电子商务技术基础

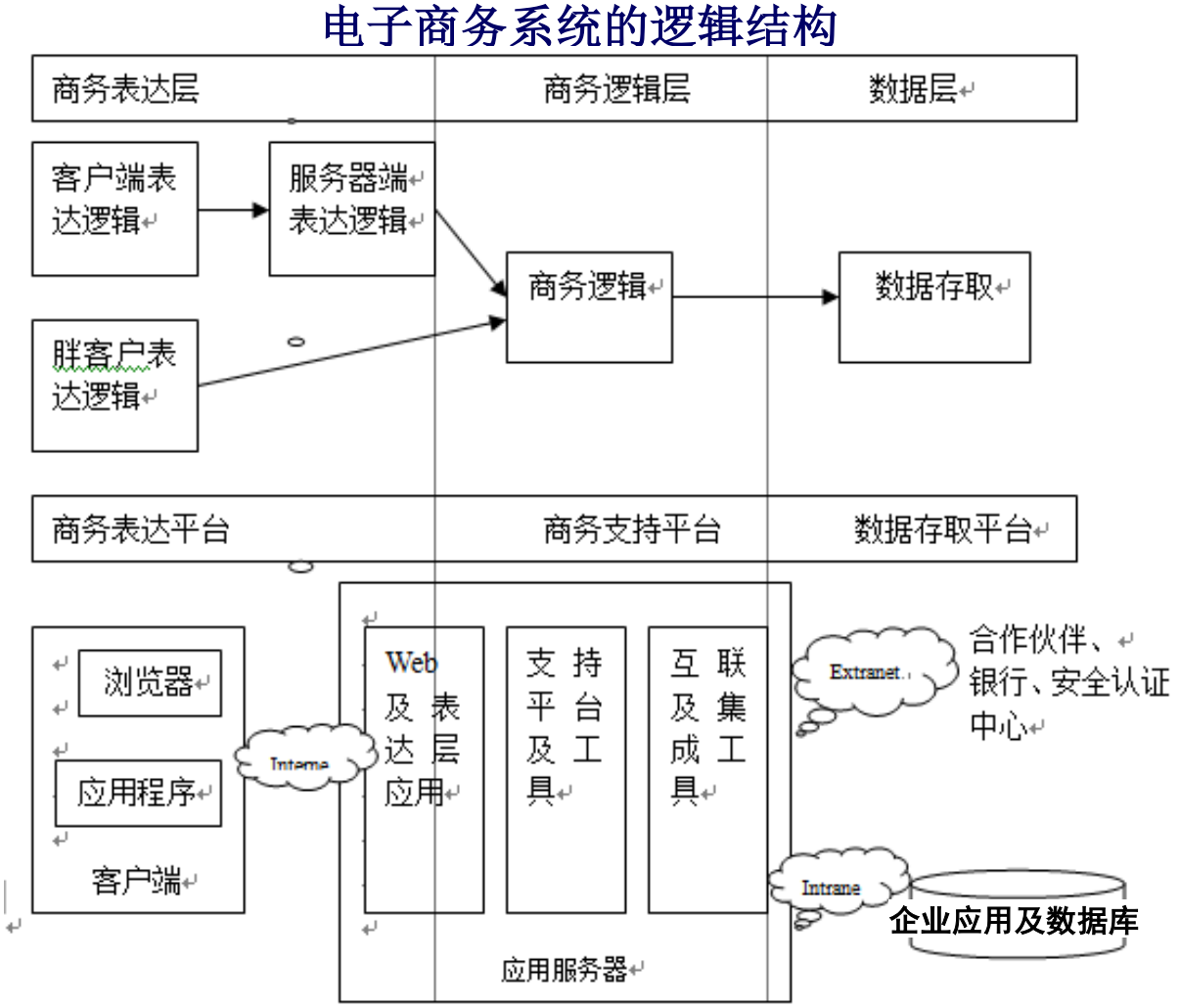
第二章 商务表达层及其技术

第二章 商务表达层及其技术

- **第一节 商务表达层的功能与实现**
- 第二节 静态网页的表达及其技术
- 第三节 标记语言
- 第四节 动态网页与客户端脚本
- 第五节 服务器脚本

第一节 商务表达层的功能与实现

电子商务系统的逻辑结构



第一节 商务表达层的功能与实现

1、商务表达层的任务

- 主要任务是表现商务数据，即通过合适的技术和方法将系统中的商务数据在用户面前表现出来，涉及的范围仅局限于客户端和Web服务器端。
- 商务表达层仅涉及客户端和Web服务器端的应用程序。

