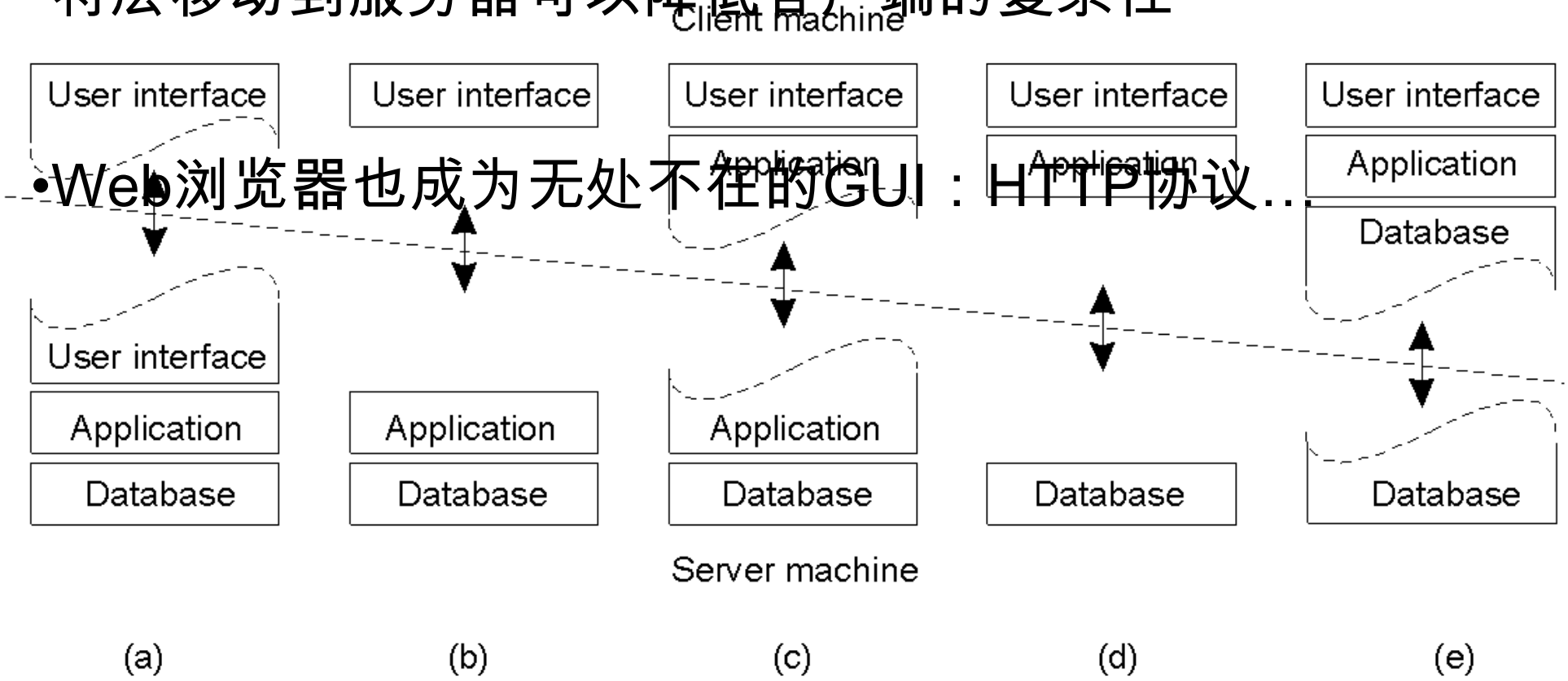
AJAX

多层架构

- 一层解决一个问题
  - –通常符合模型视图控制器（MVC）
- 将层移动到服务器可以降低客户端的复杂性
- Web浏览器也成为无处不在的GUI：HTTP协议..



HTTP：超文本传输 协议

•HTTP是支持Web浏览器和Web服务器之间通信的协议。

– “HTTP是一种应用程序级别的协议，具有分布式超媒体信息系统所需的轻量级和速度。”（ IETF ）

–通常在TCP之上（但与传输无关– RFC ）

•标准：

– RFC 1945（HTTP 1.0）

– RFC 2068和2616（HTTP 1.1）

•结构简单：

–客户发送请求

–服务器返回答复。

–可以通过单个TCP连接进行多个请求-答复交换。

HTTP请求

•请求行：

方法URI HTTP版本| r \ n

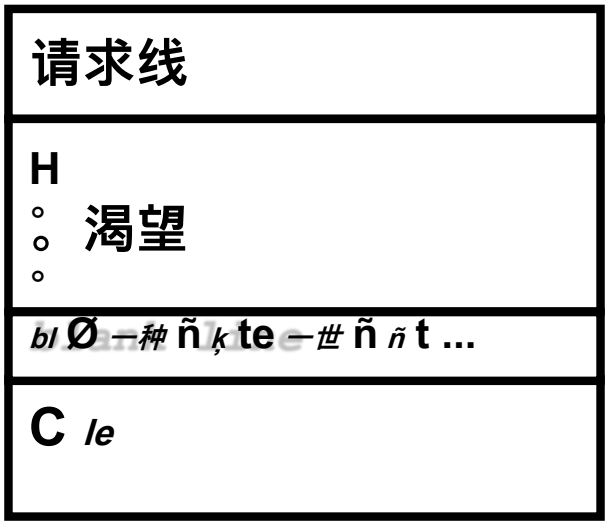
scheme：// hotstname [：port] / path或/ path

– GET：检索由URI标识的数据。

– HEAD：检索有关URI的元信息。

– POST：将数据发送到URI并检索结果。

– *PUT：将数据存储在URI命名的位置。*



•通常：

– GET用于检索HTML文档。

– HEAD用于查找文档是否已更改。

–用于提交表单的POST：包括一些
内容 (原始字节)

HTTP请求

•请求行：

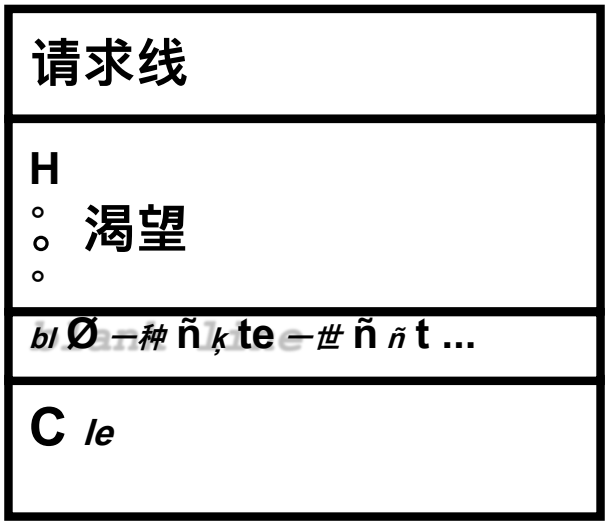
方法URI HTTP版本| r \ n

scheme：// hotstname [：port] / path或/ path

– GET：检索由URI标识的数据。
– HEAD：检索有关URI的元信息。POST：将数据发送到URI并
– 检索结果。
– *PUT：将数据存储在URI命名的位置。DELETE：删除由*
– *URI标识的实体。*
– *跟踪：通过代理，隧道等跟踪HTTP转发。*

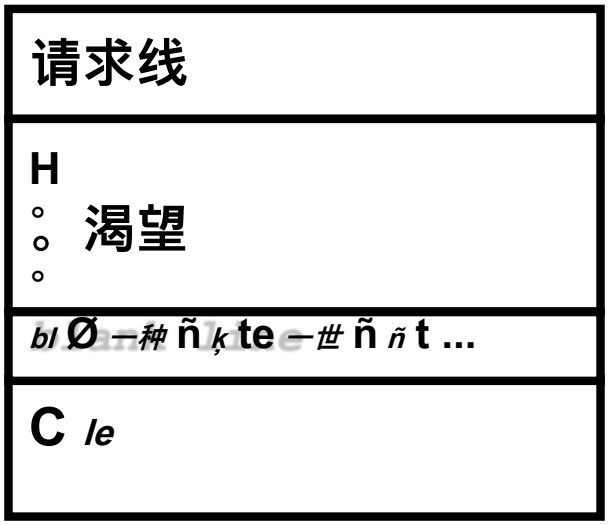
– *选项：用于确定服务器的功能或命名资源的特征。*

– *CONNECT：将请求连接转换为透明隧道（通常通过未加密的HTTP代理进行HTTPS）*



HTTP请求：标头

- 标题行：
  - –零或更多行
  - –每个标题行包含一个属性名称，后跟一个“：”然后是空格和属性值。
    - •客户类型（ **用户代理：Mozilla / 4.0** ）
    - •内容接受（ **接受：text / html** ）
    - •谁是请求者（ **主办：** www.eurecom.fr ）
    - •来源（ **推荐人：** http://google.com/blah ）
    - •POST数据的大小（ **内容长度：365** ）
  - – HTTP 1.1需要 **主办：** 标头
- •每个标头以CRLF（ **\ r \ n** ）
- •标头部分的末尾标有空白行（CRLF）



