessageElement.style.color = "red";  
        messageText = "Invalid User Id";  
    } else {  
        userMessageElement.style.color = "green";  
        messageText = "Valid User Id";  
    }  
    userMessageElement.innerHTML = messageText;  
}
```

Demo ajax-stock in tomcat



# Ajax

---

- **Ajax开发框架**

- **GWT**

- **Google Web Toolkit**

- **DWR**

- **Direct Web Remoting**

- **DOJO Toolkit**

- **ExtJS**

- **JQuery**

# Jquery下的Ajax

---

```
$.ajax({  
  url: "/api/getWeather",  
  data: {  
    zipcode: 97201  
  },  
  success: function( data ) {  
    $( "#weather-temp" ).html( "<strong>" + data + "</strong> degrees" );  
  }  
});
```

使用查询参数`zipcode=97201`调用服务器端 `/api/getWeather`, 并使用返回的文本替换元素 `#weather-temp`的html代码.

# Jquery下的Ajax

## 其他方式:对\$.ajax进行了封装以简化

- **load**( url, [data], [callback] ) : 载入远程 HTML 文件代码并插入至 DOM 中。

```
$( "button" ).click(function() {  
    $( "#div1" ).load("demo_test.txt",function(responseTxt,statusTxt,xhr) {  
        if(statusTxt=="success")  
            alert("外部内容加载成功！");  
        if(statusTxt=="error")  
            alert("Error: "+xhr.status+": "+xhr.statusText);  
    });  
});
```

- **jQuery.get**( url, [data], [callback] ) : 使用GET方式来进行异步请求

```
$( "button" ).click(function() {  
    $.get("demo_test.asp",function(data,status) {  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});
```

# Jquery下的Ajax

其他方式:对\$.ajax进行了封装以简化

**jQuery.post( url, [data], [callback], [type] )** : 使用POST方式来进行异步请求

```
$( "button" ).click(function() {  
    $.post("demo_test_post.asp",  
    {  
        name:"Donald Duck",  
        city:"Duckburg"  
    },  
    function(data,status){  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});
```

**jQuery.getScript( url, [callback] )** : 通过 GET 方式请求载入并执行一个 JavaScript 文件

```
$.getScript("AjaxEvent.js", function(){  
    alert("AjaxEvent.js 加载完成并执行完成"); }  
);
```

# AJAX总结

---

## 通过XMLHttpRequest实现DHTML上的异步通讯能力

### ■ 优点

- 至今最流行可用的RIA技术
- 大量实际应用和框架支持
- 无须插件

### ■ 缺点

- 由Javascript和AJAX引擎导致的浏览器的兼容性。
- 页面局部刷新，导致后退功能失效。
- 对流媒体的支持没有FLASH、Java Applet好。
- 一些手持设备（如智能手机、PDA等）支持性差

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。

## Reference books



书名：Ajax基础教程

作者：(美)阿斯利森 舒塔

来源：人民邮电出版社

出版时间：2006年02月

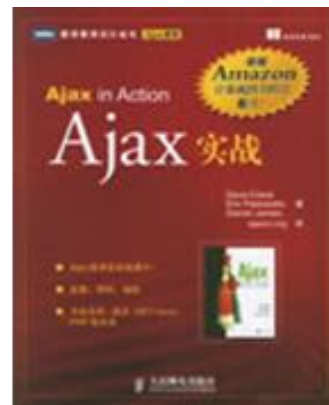


书名：Ajax实战(Ajax in action中文版)

作者：克拉恩 帕斯卡雷洛 杰姆斯

来源：人民邮电出版社

出版时间：2006年04月



书名：Ajax Hacks(英文影印版)

作者：佩里

来源：东南大学出版社

# RIA技术总结

RIA技术分类	主要技术	优点	缺点
浏览器	<b>AJAX</b>	<ul style="list-style-type: none"> <li>• 基于原有Web技术</li> <li>• 与HTML无缝集成</li> <li>• 无需学习新技术</li> </ul>	<ul style="list-style-type: none"> <li>• 把应用程序绑定在浏览器中，受浏览器安全沙箱控制</li> <li>• 浏览器的兼容性影响应用程序的开发和运行</li> </ul>
浏览器插件	<b>Flex Laszlo Silverlight</b>	<ul style="list-style-type: none"> <li>• 独立的技术</li> <li>• 与HTML部分集成</li> <li>• 丰富的界面表现</li> <li>• 特定的开发语言和工具</li> <li>• 能使用部分在Web开发中的经验</li> </ul>	<ul style="list-style-type: none"> <li>• 需要下载插件和定制的运行时</li> <li>• 运行在浏览器中，受浏览器安全沙箱控制</li> <li>• 需要学习新技术</li> </ul>
新型桌面	<b>WPF XUL AIR JavaFX</b>	<ul style="list-style-type: none"> <li>• 独立完善的技术</li> <li>• 特定的开发语言和工具</li> <li>• 独立于浏览器运行或运行于浏览器中</li> <li>• 最丰富的界面表现</li> <li>• 能方便的访问本地资源</li> <li>• 可使用桌面软件开发的技巧</li> </ul>	<ul style="list-style-type: none"> <li>• 需要安装单独的运行环境</li> <li>• 需要学习新技术</li> </ul>

# 单页应用程序

---

## ■ Single Page Application (SPA)

单页程序 **single-page application (SPA)**, 也称为单页接口 **single-page interface (SPI)**, 是一个只使用单个Web页面的 web应用或者网站, 提供用户类似桌面应用的更加流畅的体验.

### 使用Ajax开发SPA需要考虑的问题

- 搜索引擎支持
- 浏览器的前进后退按钮支持
- 收藏(书签)
- 与服务器间大量的传输
- 多次通讯之间的逻辑问题