第一节 商务表达层的功能与实现

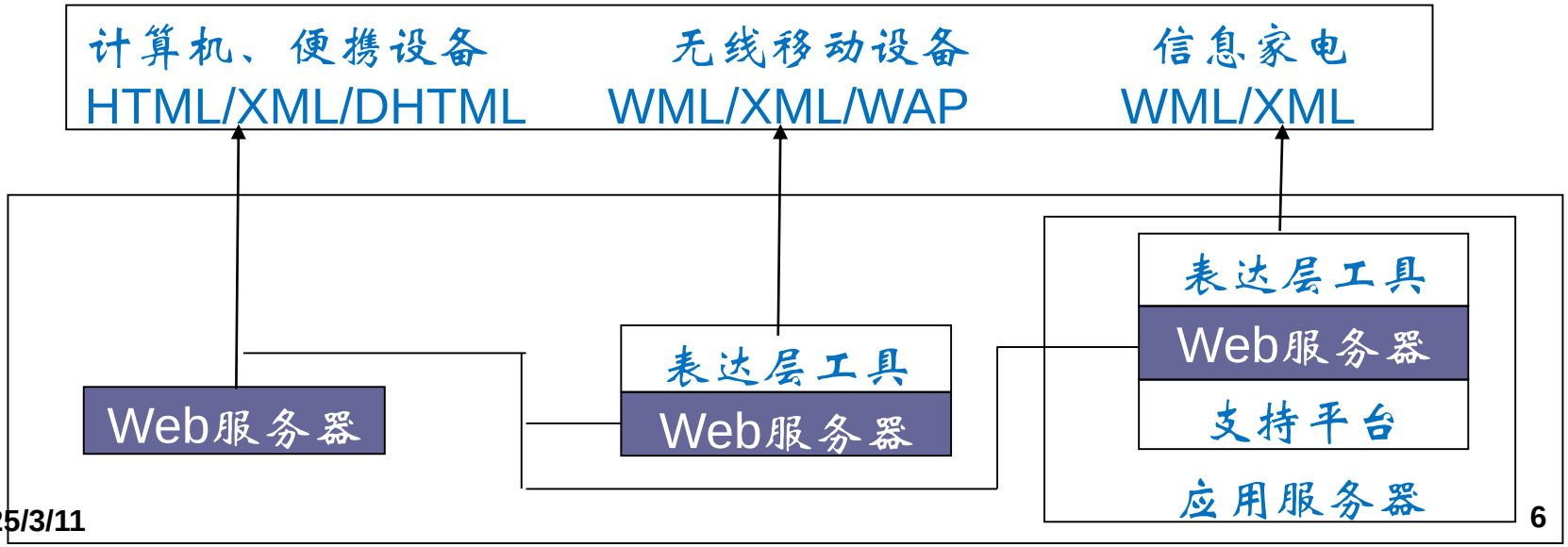
2、商务表达层的功能

- **客户端的应用程序**：表现数据，要求具有高度易用性、通用性和灵活性，具有友好的用户界面并能和用户进行数据交互。
- **Web服务器端的应用程序**：要求能够发布商务信息，并能与客户端实现双向信息交互。
- **功能**：商务表达层是“客户端独立的”，它只为最终用户提供了一个用户界面，在此界面中接受用户提交的事件，并将处理结果返回给用户。当用户页面的格局和风格发生变化时，对业务逻辑层和数据连接层不会产生影响。