HTTP响应

•状态栏：

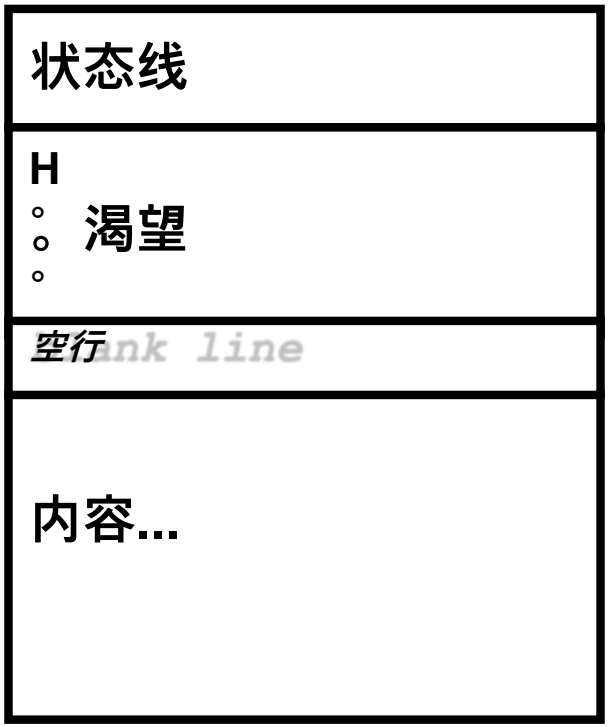
HTTP版本状态代码消息

–状态码是3位数字

- •1xx 信息性
- •2xx 成功
- •3xx 重新导向
- •4xx 客户端错误
- •5xx 服务器错误

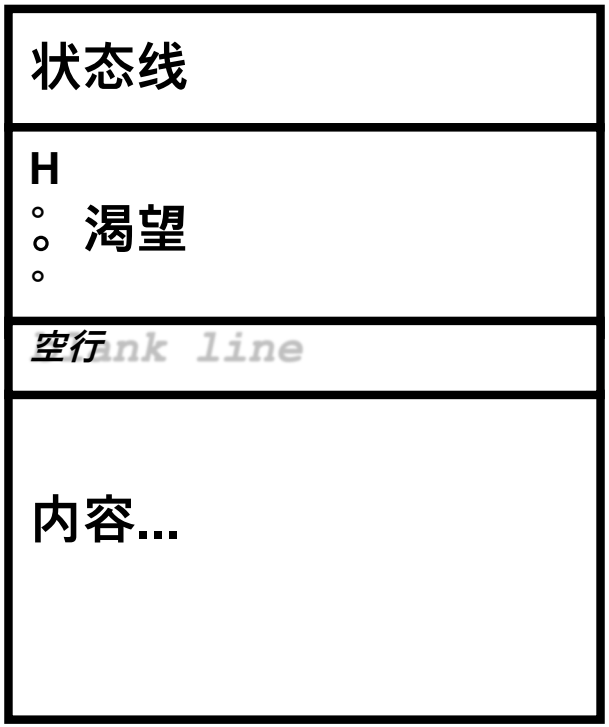
–消息为文本（针对人类）：

- •HTTP / 1.0 200确定
- •HTTP / 1.0 301永久移动
- •HTTP / 1.0 400错误请求
- •HTTP / 1.0 500内部服务器错误



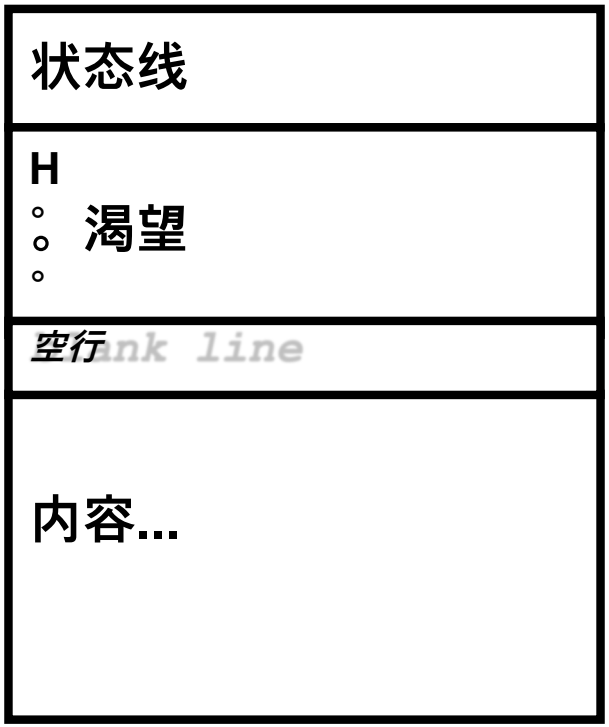
HTTP响应：标头

- 有关退回文件的信息
  - –类型（ **内容类型：text / html** ）
  - –大小（ **内容长度：1756** ）
  - –编码（ **内容编码：gzip** ）
  - –最后修改（ **日期：12月1日，星期二 2009 09:18:17 CET** ）
  - –发起人（ **伺服器：Apache / 1.17** ）



HTTP响应：内容

- 内容可以是任何内容（不仅仅是文本）
  - –通常是HTML文档或某种图像。



与HTTP的交互

- 单个请求/答复：
  - 客户端打开连接并发送请求。
  - 服务器在连接上发回相应的答复。
  - 服务器关闭其插槽。
  - 请求另一个文档的客户端打开一个新连接。
  - HTTP / 1.0的默认行为

- 持久连接：
  - 可以通过单个TCP连接处理多个请求。
  - 由以下情况触发的行为 **活着：** HTTP / 1.0客户端的标头
  - HTTP / 1.1的默认行为
  - 通过 **连接：** 标头（HTTP / 1.1）

