Advanced web technology

高级Web技术

web核心概念与协议

Web的基本概念

Web是分布在全世界的基于HTTP通信协议的服务器(Web服务器)中所有互相连接的超文本集(Web文档)，它采用浏览器/服务器模式并使用超文本技术链接Internet上的信息和资源。服务器端存放用HTML编写的网页以及其他资源，客户机通过浏览器 可以访问全球范围内各个主机上的这些信息资源。

- 基本特征
  - URI标识资源
  - 基于B/S模式
  - 使用HTML技术来创建客户端界面
  - 使用HTTP协议来传输内容

Web的诞生

Berners-Lee提出了超文本(Hyper Text)的概念并创造出超文本标记语言(Hyper Text Markup Language，简称HTML),同时开发出传输这种语言的协议HTTP。

Web的主要目的是旨在成为一种共享的信息空间(a shared information space)，人们和机器都可以通过它来进行沟通。

-Berners-Lee

Web的特征

- **Web是一种分布式超媒体系统**
  - 超文本/超媒体/链接
- **Web是图形化的和易于导航的**
  - 只需从一个链接跳到另一个链接就可在各页各站点间进行浏览
- **Web与平台无关**
  - 从Windows、UNIX、Macintosh等都可以访问Web
- **Web是分布式的**
  - 从用户来看这些信息是统一视图的
- **Web 是动态的**
  - 信息的提供者可以经常的对Web上的信息进行更新
- **Web是交互的**
  - 用户通过填写FORM表单可以向服务器提交请求，服务器返回相应信息

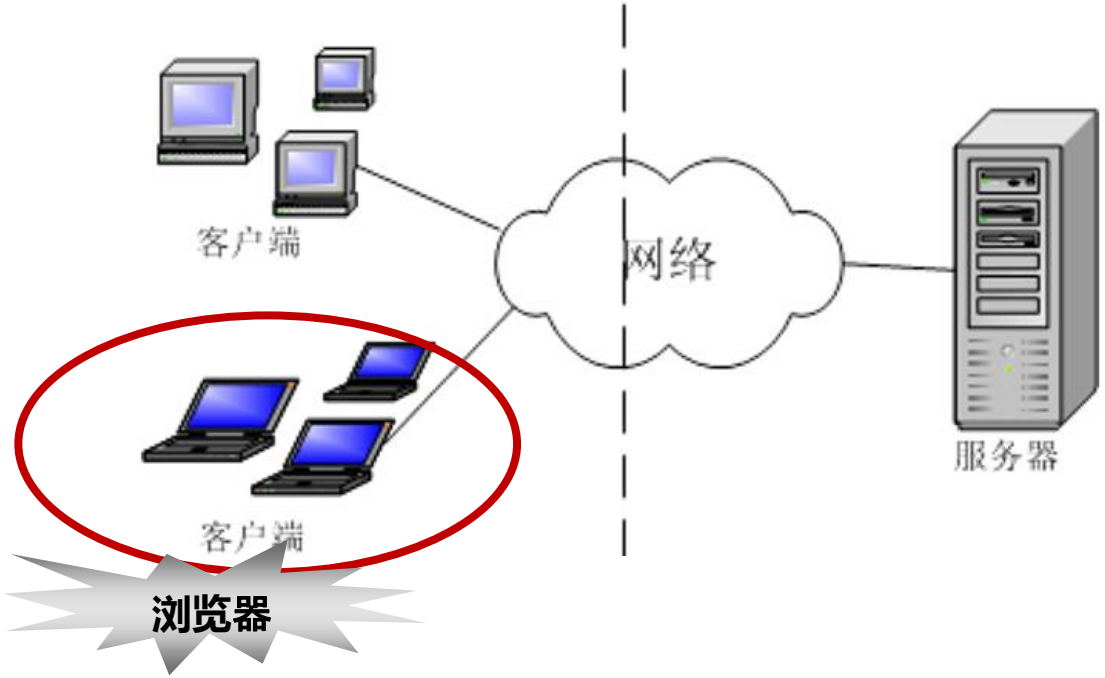
URI

- URI:统一资源标识符（Uniform Resource Identifier）
  - 抽象的，高层次；而URL和URN都是一种URI
- URL:统一资源定位符（Uniform Resource Locator ）
  - **用来确定Web网上某资源地址的字符串**
  - **语法为：**

    scheme://host:port/path/resource#section?parameters
  - **例如：http://jpkc.fudan.edu.cn/s/215/**

C/S系统与B/S

- C/S： 客户/服务器
- B/S: 浏览器/服务器



B/S - 瘦客户端

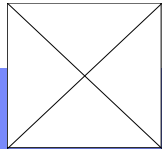
- 基本含义
  - 在客户/服务器的应用中，被设计得很小以至于大多数的数据操作均在服务器端进行的客户称为瘦客户
- 优点
  - 很容易部署
  - 很容易使用
  - 通过集中管理使系统管理更容易
    - 通过集中管理和监督可以很容易地发现问题
    - 在服务器端进行问题的解决
    - 新版本的软件只需安装在服务器上
  - 因为复杂的处理在服务器端进行，所以瘦客户使用的客户端资源很少

C/S - 胖客户端

- 基本含义
  - 在本地执行大多数的数据处理，只有数据本身存储在服务器上的客户称为胖客户
- 优点
  - 提供给终端用户更多的功能，使得终端用户根据自己的需要配置应用程序，因为胖客户机可以存储客户系统的大部分个人数据；
  - 可以减少服务器的负担，因为复杂的计算操作是由客户端自己完成
- 缺陷
  - 需要更多的可能导致错误的安装过程
  - 使用起来需要教育培训
  - 对新版本的软件需要重新安装
  - 需要较多的客户端资源（如内存和CPU处理能力）

C/S vs. B/S

- C/S的缺点
  - 系统整合性差
  - 配置和维护成本高
  - 对客户机要求高
  - 用户培训时间长
  - 伸缩性差
  - 软件复用性差
- C/S的优点:
  - 交互性强
  - 性能
  - 网络负载
  - 安全
  - 用户状态的维护



C/S vs. B/S

- **Web架构较C/S架构的优点**
  - 标准化
  - 开发代价低
  - 客户端“零花费”发布
  - 升级容易
  - 可以穿透防火墙
  - 易于在异构平台上配置集成
  - 降低客户培训费用
  - ……
- **Web架构较C/S架构的缺点**
  - 界面开发不如C/S方便
  - 速度慢,难以满足实时系统要求

HTML基础

- HTML一种含有一套语法规则的文本标记语言，用来表示网页。是后缀为(.htm,.html)的文本文件。
- 标签(TAGS)
  - HTML基本结构是由标签来标识的，每个标签表示对文挡的一种处理。
  - HTML标签的语法一般由“<元素>”引出，以“</元素>”作结尾。

```
<HTML>
    <HEAD>
        <TITLE> ..... </TITLE>
    </HEAD>
    <BODY>
        ..........
    </BODY>
</HTML>
```

HTML基础

- HTML5
  - 2014年最终定稿的最新HTML标准
    - **W3c Working Draft: http://www.w3.org/TR/html5/**
  - 促进了web上的和便携式设备的跨平台应用的开发。
  - 浏览器支持程度评测网站： http://html5test.com/