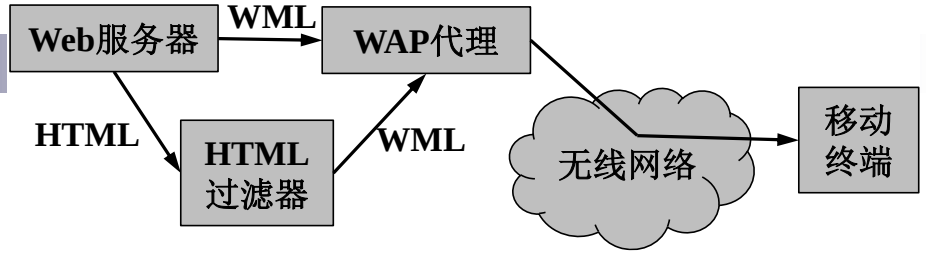
第一节 商务表达层的功能与实现

3、商务表达平台的实现

- 商务表达平台的三种实现方式
  - 基于Web，支持以HTML为主的表达方式
  - 在Web基础上增加表达层工具，扩展Web的表达功能
  - 使用应用服务器，实现分布式的系统功能



第一节 商务表达层的功能与实现



3、商务表达平台的实现

- **Web支持的以HTML为主的表达方式**：以Web服务器为基础，无需额外配置且容易实现，但只能支持HTML和XML客户端，无法直接支持WML标准的设备，应用范围稍窄。
- **扩展的Web表达方式**：在前者的基础上增加支持多种客户端的软件和硬件，扩展Web的功能，使其能支持除HTML与XML以外的其他数据表达方式（如无线应用协议WAP、多媒体邮件类型扩展协议MIME等）。好处在于可以有针对性地扩充Web的功能，使商务处理结果能够利用多种渠道由多种客户端表达。但由于需要在Web上增加独立的产品，增加了系统集成难度。

第一节 商务表达层的功能与实现

3、商务表达平台的实现

- **利用应用服务器完成表达层功能**：利用应用服务器将数据表达层的功能和Web服务器紧密地结合在一起，可以直接利用服务器来完成表达层的功能。目前应用服务器支持多种客户端设备(如计算机、PDA、移动通信设备等)和多种协议标准(如HTML、WAP、XML等)。
  - 采用应用服务器来实现应用表达层的功能，好处在于集成难度小，表达部分和应用程序之间的接口比较容易实现。
  - 然而，由于目前各个IT厂商提供的应用服务器功能差异比较大，有些产品在表达层上有不足之处，但在其他方面可能功能较强；有的则可能在表达层次上实现很好，但是在商务服务层次则有缺点。所以开发者需要在这些方面进行取舍和选择。

第一节 商务表达层的功能与实现

4、商务表达层技术结构

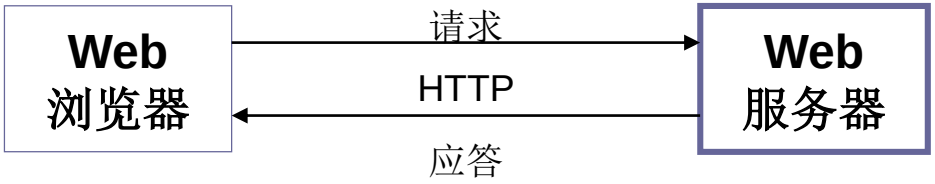
- 无论外部还是内部的电子商务系统，其信息的表达与组织通常都是由电子商务网站完成的。
- 网站主要完成电子商务表达层的逻辑任务。Web应用程序代表网站的主流技术。
- Web应用程序一般有4种基本体系结构：2层结构、3层结构以及两类4层结构。

第一节 商务表达层的功能与实现

4、商务表达层技术结构

2层体系结构

- **由Web浏览器与Web服务器两部分组成。**
  - Web浏览器与Web服务器之间的通信遵循**HTTP协议**。
  - Web浏览器向Web服务器请求文档，Web服务器则根据该请求返回相应的文档；如果该文档不存在，Web服务器将返回错误提示信息。