传统Web应用程序

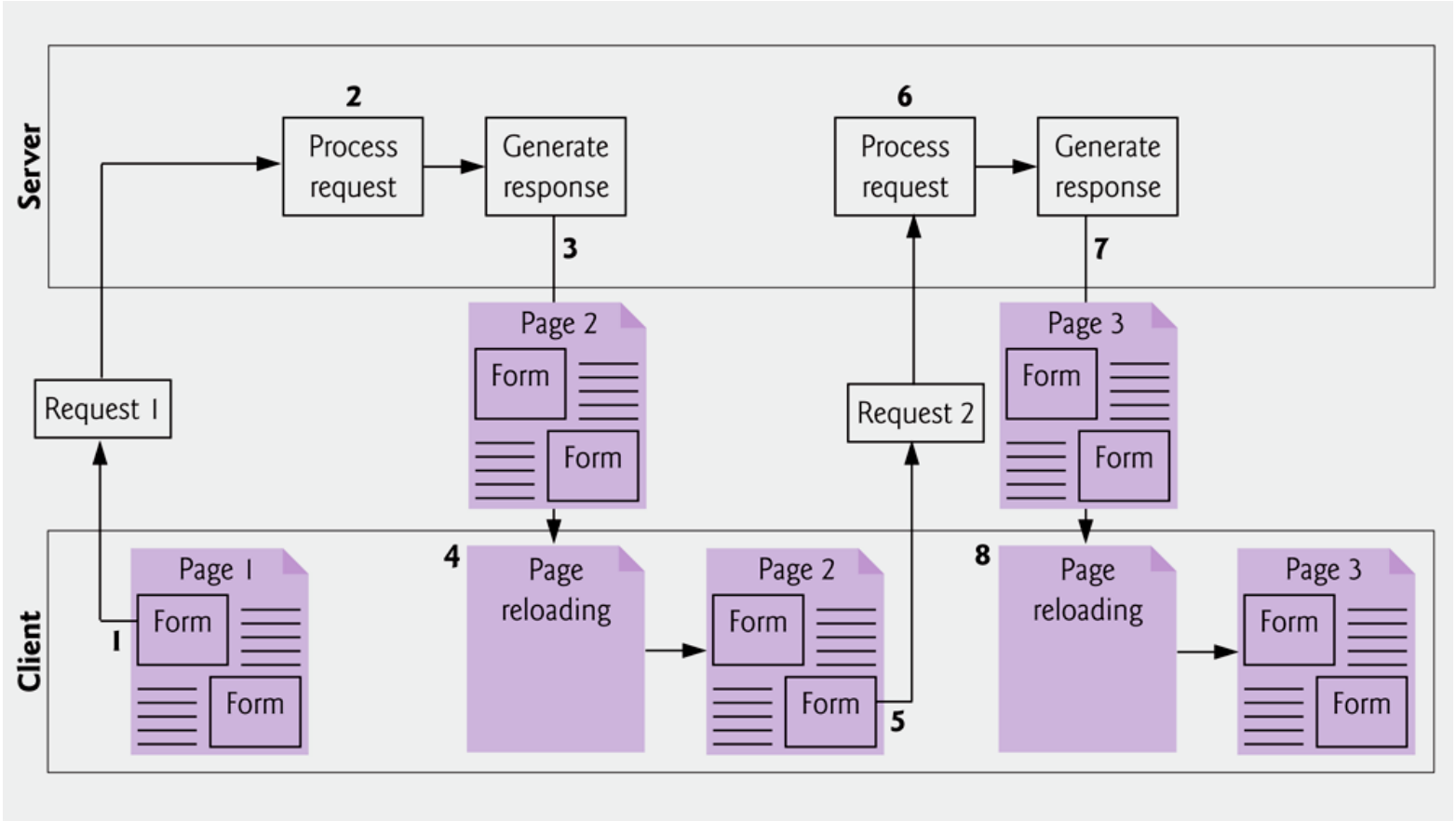
- “单击，等待并刷新”用户互动
  - –从服务器刷新所有事件，数据提交和导航所需的页面

- 同步“请求/响应”沟通模式
  - –用户必须等待响应
- 页面驱动：工作流基于页面
  - –页面导航逻辑由服务器确定

- 网页首先是静态的
  - –仅基于HTML
  - –扩展了CGI脚本以在服务器端生成动态内容

  - –在客户端也引入了XML方言（例如，来自W3C的Xforms，目前还不是HTML5的优先级）

传统Web应用程序


Server
Client
2
Process request
Generate response
3
6
Process request
Generate response
7
Page 2
Form
Form
Page 3
Form
Form
Request 1
Request 2
Page 1
Form
Form
1
4
Page reloading
Page 2
Form
Form
5
8
Page reloading
Page 3
Form
Form

传统Web应用程序：优点

- 归功于标准化
  - 正在进行的流程集中在W3C
- 简单的编程模型
  - 非常适合工作流程
  - 例如，商人收购工作流程
- HTTP协议以不同方式支持状态管理。

传统Web应用程序：缺点

- 中断用户操作
  - 用户在等待响应时无法执行任何操作
- 刷新期间失去操作环境
  - 屏幕上的信息丢失
  - 失去滚动位置
- 没有关于用户活动的即时反馈
  - 用户必须等待下一页
- 受HTML约束
  - 缺少有用的小部件

- 这解释了“ Rich Internet Applications”（ RIA ）的出现。

富互联网应用（RIA）技术

•DHTML（也包含隐藏的iframe）

•Java Applet和WebStart

•Macromedia Flash

•Microsoft Silverlight

•。NET –无接触部署

•AJAX，HTML5，CSS，jQuery

•与AJAX并发的技术问题

–需要插件

–移动浏览器没有很好的支持（例如，iOS中的性能和功耗问题）

•一个普遍关注的问题

–设计（Flash，Javascript）或浏览器实现（Java）的安全性问题

AJAX应用程序的真实示例

- 谷歌地图
  - http://maps.google.com/
- Google建议
  - http://www.google.com/webhp?complete=1&hl=zh_CN
- NetFlix
  - http://www.netflix.com/BrowseSelection?lnkctr=nmhbs
- Gmail
  - http://gmail.com/
- 雅虎地图
  - http://maps.yahoo.com/

已经相当普遍，到处都有更多的人……

什么是AJAX？

- *“异步Javascript和XML（HttpRequest）”*
  - –允许更新网页而无需重新加载页面
    - •创造更好的用户体验

•AJAX本身并不是真正的技术，而是技术的组合

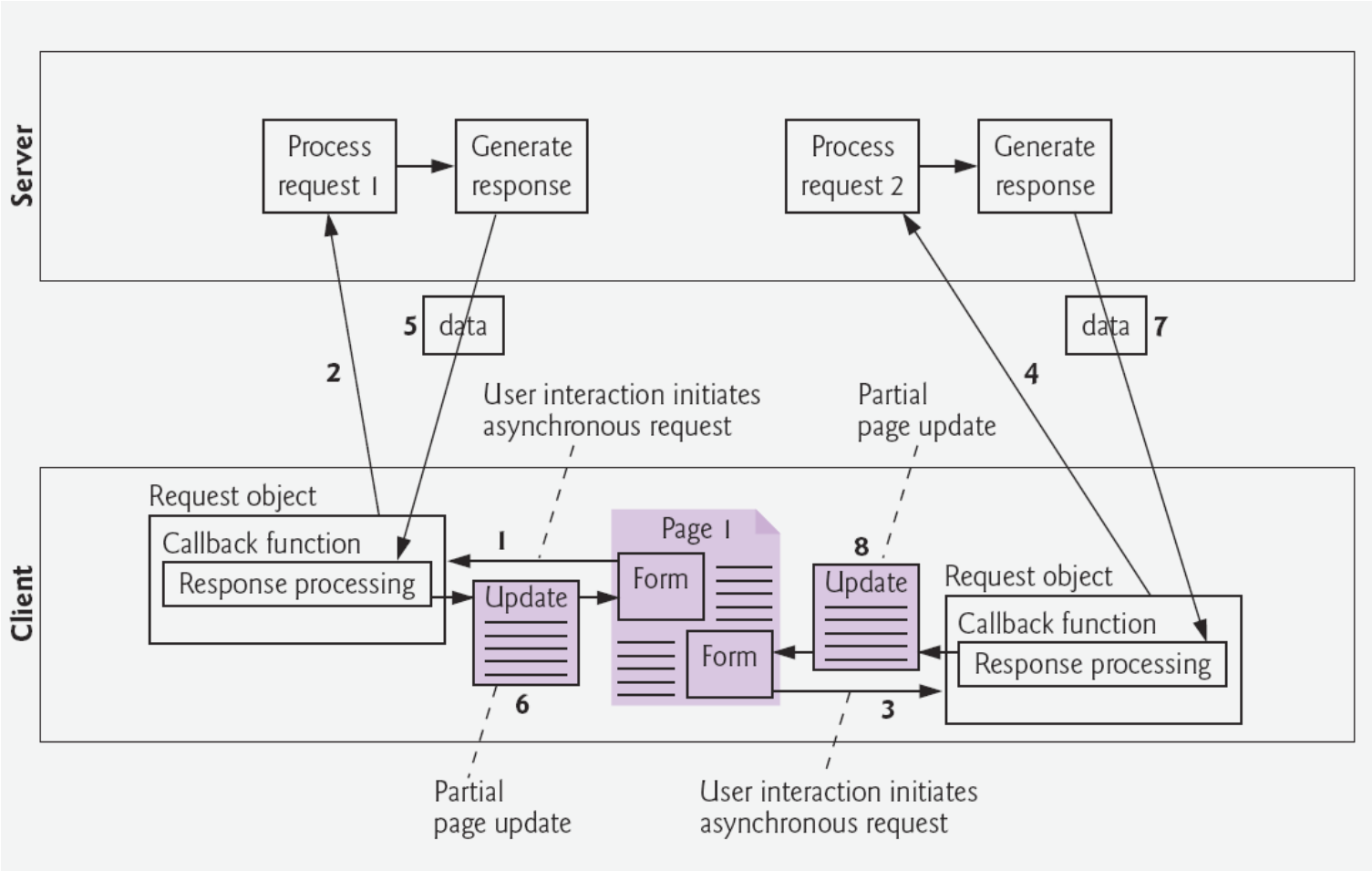
- •XHTML，CSS，DOM，XSLT
- •XML，JSON
- •Java脚本和某些用于处理数据的服务器脚本语言（服务器端脚本可以在PHP，.NET，Java Servlet或Java Server Pages中完成）