HTML基础

- **HTML5新增标签：增强语义**

article：文章

aside：内容

audio：音频

canvas：2D

command：

datalist：下

details：对象

dialog：对话

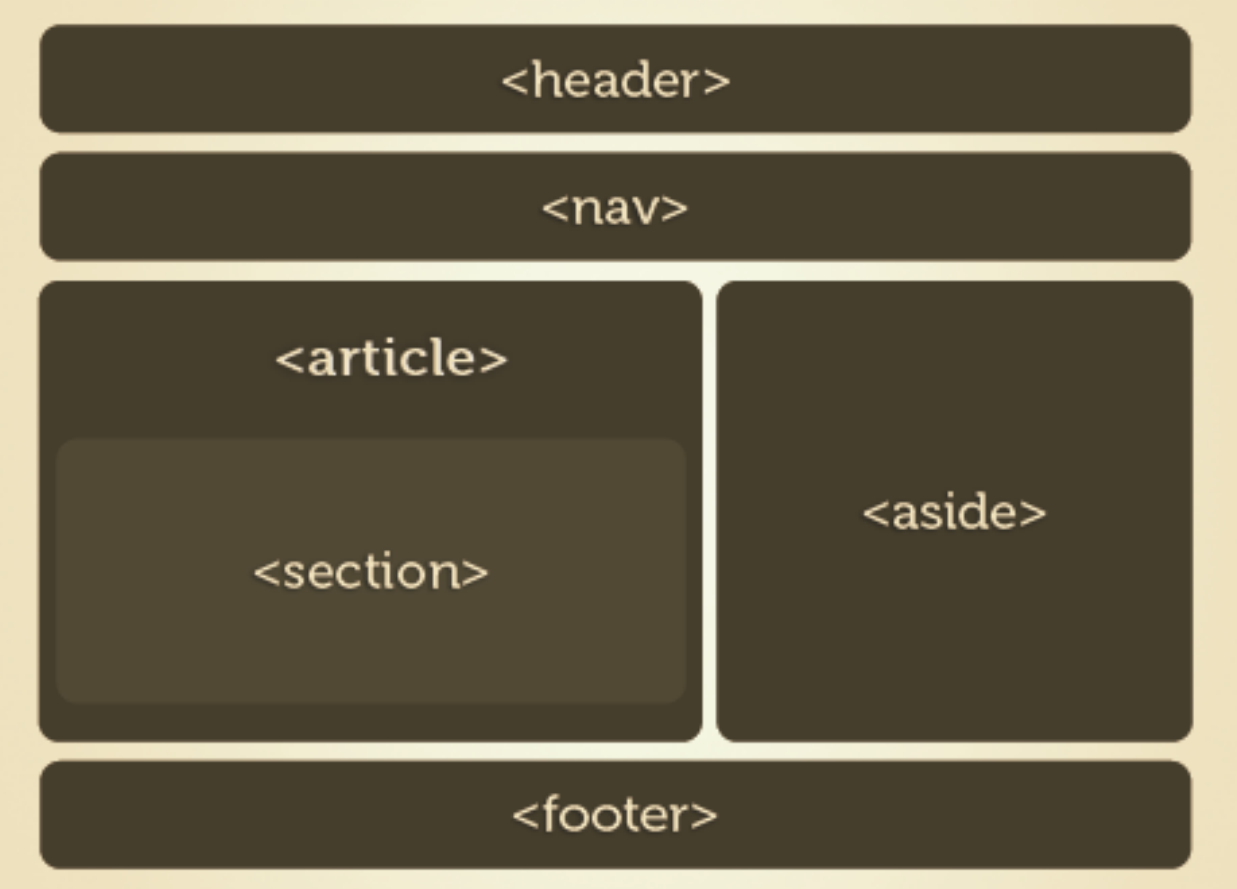
embed：外部

figure：一组

footer：页脚

header：页首

hgroup：文档某一部分的信息



time：日期的时间

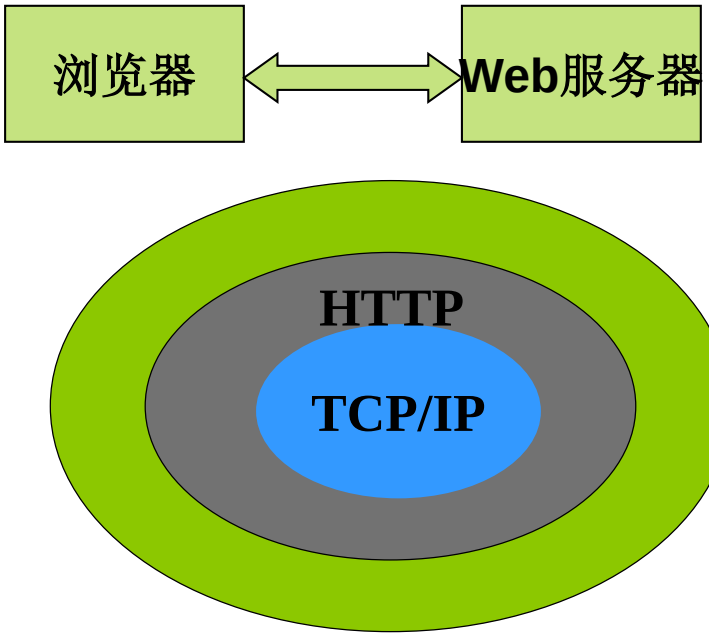
video：视频

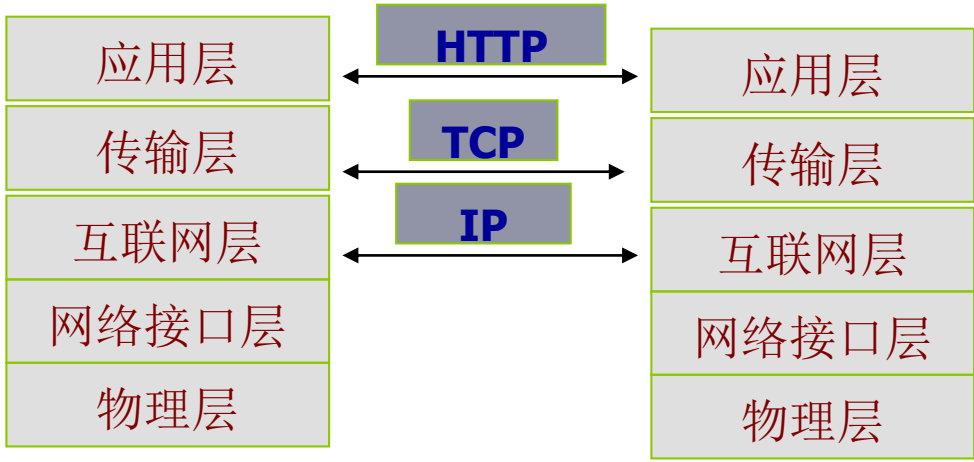
HTTP协议基础

- **HTTP协议**
  - HTTP代表HyperText Transport Protocol（超文本传输协议）
  - **Web服务器和客户端浏览器之间通过HTTP协议进行通信**
  - HTTP的两个重要功能
    - **传输文件**
    - **实现动态交互应用**

HTTP协议基础

- **HTTP协议**
  - HTTP是一种以TCP/IP通信协议为基础的应用协议
  - 无状态协议：与客户端之前请求的信息不做记忆；Cookie机制





HTTP协议基础

HTTP/1.0 和 HTTP/1.1 支持的方法：

方法	说明	支持的 HTTP 协议版本
GET	获取资源	1.0、1.1
POST	传输实体主体	1.0、1.1
PUT	传输文件	1.0、1.1
HEAD	获得报文首部	1.0、1.1
DELETE	删除文件	1.0、1.1
OPTIONS	询问支持的方法	1.1
TRACE	追踪路径	1.1
CONNECT	要求用隧道协议连接代理	1.1
LINK	建立和资源之间的联系	1.0
UNLINE	断开连接关系	1.0

HTTP基础

- **HTTP请求由三个部分构成，分别是：**
  - **请求方法 URI 协议/版本**
  - **请求头（Request Header）**
  - **请求正文**

```
POST /sample.jsp HTTP/1.1
Accept: image/gif, image/jpeg, */*
Accept-Language: zh-cn
Connection: Keep-Alive
Host: localhost
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Accept-Encoding: gzip, deflate

userName=kaiyu&password=1234
```