第一节 商务表达层的功能与实现

4、商务表达层技术结构

2层体系结构

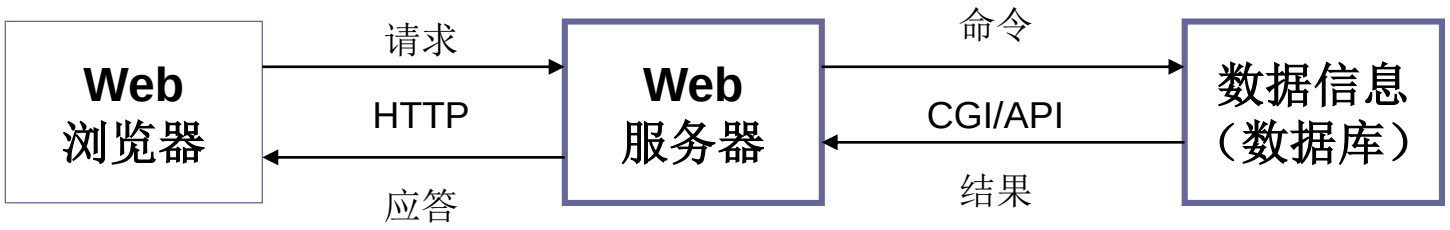
- 由Web浏览器与Web服务器两部分组成。
- Web服务器中的文档是事先写好的，任何用户请求文档时返回的结果都一样，因此被称为“静态网站”。
- 典型例子：
  - 对某些专题的FAQs(Frequently Ask Questions)
  - 一些软件的HTML帮助系统(如Apache Web Server)。

第一节 商务表达层的功能与实现

4、商务表达层技术结构

3层体系结构

- 由Web浏览器、Web服务器和数据信息（即数据库）3部分组成。
  - Web浏览器与Web服务器之间的通信遵循**HTTP协议**。
  - Web服务器与数据信息之间的通信遵循**CGI或者Server API规范**。



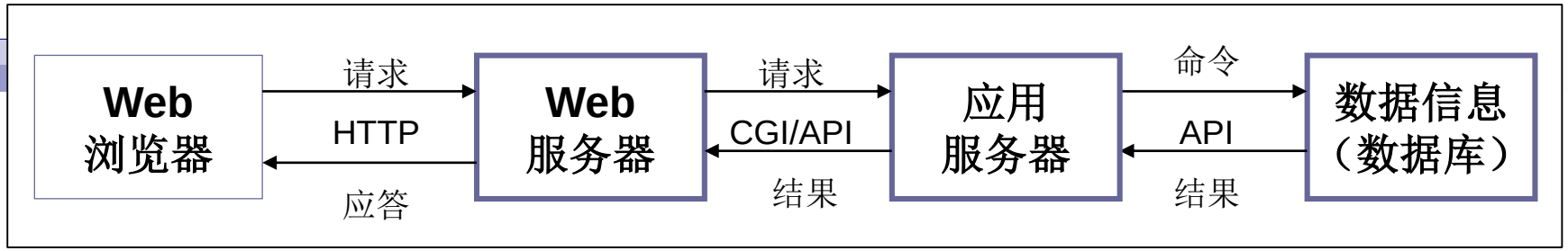
第一节 商务表达层的功能与实现

4、商务表达层技术结构

3层体系结构

- 如果Web浏览器请求的是某个HTML文档，Web服务器就返回该文档；如果Web浏览器请求的是某个CGI程序或者API程序，Web服务器则执行（或调用外部程序执行）该程序，然后将程序执行结果返回给Web浏览器。
- 通过CGI脚本或者Server API程序来扩展Web服务器的功能，是Web技术发展史上的一个里程碑。CGI脚本或Server API程序可以根据用户的不同请求，返回不同的结果，因此把采用3层体系结构的网站称为“动态网站”。
- 典型例子：在线搜索查询。