•AJAX结合了以下功能： **异步交互** 与服务器端组件和 **动态更新/重写HTML页面的源** 在浏览器中基于生成的XML / JSON /文本响应

- •最初是W3C标准的一部分，但Firefox，Internet Explorer，Safari，Opera和其他流行的浏览器都支持（以不同的方式）。
- •“ AJAX”一词创建于2005年，但在此之前的几年中，Internet Explorer首次支持核心XMLHttpRequest对象。

AJAX方法

- AJAX Web应用程序在客户端和服务器之间添加了一层，以管理两者之间的通信。



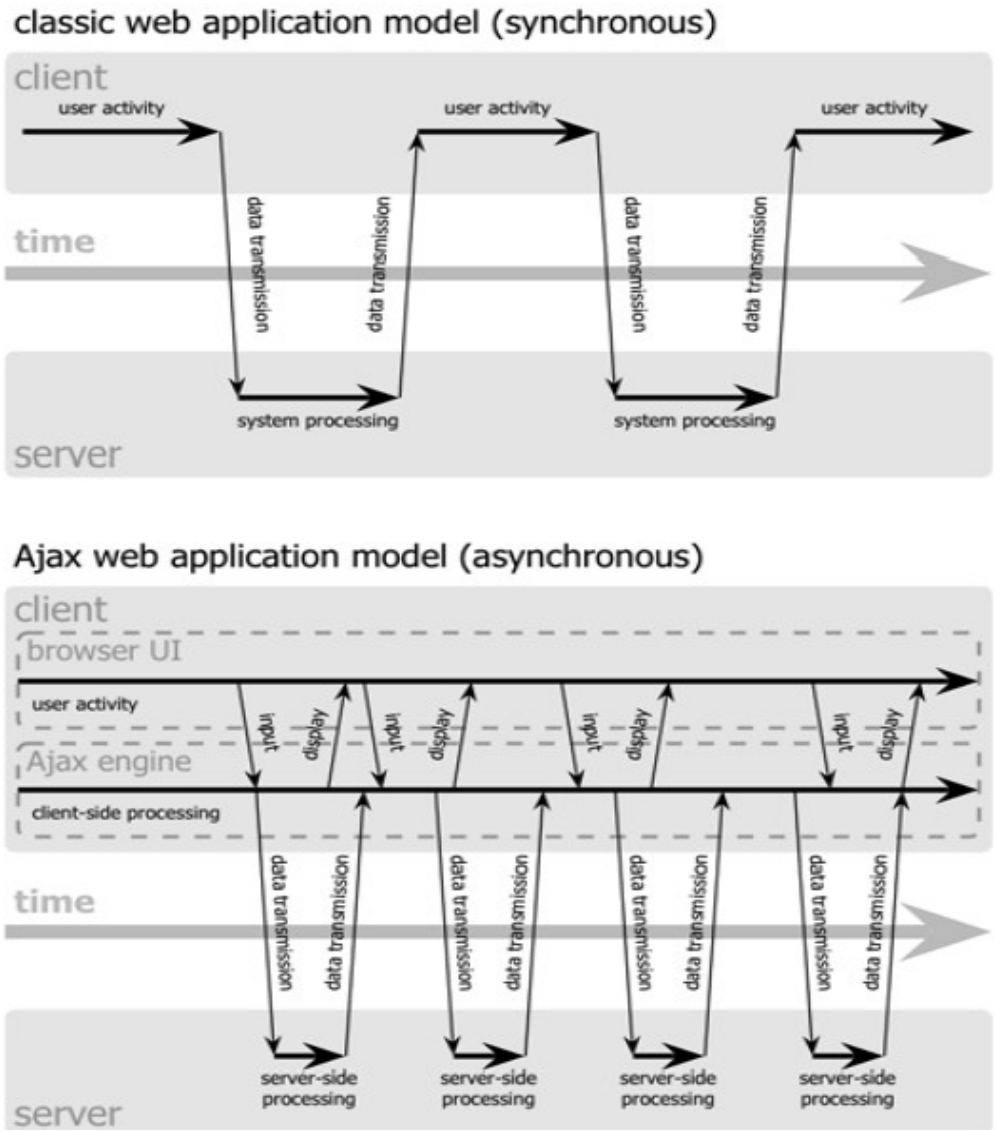
AJAX：客户端-服务器交互

- 当用户与页面交互时，客户端从服务器请求信息（步骤1）。
- 该请求被AJAX控件拦截，并作为 *异步请求（* 第2步）
- 当服务器处理请求时，用户可以继续与客户端浏览器中的应用程序进行交互。
- 其他用户交互可能会导致对服务器的其他请求（步骤3和4）。
- 一旦服务器响应了原始请求（步骤5），AJAX控件就会调用客户端功能来处理服务器返回的数据，并可能将其显示给用户。
- 此函数称为回调函数，它使用部分页面更新（步骤6）来显示数据，而无需重新加载整个页面。
- 同时，服务器可能正在响应第二个请求（步骤7），而客户端浏览器可能正在开始另一个部分页面更新（步骤8）。

AJAX与HTTP请求/响应

- 标准请求/响应
  - –同步：操作被阻止，甚至禁用页面显示
  - –每次单击超链接或按钮都会显示（和传输）和整个新页面
- AJAX：
  - –异步：每个动作在后台发送数据并接收结果
  - –回调函数仅更新页面的指定部分。
  - –这样的部分页面更新有助于使Web应用程序响应速度更快，使其更像桌面应用程序。
  - – Web应用程序在用户与其交互时不会加载新页面。