HTTP基础

和HTTP请求相似，也由三个部分构成：

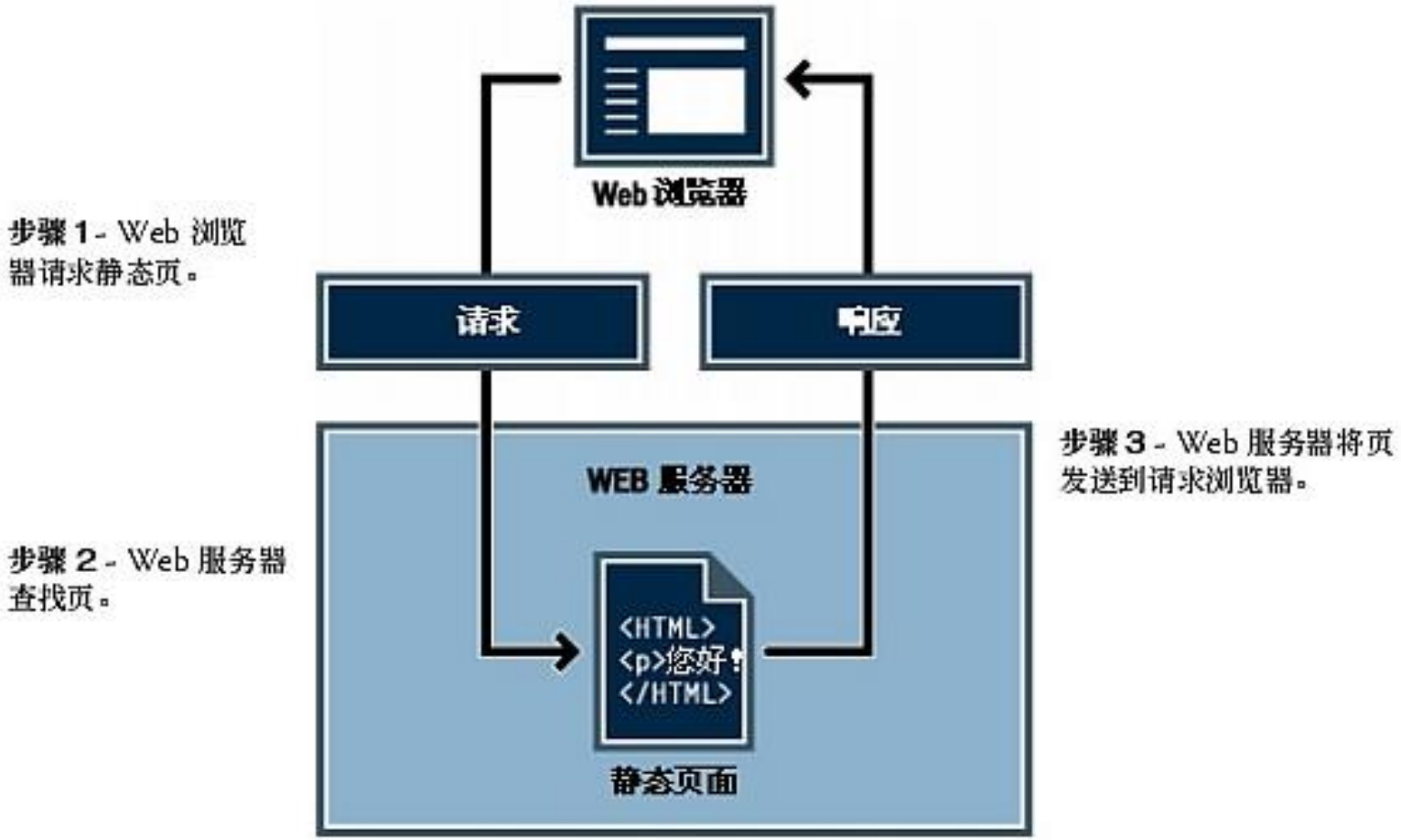
- 协议 状态代码 描述
- 响应头（Response Header）
- 响应正文

```
HTTP/1.1 200 OK
Server: ApacheTomcat/5.0.12
Date: Mon, 2 Oct 2017 13:13:33 GMT
Content-Type: text/html
Last-Modified: Mon, 2 Oct 2017 13:23:42 GMT
Content-Length: 112

<html>
<head>
<title>HTTP响应示例</title>
</head>
<body>
    Hello HTTP!
</body>
</html>
```