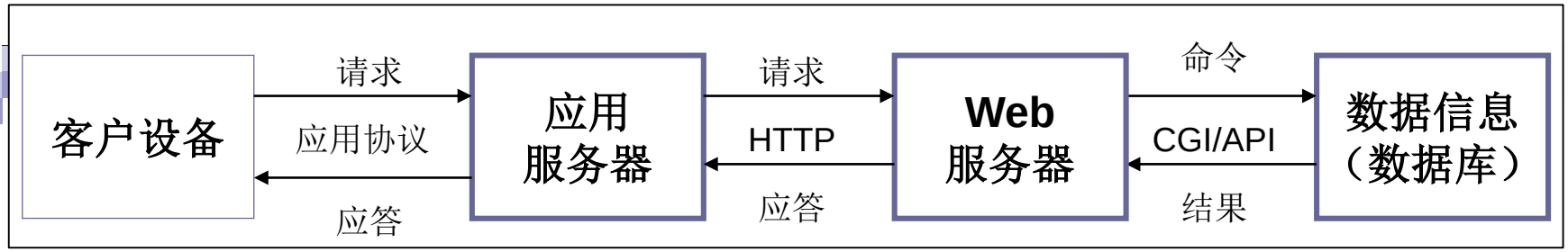
第一节 商务表达层的功能与实现



4层体系结构

- **第一类4层结构：由Web浏览器、Web服务器、应用服务器和数据信息4部分组成。**
- **Web浏览器向Web服务器提出请求，Web服务器分析请求：**
  - 若Web浏览器请求的是简单的HTML文档，Web服务器就返回相应的文档；
  - 若Web浏览器请求的是特殊文档，Web服务器就将该请求交给应用服务器执行，应用服务器根据请求，访问相应的数据信息，然后把执行结果返回给Web服务器，并通过Web服务器将结果以HTML的形式返回给Web浏览器。
- **典型例子：基于Web的MATLAB应用程序，实现动态多媒体教学和远程实验室应用。**

第一节 商务表达层的功能与实现



4层体系结构

- **第二类4层结构：由客户设备、应用服务器、Web服务器和数据信息4部分组成。**
  - Web浏览器被客户设备（手机、掌上电脑等）取代。
  - 客户设备与应用服务器之间的通信遵循专门的**通信协议**。
  - 应用服务器与Web服务器之间的通信仍然遵循**HTTP协议**。
  - Web服务器与数据信息之间的通信遵循**CGI或Server API规范**。
- **典型例子：WAP应用，客户设备是手机，应用服务器是WAP网关；手机与WAP网关的通信采用WAP协议；手机屏幕显示的文档由WAL实现。**

第一节 商务表达层的功能与实现

5、客户端的实现

- **客户端是电子商务系统的最终用户接口。**它包括两层含义，既是指具体的用户硬件设备，又是指用户硬件设备上所运行的应用程序（如浏览器等）。
- **从设备角度，客户端分成普通的计算机**（包括PC、工作站等）、**移动终端**（如手机、PDA、寻呼机等）、**其它信息终端**（如通过无线或有线方式与电子商务系统连接的家用电器、ATM取款机等）这三类。
- 在技术上，客户端要求这些设备能够支持标准的协议（HTML、WML、XML），支持连接诊断或在线升级，能够从网上下载插件，通过Internet能与电子商务系统相连接，并能进行交互。

第一节 商务表达层的功能与实现

5、客户端的实现

- 客户端的分类（从逻辑结构角度）
  - 胖客户端 — 具有数据处理功能的传统应用程序和需特定插件的多媒体程序，具备较强的数据处理功能，不需要服务器参与即可完成，可以减轻服务器的负担。
  - 瘦客户端 — 基于浏览器的程序，大多数的数据操作需要依赖服务器才能进行，用户界面风格统一、操作简单，管理灵活，在电子商务中比较常见。
  - 富客户端 — Microsoft的Ajax异步数据传送技术，将胖/瘦服务器优势集于一身，有很强的交互性。

第一节 商务表达层的功能与实现

5、客户端的实现

- 客户端显示内容
  - 静态内容：客户端上显示的内容预先放置在Web服务器上，与用户的请求无关，其特点是“千人一面”。
  - 动态内容：客户端上显示的内容是在客户请求发出后，根据请求内容和后台数据库中的数据，通过调用执行一个文件（通常为动态脚本）动态产生的，其特点是“一人一面”。

第二章 商务表达层及其技术

- 第一节 商务表达层的功能与实现
- **第二节 静态网页的表达及其技术**
- 第三节 标记语言
- 第四节 动态网页与客户端脚本
- 第五节 服务器脚本

第二节 静态网页的表达及其技术

1、Web概述

- **万维网(World Wide Web)概念：**

欧洲粒子物理研究所的Tim Berners-Lee和Robert Calliau于1990年提出的基于超文本Hypertext、国际互联网Internet和图形用户界面GUI的分布式网络信息系统。