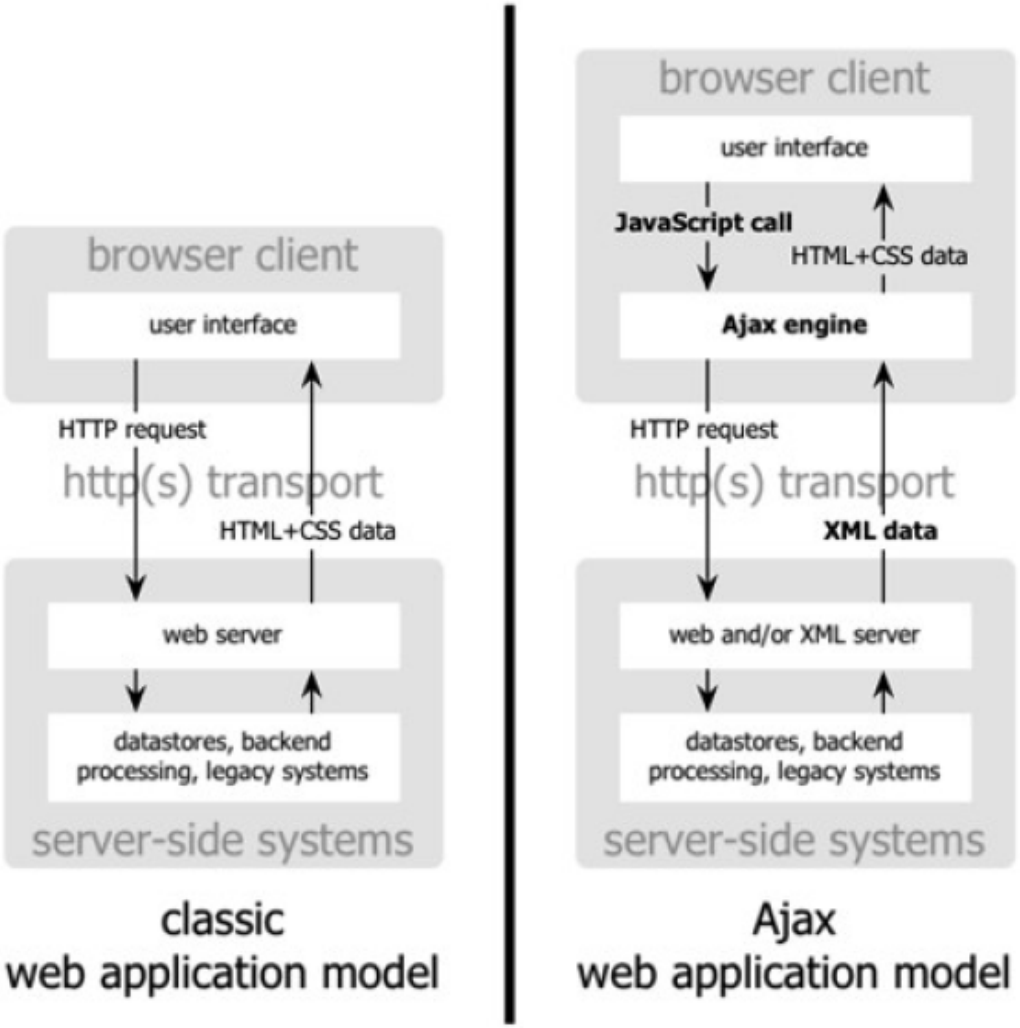
AJAX：客户端-服务器交互



Interrupted user operation while the data is being fetched

Uninterrupted user operation while data is being fetched

AJAX堆栈


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend
processing, legacy systems
server-side systems
classic
web application model
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend
processing, legacy systems
server-side systems
Ajax
web application model

为什么选择AJAX？

- 直观自然的用户交互
  - –无需特殊点击
  - –鼠标移动足以触发事件
- “部分屏幕更新”替代了“单击，等待和刷新”用户交互模型
  - –仅更新包含新信息的用户界面元素（快速响应）
  - –其余用户界面保持显示而不会中断（不会丢失操作上下文）
- 数据驱动（而不是页面驱动）
  - –在服务器提供数据的同时，在客户端中处理UI
- 异步通信取代了“同步请求/响应模型”。
  - –当客户端程序在后台从服务器请求信息时，用户可以继续使用该应用程序
  - –将显示与数据提取分开
- 宽带通信的出现使之成为可能
  - –基于AJAX的JavaScript可能需要大量带宽才能下载

实施AJAX

- 要实现AJAX，我们需要回答三个问题：
  - 什么触发了AJAX请求？
    - 通常是Javascript事件（onblur，onclick等）
  - 处理AJAX请求并发出响应的服务器进程是什么？
    - 某种URL（使用服务定位器）
  - 什么处理来自服务器的响应（什么是回调方法）？
    - 一个Javascript函数，它基于返回的文本获取响应并处理DOM。

AJAX中使用的技术

- Javascript
  - –松散键入的脚本语言
  - – **页面中发生事件时调用的Javascript函数**
  - –整个AJAX操作的胶水
- DOM
  - –用于访问和处理结构化文档的API
  - –表示XML的结构 **和HTML文档**
- CSS
  - –附加到HTML文档的样式表 **可以通过编程方式进行更改** 通过Javascript
- XMLHttpRequest
  - –执行的JavaScript对象 **异步交互** 与服务器

JavaScript语言

- 由Brendan Eich在1995年于Netscape Navigator 2.0中创建
  - –以LiveScript的名义引入“ JavaScript”，用于营销目的
  - –标准化为ECMAScript
- 基于原型的脚本语言
  - – 动态类型（与值关联的类型，而不是可以连续具有不同类型的变量）
  - – 基于对象：JavaScript对象=关联数组+原型（无继承）
  - – 函数是第一类（本身是对象），因此具有属性和方法（例如
    。call（）和.bind（））
  - – 解释的语言：例如，可以在字符串上调用eval（）某些性能问题，存在优化（取决于浏览
  - – 器）单线程
  - –

JavaScript用法

- 在用户的浏览器上运行
  - –放在HTML代码中的<script> ... </ script>之间
  - –在页面加载时运行
- 也用于服务器端
  - –最初由Netscape（1995）
  - –尤其是Node.js框架（2009），用于实现事件驱动的非阻塞I / O服务器

- 用法
  - AJAX
  - 页面元素的动画，淡入和淡出，调整大小，移动它们等。交互式内容，例如游戏，以及播放音频和视频
  - –
  - 在提交表单之前验证表单的输入值，以确保它们是可接受的
  - 收集用户的阅读习惯和浏览活动（网络分析，广告跟踪，个性化）

JavaScript用法

•示例脚本：

```
< ! DOCTYPE html>
 <meta charset =“ utf-8”>
<title>最小示例</ title>
 <h1 id =“ header”>这是JavaScript </ h1>

<脚本>

    document.body.appendChild ( document.createTextNode ( 'Hello World ! ' ) ) ;

    var h1 = document.getElementById ( 'header' ) ; //保存对<h1>标记的引用
    h1 = document.getElementsByTagName ( 'h1' ) [0]; //访问相同的<h1>元素

</ script>

<noscript>您的浏览器不支持JavaScript或已将其关闭。</ noscript>
```

在页面中包含JavaScript

- 内联：

```
<html> <head> </ head>
  <身体>
    <h1>这是我的文档</ h1> <script type =“ t
    ext / javascript”>
      // < ! [CDATA [
        alert ( “ Hello from JavaScript” ) ;
      //]]>
    </ script>
    <h1>这是我的第二个标题</ h1> </ body>

</ html
```

对于XHTML和HTML（严格）验证

- 图书馆：

```
<html> <head>
    <script type =” text / javascript” src =” lib / library.js”> </ script> </ head>

  <身体>
    <script> < ！-浏览器将执行此代码-> var foo = 3;

      var bar = functionInLibrary ( foo ) ;
    </ script>
  </ body>
</ html>
```

JavaScript：事件处理

- 一些HTML标签具有包含JavaScript的属性，这些JavaScript在与该标签相关的DOM事件发生时运行
  - –例如：onchange，onclick，onmousedown，onmouseup…
  - –描述于http://en.wikipedia.org/wiki/DOM_events
  - –在HTML页面中：

```
<a href="http://www.eurecom.fr" onclick="alert('Hi!')">你好！</a>
<ahref="http://www.noshow.com"onclick="alert('Stop!');返回false;">计划B </a>
```

- 返回false禁止链接
- 更强大的事件：

```
<body onload =” message (  ) “>
<input type =“ button” onclick =“ window.location ='/';返回false;”
```

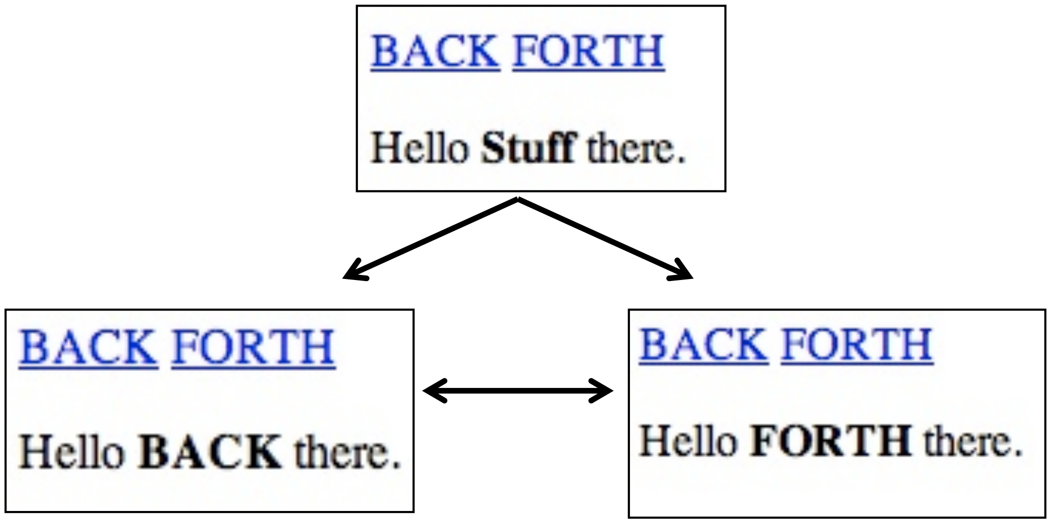
JavaScript和DOM

- JavaScript可以熟练运用 ... nts
  - 文档对象模型描述的页面的语法
  - Cf。http：// ... dia.org/wi ...