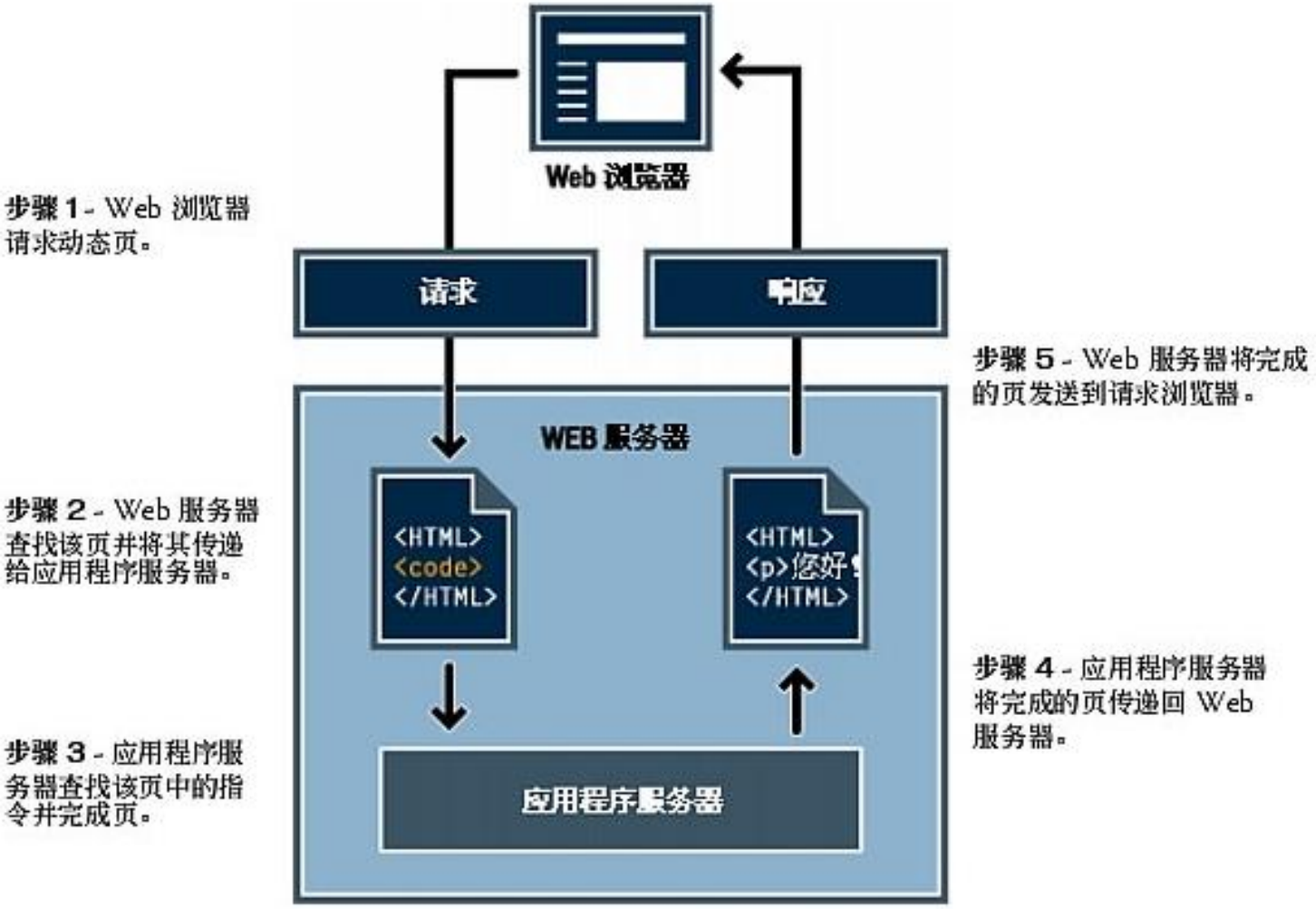
Web应用程序如何工作

- 处理静态 Web 页（文档Web）
  - 静态页是在发送到浏览器时不进行修改的页



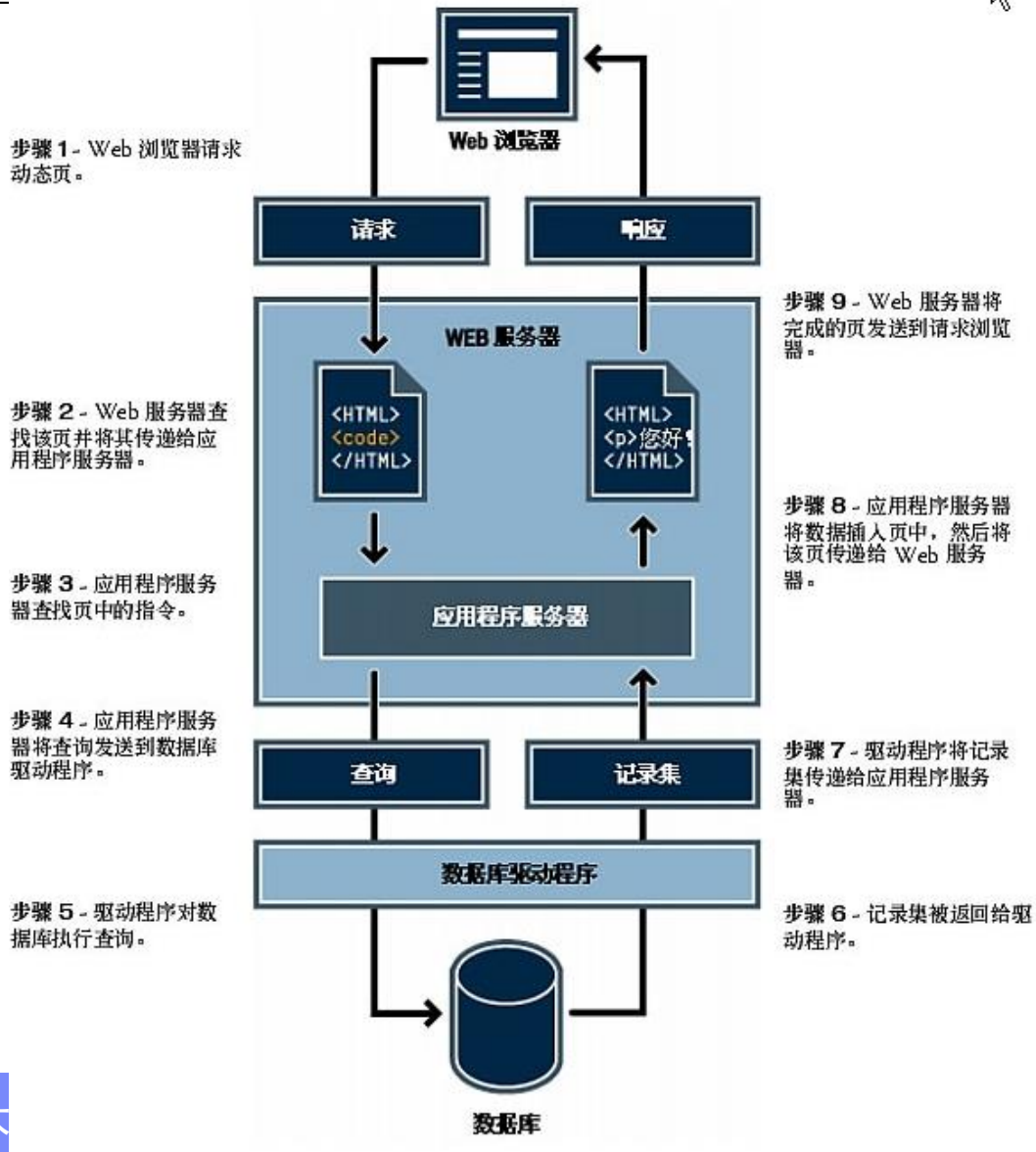
Web应用程序如何工作

- 处理动态 Web 页



Web应用程序如何工作

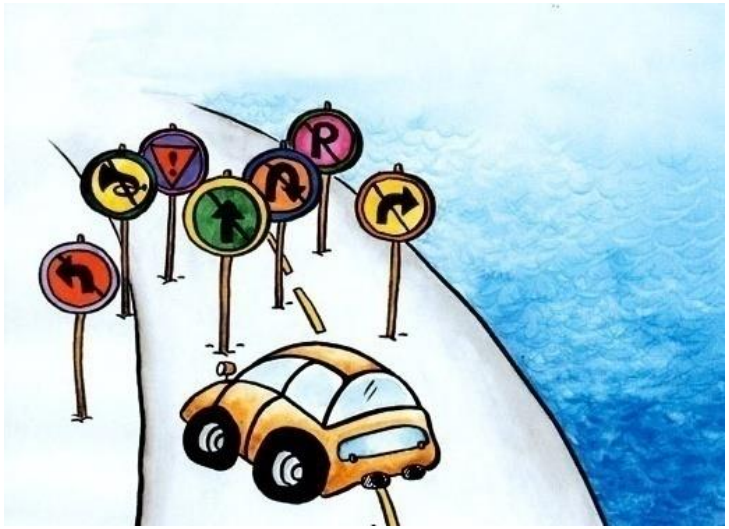
- 访问数据库



Rich Internet Application技术

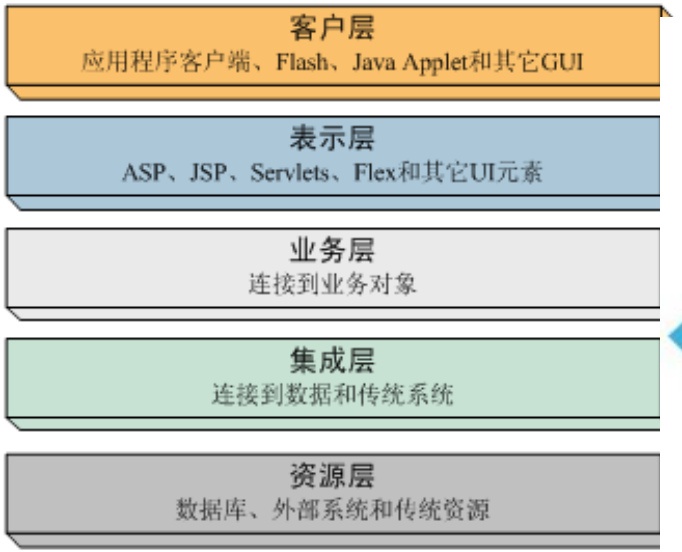
Web的不足

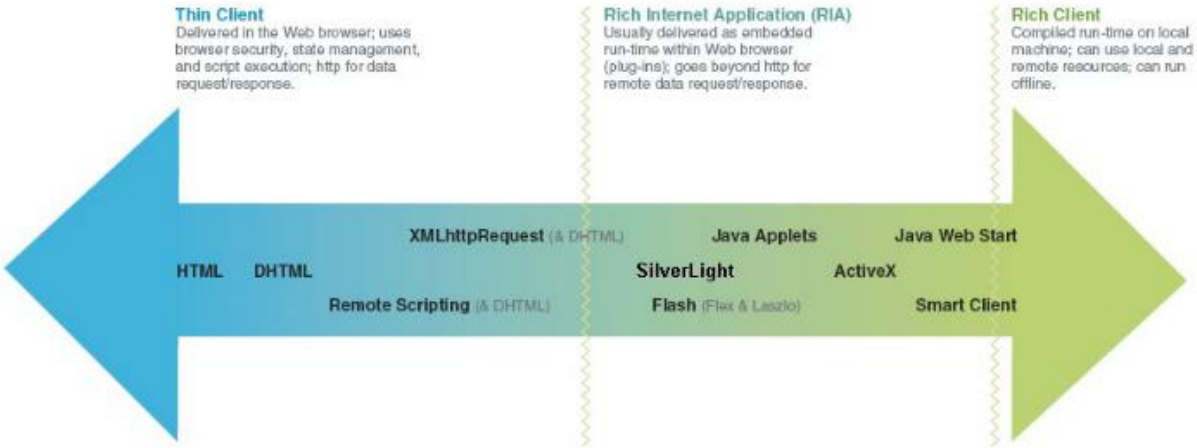
- 现在的**Web**应用程序对完成复杂应用方面却始终跟不上步伐
- **Web**模型是基于页面的模型，缺少客户端智能机制
- 几乎无法完成复杂的用户交互（如传统的**C/S**应用程序和桌面应用程序中的用户交互）



RIA的应用程序模型

- RIA中的 Rich Client（丰富客户端）提供可承载已编译客户端应用程序（以文件形式用HTTP传递）的运行环境，客户端应用程序使用异步客户/服务器架构连接现有的后端应用服务器。



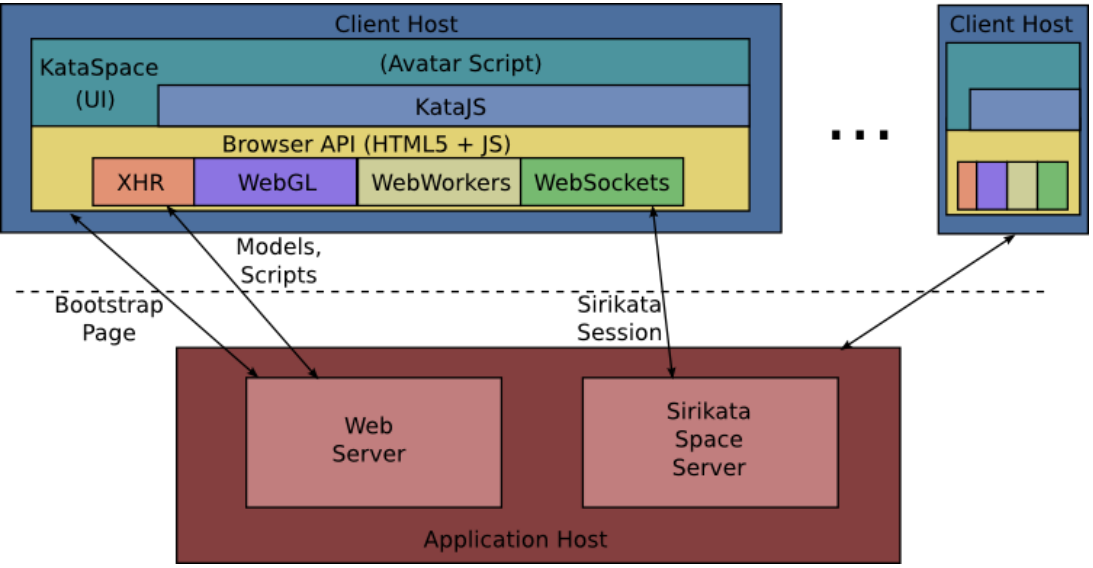


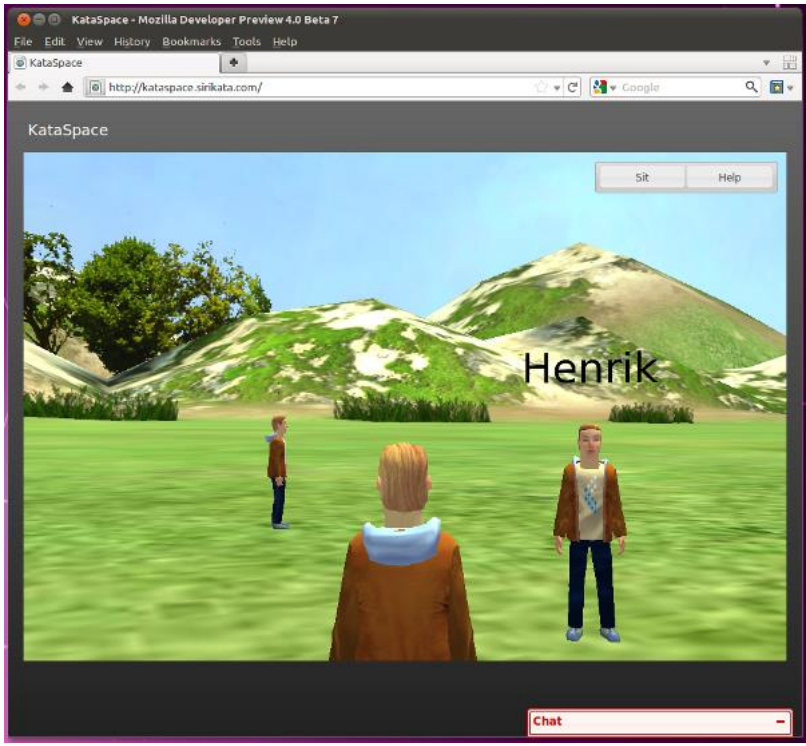
Rich Internet Application技术

- **RIA的优势**
  - *数据模型的丰富*
    - 用户界面可以显示和操作更为复杂的嵌入在客户端的数据模型
  - *用户界面的丰富*
    - 提供了灵活多样的界面控制元素，这些控制元素可以很好的与数据模型相结合
  - *最好的通讯模式*
    - 无刷新页面之下提供快捷的界面响应时间
    - 双向互动声音和图像。

Rich Internet Application技术

- **RIA应用示例**
  - **基于WebGL的NVE**
  - Kataspace: browser-based virtual worlds built with WebGL and HTML5





Rich Internet Application技术

- **Ajax**

adaptive path

blog | about us | services | events | **publications** | products | contact | home

publications

reports | latest essay | **essay archives** | newsletter | reading list

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

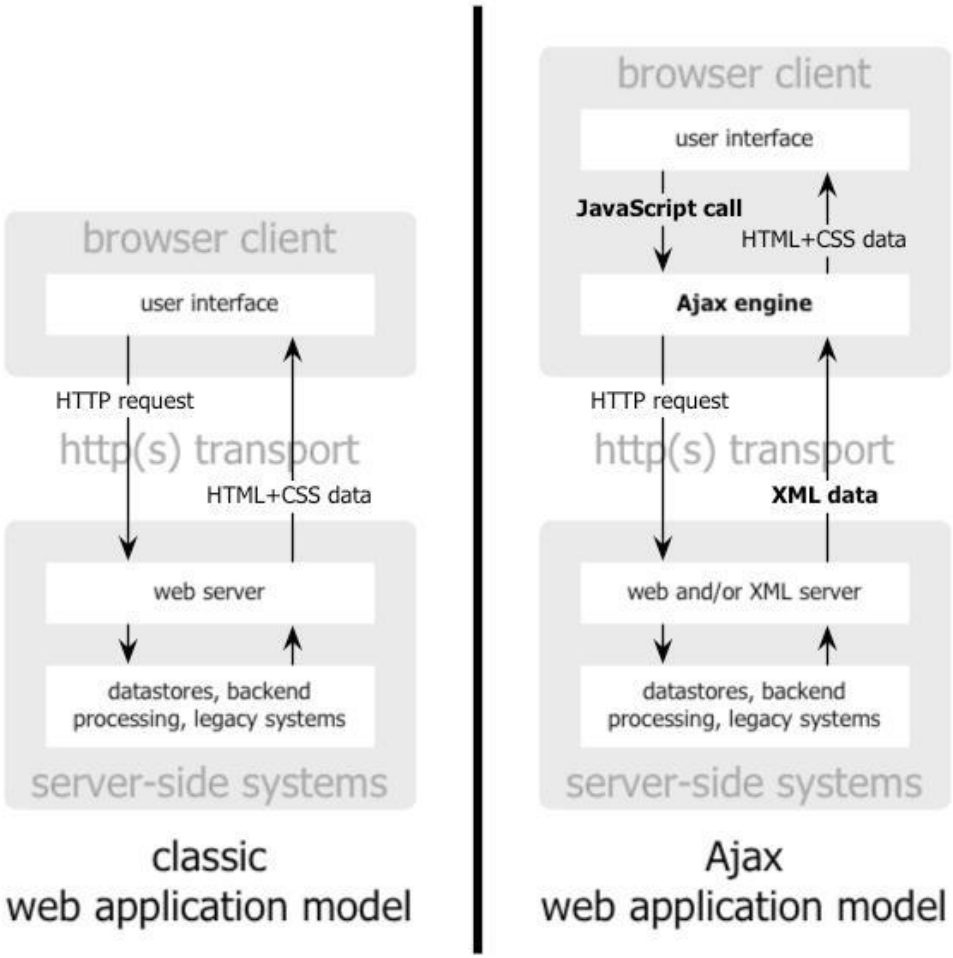
Tagging vs. Cataloging: What It's All About
November 30, 2006

Organizing Your Global Corporate Intranet
November 10, 2006

Essay Archives »