d Ø cument.getElementById ('stuff')

```
<p>
<a href="#" onclick="document.getElementById('stuff').innerHTML ='BACK';">返回</a>
<a ... innerHTML ='FORTH';"> FORTH </a>
您好<b> <span id =" stuff">材料</ span> < / b >
</ p>
```

抄送：查尔斯·塞弗兰斯（BY Charles Severance），吉姆（Jim Eng），2009



CSS：级联样式表

- 用于创建附加到网页的样 式表的数据格式
  - –包括从其他样式表（“层叠”）的继承
  - –反映文档逻辑树的格式化结构
- 目标：
  - –允许将演示样式与内容明确区分
  - –减少页面延迟传输（样式表用于网站的所有HTML页面）
- CSS用法必须与AJAX一起修改，例如：
  - –在jQuery中加载（HTML文件）：

```
<script type =“ text / javascript”>
    if（ ！（$（'＃ajaxCss'）。length ））{//检查是否已经加载
        $（“ head”）。append（“ <link>”）;;
        $（“ head”）。children（“：last”）。attr（{
          id：“ ajaxCss”，
          rel：“样式表”，
          输入：“ text / css”，
          href：“ ./your-file.css»}）;
    }
</ script>
```

  - –等待CSS文件加载（HTML文件）：< div id =“ ajaxMainFrame” style =“ visibility：hidden;”> </ div>
  - –最后使它可用（CSS文件）：< div id =“ ajaxMainFrame” style =“ visibility：hidden;”> </ div>

JSON：JavaScript对象表示法

- XML的替代品
  - –更易于解析，尤其是在JavaScript中（参见RESTful Web服务）
- 客户端：可以使用JavaScript eval（ ）方法检索数据
  - –示例：.json文件：{ “菜单”：“文件”，“命令”：[
    {” title”：“新建”，“ action”：“ CreateDoc”}，
    {“标题”：“打开”，“动作”：“ OpenDoc”}，{“标题”：“
    关闭”，“动作”：“ CloseDoc”}
  - –相应的JavaScript代码：

```
req.open ( “ GET”，“ fichier.json”，true ) ; //请求
var doc = eval ( ' ( '+ req.responseText +' ) ' ) ; // 恢复
var nameMenu = document.getElementById ( 'jsmenu' ) ; //查找nameMenu.value
 = doc .menu。价值; //分配doc.commands [0]。title //在数组中读取值“ title”

doc.commands [0] .action //在数组中读取相应的值“ action”
```

- 服务器端：使用特定于所使用语言的库：
  - – Java：org.json。*
  - – Perl：JSON
  - – PHP：json
  - –…

XMLHttpRequest

- AJAX的心脏
  - –最初由Microsoft于1999年在其浏览器中引入
  - –目标是更换框架，全屏更新，提供拖放，自动字段完成等。
- 被现代浏览器采用
  - – Mozilla™，Firefox，Safari和Opera
  - –用于获取资源的W3C API –在以下网址定义：http：//www.w3.org/TR/XMLHttpRequest/
  - –通过标准的HTTP GET / POST与服务器通信
- Javascript对象，在后台运行（不会中断用户）
  - –与后端服务器的异步通信
- XMLHttpRequest名称具有误导性：
  - –这可以发送任何数据，而不仅仅是XML（最初设想/实现）
  - –特别支持任何基于文本的格式，包括XML和JSON
  - –可用于通过HTTP和HTTPS发出请求
  - –在与HTTP有关的广义上支持“请求”；即与HTTP请求或已定义HTTP方法的响应有关的所有活动。

XMLHttpRequest对象

- 一些方法：
  - abort（ ）-停止当前请求
  - getAllResponseHeaders-以字符串形式返回完整的标题集（标签和值）
  - getResponseHeader（ ：headerLabel ”）–返回请求的标头字段的字符串值
  - –打开（“方法”,“网址”）设置待处理的请求
  - –发送（内容）–发送请求
  - setRequestHeader（“标签”,“值”）–在标题中设置标签/值
  - –…

XMLHttpRequest属性

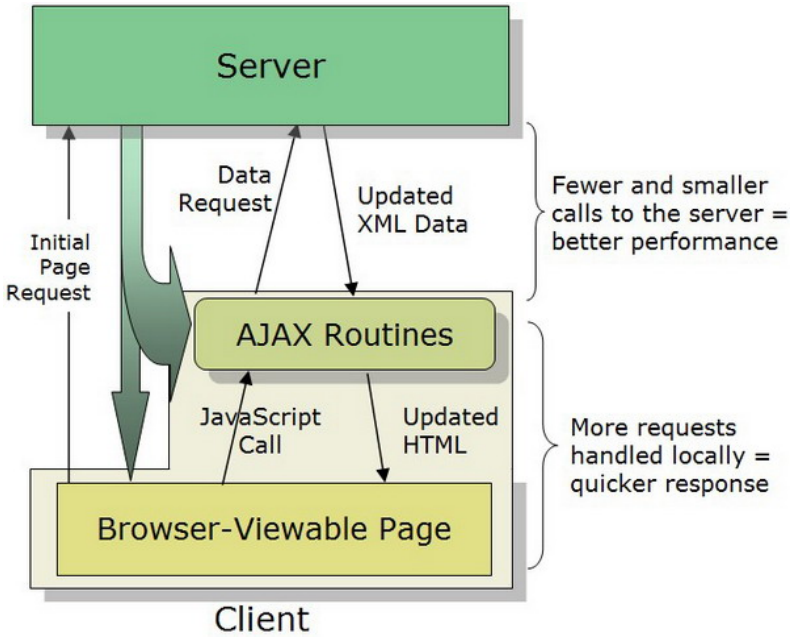
- onreadystatechange
  - 设置一个JavaScript事件处理程序，该处理程序在每次状态更改时触发
- readyState –请求的当前状态
  - 0 =未初始化
  - 1 =正在加载
  - 2 =已加载
  - 3 =交互式（已返回一些数据）4 =完成
  -
- 状态
  - 从服务器返回的HTTP状态：200 =确定
- responseText
  - –从服务器返回的数据的字符串版本
- responseXML
  - –服务器返回的数据的XML文档
- statusText
  - –服务器返回的状态文本

客户端AJAX处理

- 请求发送
  - –请求对象（XMLHttpRequest）
    被创造
  - –请求元素（URL，方法，HTTP标头，参数）
    指定的
  - –事件处理程序定义
  - –对象发送
- 响应接收
  - –对于请求状态的每次更改：测试是否处于就绪状态
  - –处理接收到的数据（页面更新，XSL转换）
    如果将XML用作数据格式等）