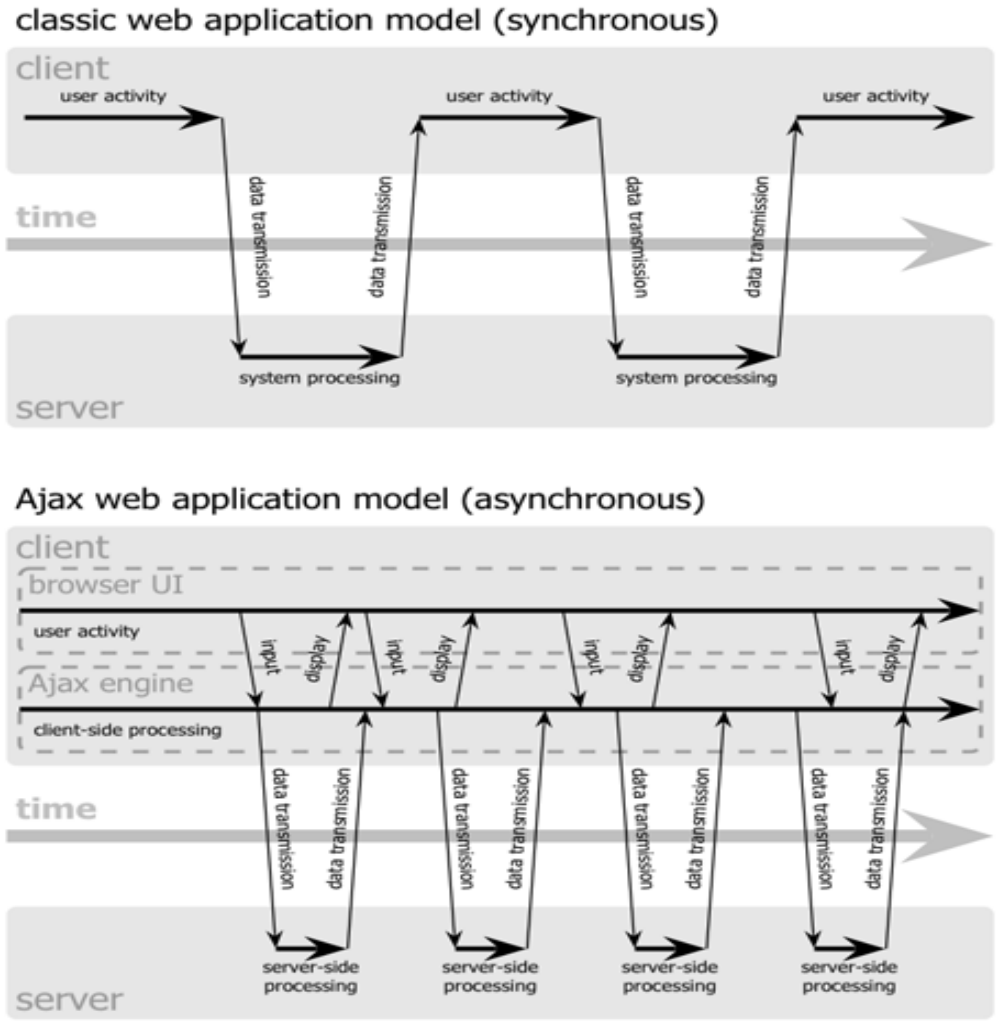
Rich Internet Application技术

- **Ajax**



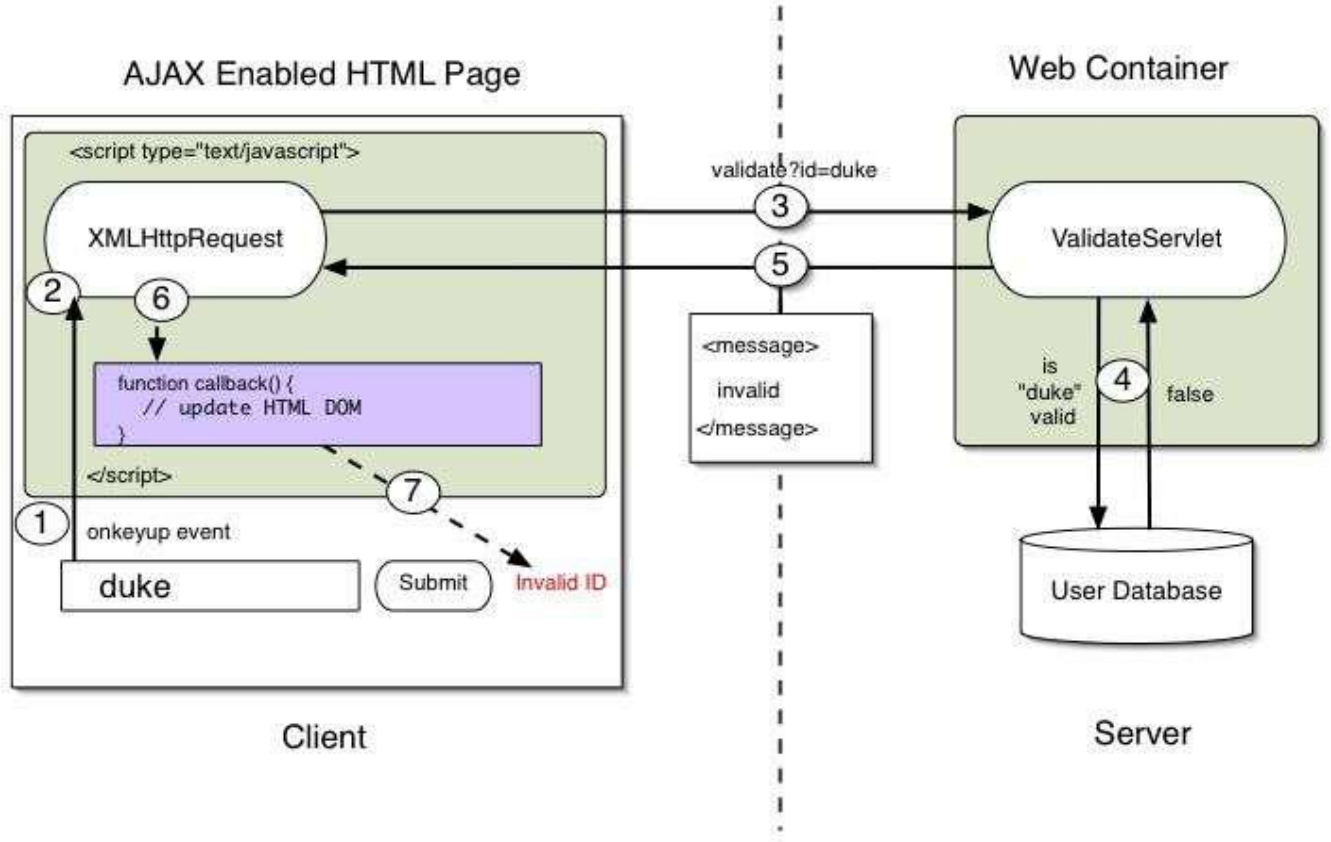
Rich Internet Application技术

- **Ajax**



Rich Internet Application技术

- **Ajax**



Rich Internet Application技术

- **Ajax**

1. 发生一个客户端事件

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

Rich Internet Application技术

- **Ajax**

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);
}
```

Rich Internet Application技术

- **Ajax**

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);
}
```

Rich Internet Application技术

- **Ajax**

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被设为validate?id=greg

Rich Internet Application技术

- **Ajax**

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>");
    }
  }
```

Rich Internet Application技术

- **Ajax**

6. XMLHttpRequest对象调用回调函数并处理结果

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

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

Rich Internet Application技术

- **Ajax**

7. 更新HTML DOM

```
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>
```

Rich Internet Application技术

- **Ajax**

使用Jquery

```
$.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.

Rich Internet Application技术

- **Ajax**

使用Jquery

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);
  });
});
```

Rich Internet Application技术

- **Ajax**

使用Jquery

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"); }
);
```

HTML5关键技术

- **Web Storage**
  - 简介
    - **提供了一种比cookie(4K)更加方便地本地存储机制,一般能存储最小5MB**
    - **sessionStorage（生命周期为页面会话期间）和localStorage（没有时间限制）两种机制。**

`Storage.length` 只读

返回一个整数，表示存储在 `Storage` 对象中的数据项数量。

`Storage.key()`

该方法接受一个数值 n 作为参数，并返回存储中的第 n 个键名。

`Storage.getItem()`

该方法接受一个键名作为参数，返回键名对应的值。

`Storage.setItem()`

该方法接受一个键名和值作为参数，将会把键值对添加到存储中，如果键名存在，则更新其对应的值。

`Storage.removeItem()`

该方法接受一个键名作为参数，并把该键名从存储中删除。

`Storage.clear()`

调用该方法会清空存储中的所有键名。

HTML5关键技术

- **Web Socket**
  - **实现服务器端推送的方法**

传统轮询
- 浏览器定时发送数据请求
- 服务器查找数据
- 返回数据给客户端

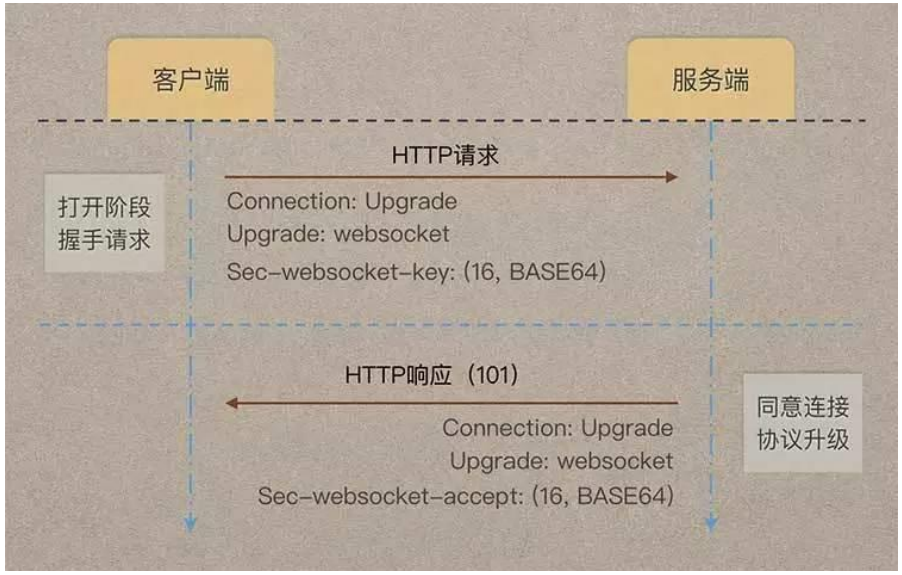
长轮询
- 浏览器发送数据请求，
- 服务器持有该请求，等待新数据
- 返回数据给客户端