功能需求

施马



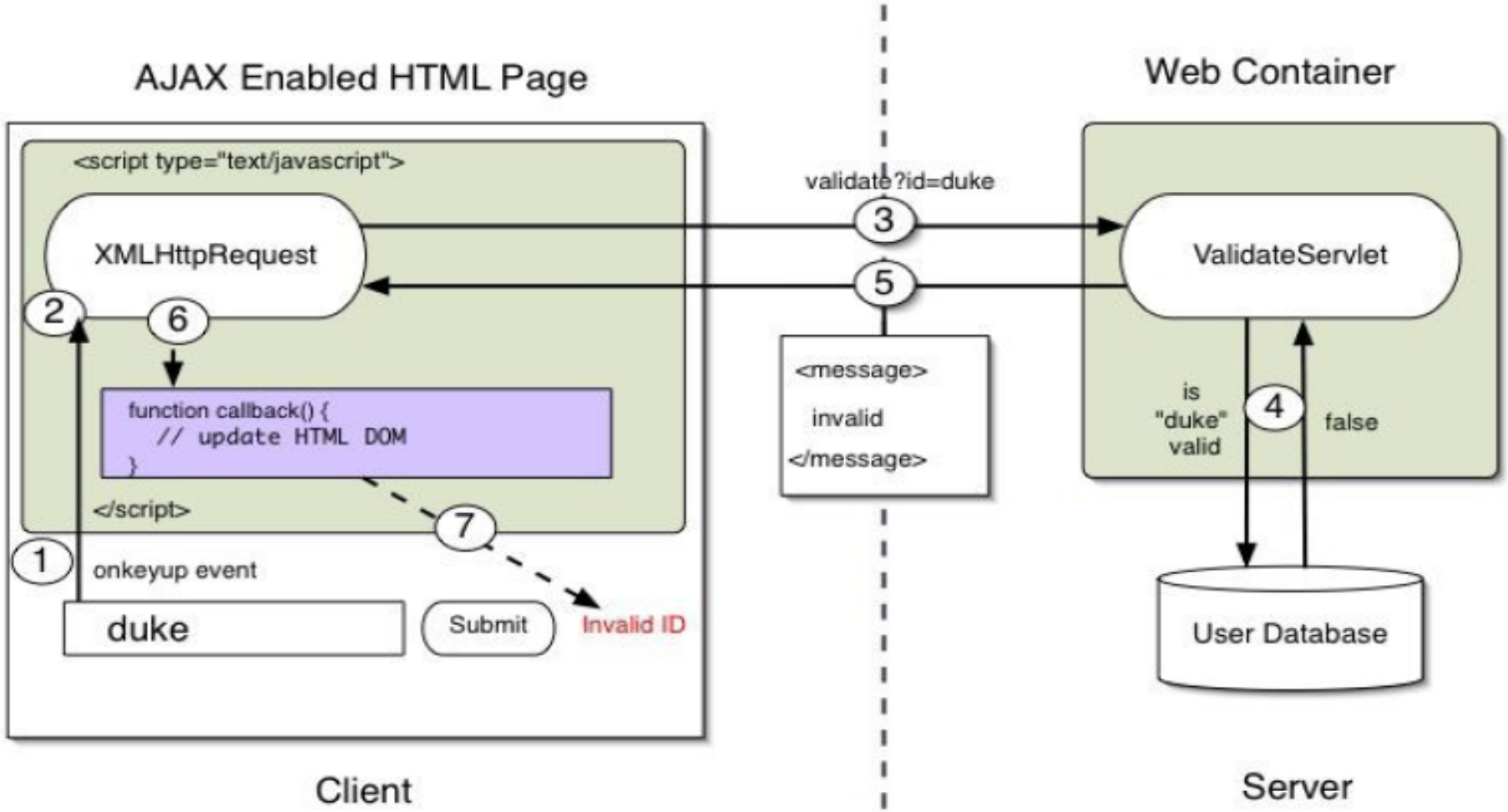
资源：HTTP://www.codeproject.com/KB/showcase/FarPointAJAX.aspx

//www.codeproject.com/KB/showcase/FarPointAJAX.aspx

服务器端AJAX请求处理

- 服务器编程模型保持不变
  - 接收标准的HTTP GET / POST
  - 可以使用Servlet，JSP，JSF，...
- 约束较小
  - 来自客户的更频繁，更细粒度的请求
  - 响应内容类型可以是
    - text / xml
    - 文字/纯文字
    - text / json
    - 文字/ JavaScript

AJAX：示例应用


AJAX Enabled HTML Page
Web Container
<script type="text/javascript">
XMLHttpRequest
validate?id=duke
3
5
ValidateServlet
2
6
function callback() {
// update HTML DOM
}
</script>
<message>
invalid
</message>
is
"duke"
valid
4
false
1
onkeyup event
7
duke
Submit
Invalid ID
User Database
Client
Server

AJAX操作步骤

1。 发生客户端事件

2。 创建一个XMLHttpRequest对象配置XMLHttpRequest对象

3。

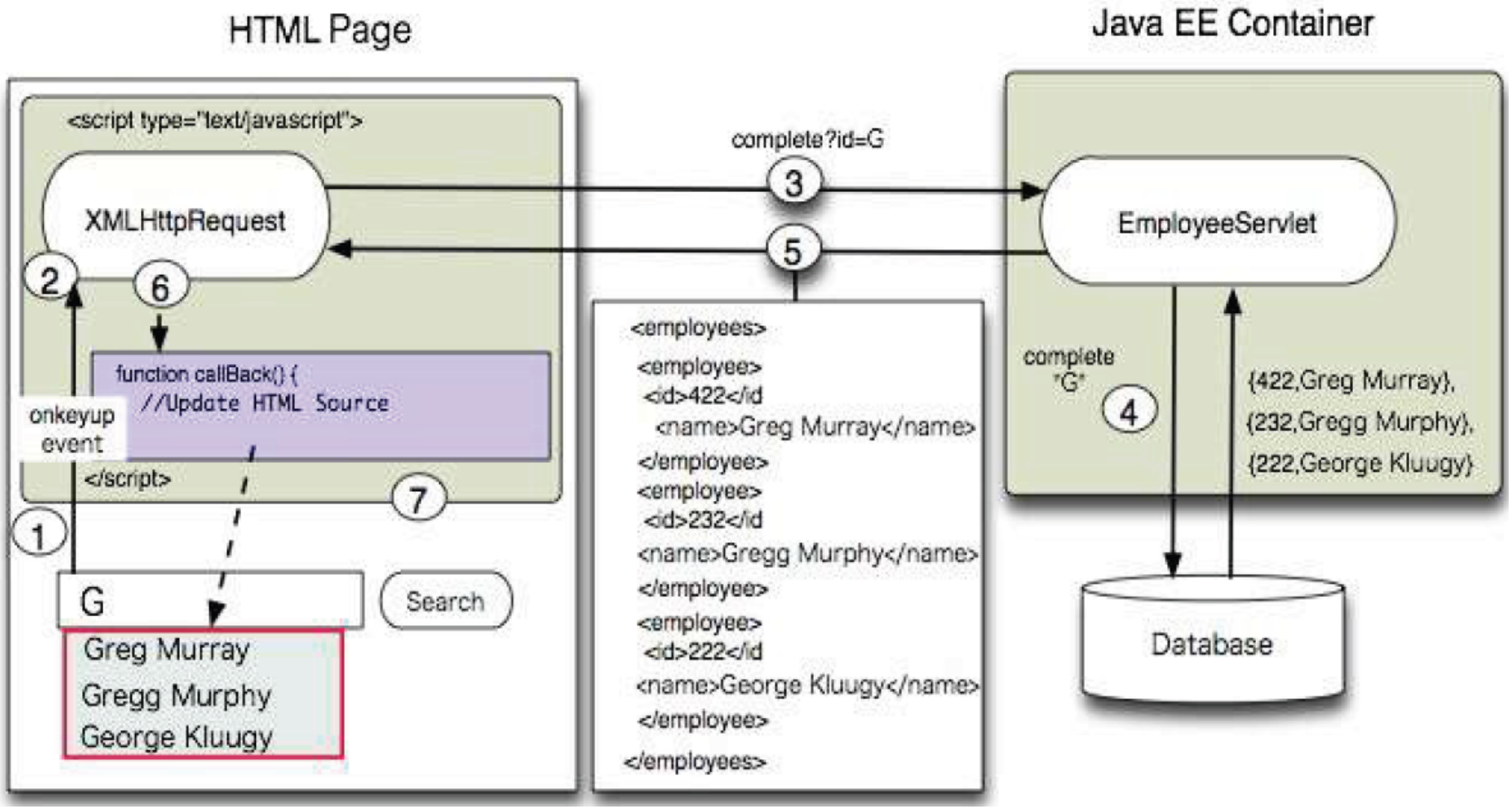
4。 XMLHttpRequest对象发出异步请求

5， ValidateServlet返回包含结果的XML文档

6。 XMLHttpRequest对象调用callback（ ）函数并处理结果

7。 HTML DOM已更新

AJAX：另一个应用程序



AJAX：涉及编码

•index.jsp页面自动完成表单

```
<form name =“ autofillform” action =“ autocomplete” method =“ get”>
    <input type =“ text” size =“ 20” autocomplete =“ off”
        id =“ completeField” name =“ id”
        onkeyup =“ doCompletion (  ) ;”>
    <input id =“ submit_btn” type =“ Submit” value =“查阅员工”> </ form>
```