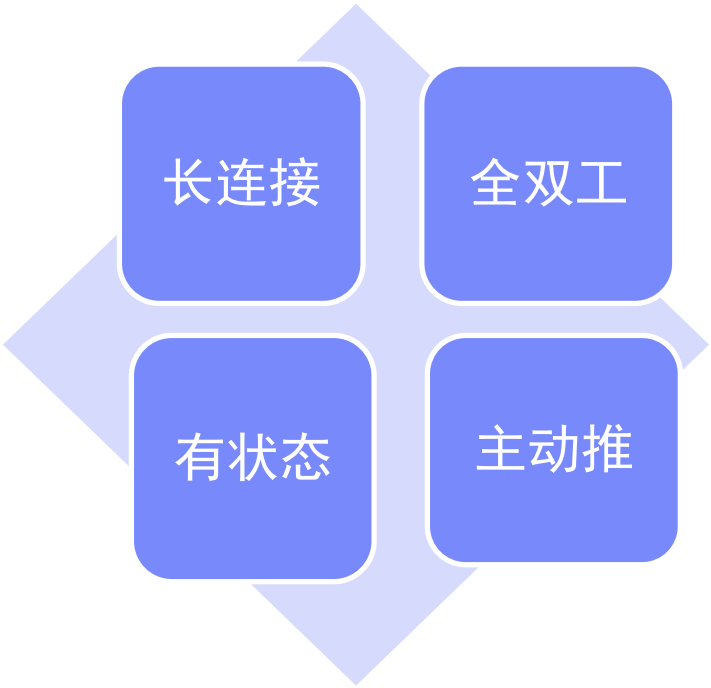
长连接
- 浏览器嵌入一个隐藏iframe，并设置src属性设为对一个长连接的请求
- 服务器端发现新数据时告知浏览器
- 浏览器发起新数据请求

WebSocket
- 浏览器发起连接请求
- 服务器与客户端建立连接
- 服务器推送数据
- 客户端接收展示

HTML5关键技术

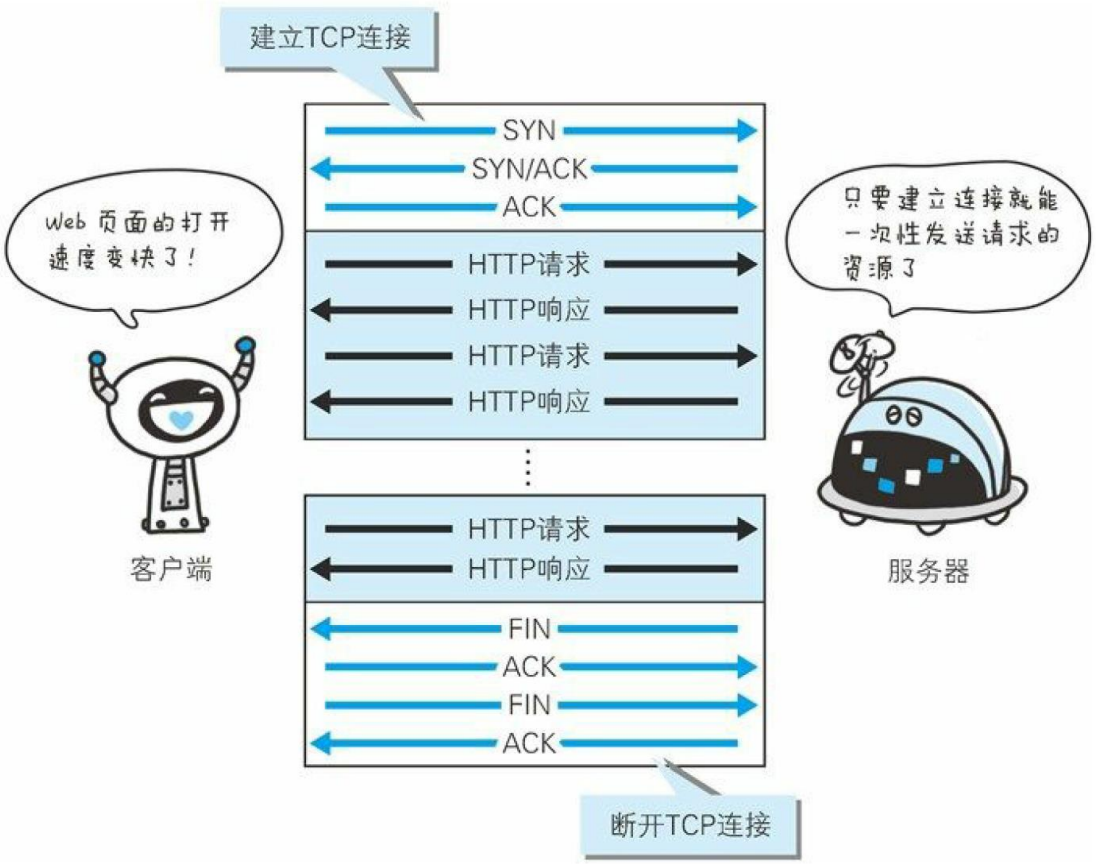
- **Web Socket**
  - **基于TCP，实现了浏览器与服务器全双工通信，即允许服务器主动发送信息给客户端。**





HTML5关键技术

- **Web Socket**
  - 实现服务器端推送的方法



引自《图解http》

HTML5关键技术

- **Web Socket**
  - 简介

```
var websocket;

function login() {

    var wsUri = "ws://" + window.location.host + "/wsockdemo/chat/"  + nameField.value;

    websocket = new WebSocket(wsUri);

    websocket.onopen = function(evt) {
        printMessage("CONNECTED");
    };

    websocket.onmessage = function(evt) {
        printMessage(evt.data);
    };

}

function sendMessage() {
    websocket.send(messField.value);
}
```

```
@ServerEndpoint("/chat/{user-name}")
public class Chat {
    private static final Set<Session> sessions = new HashSet<Session>();
    @OnOpen
    public void login(Session session, @PathParam("user-name") String userName) {
        session.getUserProperties().put("name", userName);
        sessions.add(session);
        for(Session curerent : sessions){
            if(session != curerent){
                try {
                    if(curerent.isOpen()){
                        curerent.getBasicRemote().sendText(userName+"上线了");
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

    }
    @OnMessage
    public String messge(String message, Session session) throws IOException {
        for (Session client : sessions) {
            if (!client.equals(session)) {
                client.getBasicRemote().sendText(session.getUserProperties().get("name") + ": " + message);
            }
        }
        return "I said: " + message;
    }
```

HTML5关键技术

- **Web Worker**
  - **特征**
    - **允许JavaScript创建多个线程，子线程完全受主线程控制**
    - **不得操作DOM，不能访问全局变量（window、document之类的浏览器全局变量）或是全局函数，不能调用alert()之类函数**
  - **两种web worker**
    - **专用线程dedicated web worker: 只能被创建它的页面访问,随当前页面的关闭而结束；**
    - **Shared web worker可以被多个页面访问**

HTML5关键技术

- **Web Worker**
  - dedicated **web worker**
    - **可以通过注册监听器和onMessage（onError）两种方式来进行主线程和子线程之间的通信。**

```
var worker = new Worker("script/lenthytask.js");
worker.addEventListener("message", function (evt) {
    alert(evt.data);
}, false);
worker.postMessage(10000)
```

主线程

```
addEventListener("message", function (evt) {
    var date = new Date();
    var currentDate = null;
    do {
        currentDate = new Date();
    } while (currentDate - date < evt.data);
    postMessage(currentDate);
}, false);
```

子线程

HTML5关键技术

- **Web Worker**
  - dedicated **web worker**
    - **postMessage(data)：用于子线程与主线程之间互相通信**
    - **terminate()：主线程中终止worker**
    - **onMessage:有消息时触发该事件, 消息内容可通过事件对象的data来获取**
    - **error：出错处理。错误信息：e.message，e.filename，e.lineno**

```
//创建一个Worker对象，并向它传递将在新线程中执行的脚本url
var worker = new Worker('worker.js');
//接收worker传递过来的数据
worker.onmessage = function(event){
    document.getElementById('result').innerHTML+=event.data+"<br/>" ;
};
```

```
//worker = new Worker('url');
//worker.postMessage传递给子线程数据，对象
worker.postMessage({first:1,second:2});

//子线程中也可以使用postMessage，如传递字符串
postMessage('test');
```

HTML5关键技术

- **Web Worker**
  - **web worker的Ajax调用**
    - **可以使用XMLHttpRequest与服务端通信**
    - **适合MVVM模式**

```
addEventListener("message", function (evt) {
    var xhr = new XMLHttpRequest();
    //访问天气预报的API
    xhr.open("GET", "http://www.help.bj.cn/apis/weather?id=101060101");
    xhr.onload = function () {
        postMessage(xhr.responseText);
    };
    xhr.send();
}, false);
```

HTML5关键技术

- **Web Worker**
  - **web worker的Ajax调用**
    - **可以使用XMLHttpRequest与服务端通信**
    - **适合MVVM模式**

```
addEventListener("message", function (evt) {
    var xhr = new XMLHttpRequest();
    //访问天气预报的API
    xhr.open("GET", "http://www.help.bj.cn/apis/weather?id=101060101");
    xhr.onload = function () {
        postMessage(xhr.responseText);
    };
    xhr.send();
}, false);
```

HTML5关键技术

HTML5其他特征

- 对使用CSS3来管理GUI的支持，这意味着HTML5可以是面向内容的
- 使用 Web SQL Databases 支持SQL规范的本地数据库
- 画布Canvas(支持JS在之上绘图)和音视频支持（audio**video**），可在无需安装第三方插件的情况下添加图形和视频
- Geolocation API规范，支持共享地理位置；通过使用智能手机定位功能来纳入移动云服务和应用

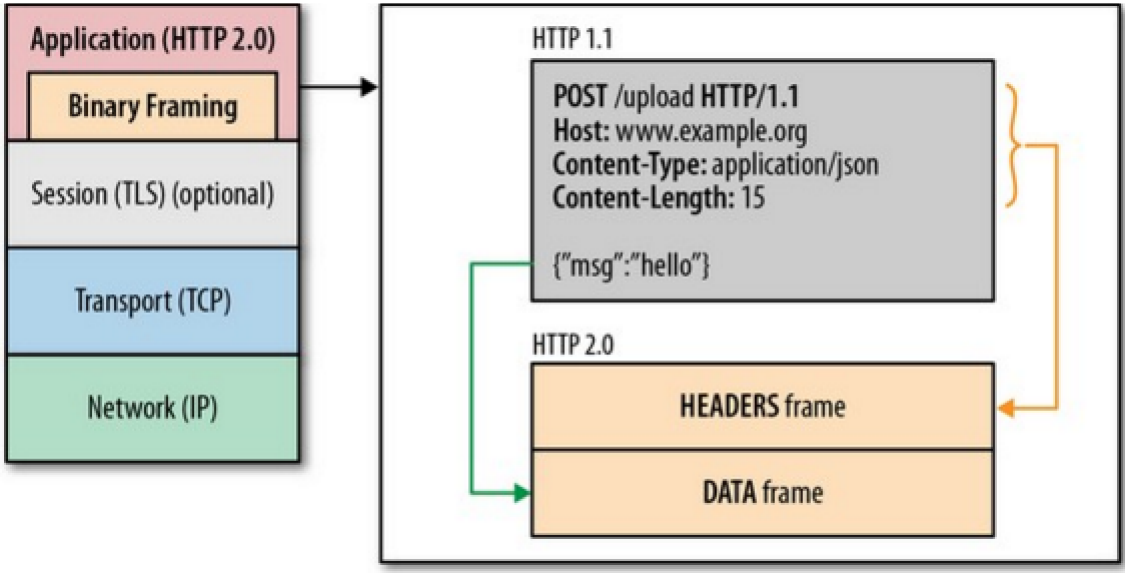
HTML5关键技术

HTML5其他特征

- 增强型的表单，其降低了下载JavaScript代码的这种必要性，允许在移动设备和云服务之间进行更多高效的通信。
- 作为HTML5中的 <canvas>标签的一个特殊的上下文（experimental-webgl）实现在浏览器中的 WebGL
- 通过ApplicationCache接口使离线存储成为可能，离线存储使得你的web应用可以在用户离线的状况下进行访问
- 使用**Notifications api 桌面提醒接口**支持**Web通知**；提供了与本地文件交互的标准方法：File API规范；支持拖曳（Drag & Drop）

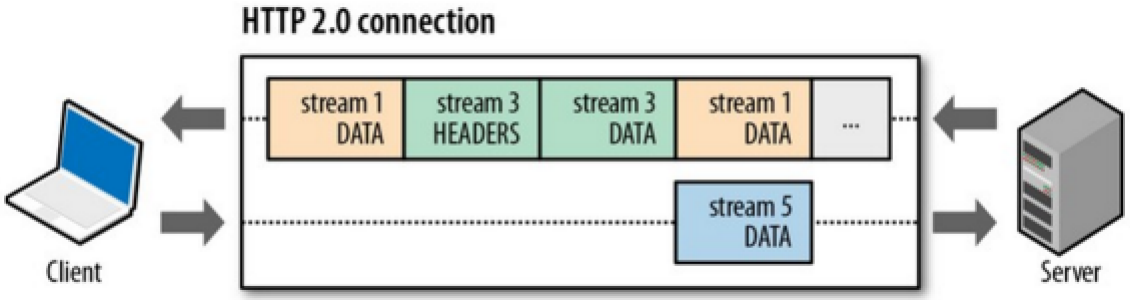
HTTP2.0

- **性能增强：** 不改动HTTP 的语义，改进传输性能，实现低延迟和高吞吐量
  - **二进制分帧**
- **HTTP2.0 首部压缩**
  - 使用“首部表”来跟踪和存储之前发送的键-值对，不发送重复数据。
- **HTTP2.0的请求都在一个TCP链接上**



HTTP2.0

- **并行双向字节流的请求和响应**
  - 把 HTTP 消息分解为独立的帧,交错发送,然后在另一端重新组装

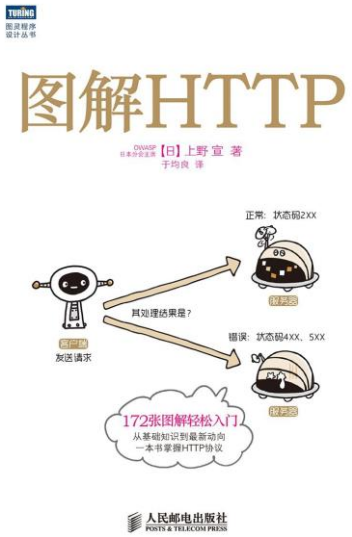


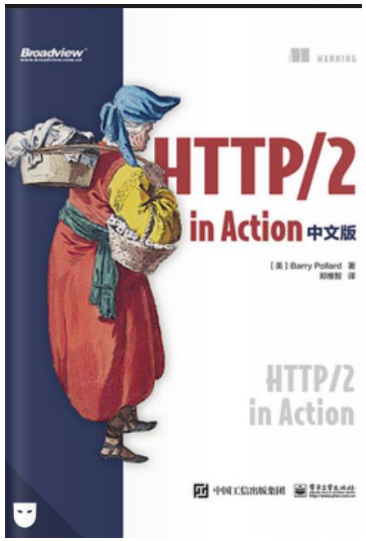
- **HTTP2.0的请求优先级**
- **HTTP2.0的服务器推送**
  - 就是服务器可以对一个客户端请求发送多个响应。

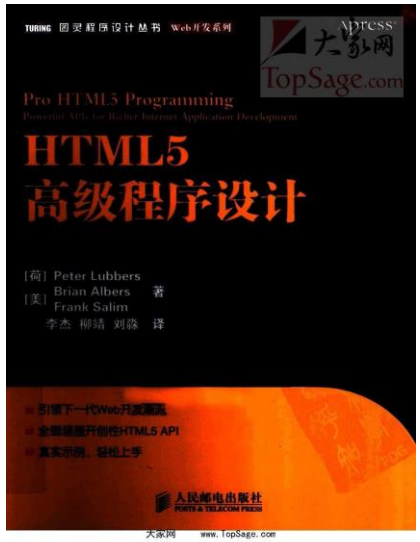
HTTPS&QUIC& SPDY

- **HTTPS：**
  - **使用TLS/SSL协议用于对HTTP协议传输的数据进行加密。**
    - 非对称加密算法用于在握手过程中加密生成的密码，对称加密算法用于对真正传输的数据进行加密，而HASH算法用于验证数据的完整性。
- **QUIC（ Quick UDP Internet Connections ）**
  - **谷歌公司研制的结合TCP和UDP的一种UDP通信的改进版。**
- **SPDY**
  - **谷歌研制的提升HTTP速度的协议，是HTTP/2.0的基础。**

参考书


图解HTTP
【日】上野宣 著
正常：状态码2XX
其处理结果是？
错误：状态码4XX、5XX
友区请求
172张图解轻松入门
从基础知识到最新动向
一本书掌握HTTP协议
人民邮电出版社


Broadview
HTTP/2
in Action 中文版
HTTP/2
in Action


Apress
Pro HTML5 Programming
HTML5
高级程序设计
[荷] Peter Lubbers
[美] Brian Albers
Frank Salim
著
李杰 柳靖 刘淼 译
人民邮电出版社
TopSage.com