AJAX：涉及编码

•客户端：自动完成XMLHttpRequest

–值得注意：对象的构造取决于浏览器！

```
函数initRequest（url）{
    如果（window.XMLHttpRequest）{
        返回新的XMLHttpRequest（ ）;
    } else if（window.ActiveXObject）{
        isIE = true;
        返回新的ActiveXObject（“Microsoft.XMLHTTP”）;
    }
}
```

AJAX：涉及编码

•客户端：自动完成事件处理程序

```
函数doCompletion（ ）{
    如果（completeField.value ==“” ）{
        clearTable（ ）;
    }其他{
        var url =“ autocomplete ? action = complete & id =” +
                                                    转义（completeField.value ）;
        var req = initRequest（url ）;
        req.onreadystatechange = function（ ）{
            如果（req.readyState == 4 ）{
                如果（需求状态== 200 ）{
                    parseMessages（req.responseXML ）;
                }否则，如果（req.status == 204 ）{
                    clearTable（ ）;
                }
            }
        };
        req.open（“ GET”，url，true ）;
        req.send（null ）;
    }
}
```

AJAX：涉及编码

•服务器端：自动完成Servlet doGet（ ）

```
公共无效doGet（HttpServletRequest请求，HttpServletResponse响应）引发IOException，ServletException
        {
。。。
字符串targetId = request.getParameter（“ id”）; 迭代器= employ
ee.keySet（ ）。iterator（ ）; while（it.hasNext（ ））{

        EmployeeBean e =（EmployeeBean）employees.get（（String）it.next（ ））; 如果（（（e.getFirstName
        （ ）。toLowerCase（ ）。startsWith（targetId）||
                e.getLastName（ ）。toLowerCase（ ）。startsWith（targetId ））&&！targetId.equals（“”））{sb.append（“ <employ
                ee>”）;
                sb.append（“ <id>” + e.getId（ ）+“ </ id>”）;
                sb.append（“ <firstName>” + e.getFirstName（ ）+“ </ firstName>”）; sb.append（“
                <lastName>” + e.getLastName（ ）+“ </ lastName>”）; sb.append（“ </ employee>”）;

                姓名=真; } //如果
} // while
if（名称收藏）{
        response.setContentType（“ text / xml”）;
        response.setHeader（“ Cache-Control”，“ no-cache”）;
        response.getWriter（ ）。write（“ <employees>” + sb.toString（ ）+“ </ employees>”）; response.setStatus（Ht
}其他{
        tpServletResponse.SC_NO_CONTENT）;
}
} // doGet
```

AJAX：涉及编码

•客户端：处理响应

```
函数parseMessages（responseXML）{
clearTable（ ）;
var employee = responseXML.getElementsByTagName（“ employees”）[0]; 如果（employees.childNode
s.length> 0）{
        completeTable.setAttribute（“ bordercolor”，“ black”）;
        completeTable.setAttribute（“ border”，“ 1”）;
}其他{
        clearTable（ ）;
}
for（循环= 0;循环<employee.childNodes.length;循环++）{
        var employee = employee.childNodes [loop];
        var firstName = employee.getElementsByTagName（“ firstName”）[0]; var lastName =
        employee.getElementsByTagName（“ lastName”）[0]; var employeeId =
        employee.getElementsByTagName（“ id”）[0];
        appendEmployee（firstName.childNodes [0] .nodeValue，lastName.childNodes [0] .nodeValue
         ，employeeId.childNodes [0] .nodeValue）;
}
}
```

框架：jQuery

- 支持AJAX的跨平台JavaScript库
  - –当今最受欢迎的图书馆
  - –请参阅其他网站：http://en.wikipedia.org/wiki/List_of_Ajax_frameworks
- 要使用jQuery，必须将其包含在HTML中
  - –通常在页面的<head>区域中完成：

```
<头>
<title> App Engine-HTML </ title>
<link href =“ / static / glike.css” rel =“ stylesheet” type =“ text / css” />
<script type =“ text / javascript” src =“ / static / js / jquery-1.2.6.min.js”> </ script>

</ head>
```

  - 旨在简化HTML的客户端脚本
  - – DOM元素的选择，遍历和操作-由其启用 *选择器引擎*（“嘶嘶声”），JSON解析
  - –可扩展（插件），尤其是：jQuery UI，用于抽象高级效果的插件，动画，可设置化的小部件等。
  - –编程风格融合了算法和DOM数据结构
    - 函数可以链接在一起，因为它们都返回jQuery对象
- 两种使用方式
  - – $函数是jQuery对象的工厂方法
  - – $。前缀的函数，不要直接作用于jQuery对象

jQuery：$函数

- $功能也称为命令
  - – jQuery对象的别名
  - –执行后，文档的DOM可用：

```
jQuery（document）.ready（function（）{
  // DOM在这里完全定义...}
```

或：$（ document）.ready（function（）{

```
//完全在此处定义的DOM…}）;
```

- 通常用于访问和操作多个DOM节点
  - –命令可能包含CSS选择器字符串：$（selector）
    - •选择器可能引用所有标签的标签名或#ID或.CLASS（标签的类属性）或*
    - •选择器还可以引用具有属性：[attr]或基于其值的标签：[value =“ val”]
  - – HTML页面中与jQuery对象匹配的元素的结果
  - –然后可以在jQuery对象或节点本身上调用方法

- 示例：找到ID =“ carmakes”的HTML SELECT元素，并添加一个值为“ VAG”和文本为“ Volkswagen”的OPTION元素：

```
$（'select＃carmakes'）。append（$（'<option />').attr({value:"VAG"}).append("Volkswagen“ ）；;
```

jQuery：$。前缀的函数

- $.-或$（...）前缀的函数，也称为实用程序函数
  - –示例：.html（'text'）扩展HTML，.css（ ）更新样式等。
- •通常用于实现独立于浏览器的AJAX查询并处理远程数据：
  - – $ .ajax函数及其关联方法
  - –同样$ .get（ ），$。post（ ），$。getScript（ ），$。getJSON（ ）

示例：将数据发布到服务器
向用户提供反馈：

```
$ .ajax ( {
  输入：“ POST”，
  网址：“ example.php”，
  数据：“名称= John＆location =波士顿”
} ) 。success ( function ( msg ) {
  alert ( “已保存数据：” + msg ) ;
} ) 。fail ( function ( xmlHttpRequest，statusText，errorThrown ) {alert (

    “您的表单提交失败。\ n \ n”
      +“ XML Http请求：” + JSON.stringify ( xmlHttpRequest )
      +“，\ n状态文字：” + statusText
      +“，\ n抛出错误：” + errorThrown ) ;
} ) ;
```

示例：更新发送的消息
通过聊天应用程序：

```
<div id =“ chatcontent”>            正在加载... </ div>

<script> / * < ! [CDATA [* /
function updateMsg ( ) {
  $ .ajax ( {
    网址：“ / messages”，
    缓存：假，
    success：function ( frag ) {$ ( “ # chatcontent” ) 。html ( frag ) ;}} ) ) ; setTimeout ( 'up
  dateMsg ( ) ，4000 ) ;
}
updateMsg ( ) ;
/ *]]> * / </ script>
```

鲁迪耶

AJAX：结论

- URL：次要问题
  - 浏览器中的历史记录
  - 为页面添加书签
    - 如果AJAX实现了功能完善的应用程序（例如电子表格），这不是问题
  - 搜索引擎索引（SEO）
- 解决方案：
  - URL锚（ # ）可以作为解决方案进行修改
  - 独特的网址格式
  - 也通过成熟的API解决：HTML5历史记录API，jQuery BBQ：后退按钮和查询库，PathJS库…

AJAX：结论

- 安全=主要问题：
  - 需要附加的浏览器防火墙（例如Google Chrome）
  - 跨站点漏洞：XSS，CSRF
  - 恶意客户
  - 浏览器和沙箱实施错误
- 现在成熟的技术
  - 反应灵敏，提供良好的交互性
  - 用于在Web平台上实施应用程序
  - 大量可用的框架来支持其部署
    - 例如GWT SDK，用于开发基于浏览器的Web应用程序
- 主要优势：它可以部署到任何地方