第二节 静态网页的表达及其技术

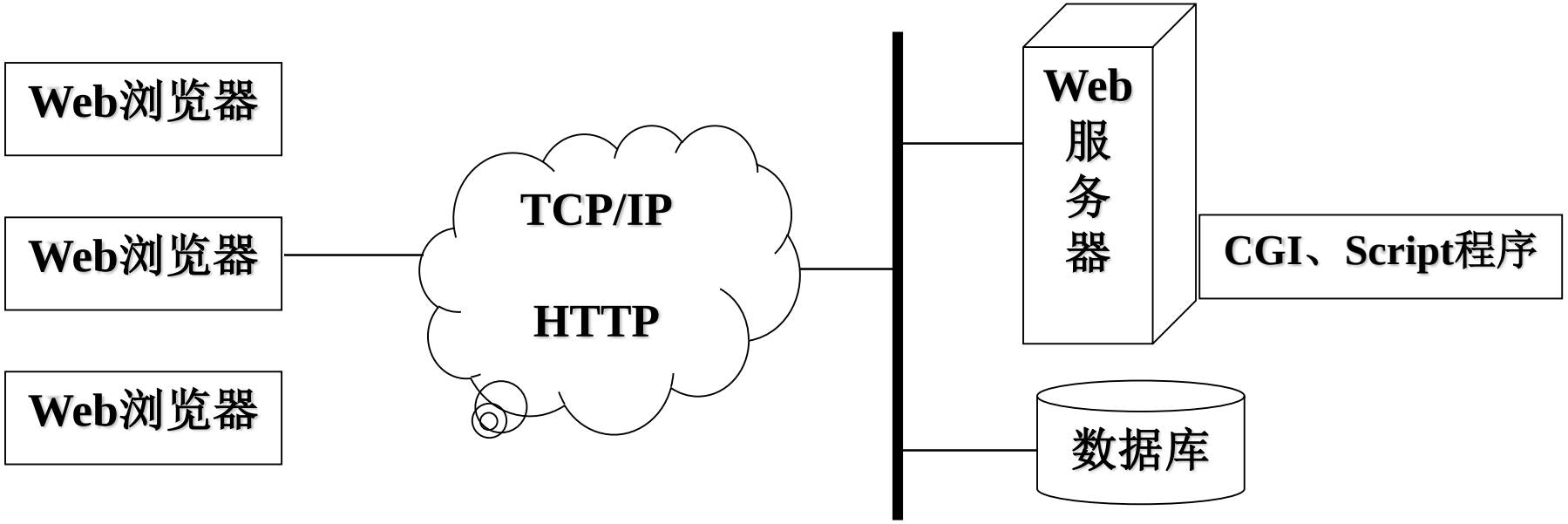
1、Web概述

- 万维网是分布式超媒体(hypermedia)系统，它是超文本(hypertext)系统的扩充。
- 一个超文本由多个信息源链接成，利用一个链接可使用户找到另一个文档。这些文档可以位于世界上任何一个接在因特网上的超文本系统中。超文本是万维网的基础。
- 超媒体与超文本的区别
  - 两者的文档内容不同。超文本文档仅包含文本信息，而超媒体文档还包含其他表示方式的信息，如图形、图像、声音、动画，甚至活动视频图像。

第二节 静态网页的表达及其技术

1、Web概述

- **Web技术原理**：商业数据被组织到一个超文本文件中，借助HTTP协议并通过网络来实现数据的传送和相互通信。



第二节 静态网页的表达及其技术

1、Web概述

- **万维网必须解决的问题**

(1) 怎样标志分布在整个因特网上的万维网文档?

- 使用**统一资源定位符 URL (Uniform Resource Locator)**来标志万维网上的各种文档。
- 使每一个文档在整个因特网的范围内具有唯一的标识符URL。

第二节 静态网页的表达及其技术

1、Web概述

- **万维网必须解决的问题**

(2) 用何协议实现万维网上各种超链的链接?

- 在万维网客户程序与万维网服务器程序之间进行交互所使用的协议，是**超文本传送协议 HTTP (HyperText Transfer Protocol)**。
- HTTP 是一个应用层协议，它使用 TCP 连接进行可靠的传送。

第二节 静态网页的表达及其技术

1、Web概述

- **万维网必须解决的问题**

(3) 怎样使各种万维网文档都能在因特网上的各种计算机上显示出来，同时使用户清楚地知道在什么地方存在着超链？

- **超文本标记语言 HTML (HyperText Markup Language)**
  使得万维网页面的设计者可以很方便地用一个超链从本页面的某处链接到因特网上的任何一个万维网页面，并且能够在自己的计算机屏幕上将这些页面显示出来。

第二节 静态网页的表达及其技术

2、统一资源定位符URL

- **统一资源定位符URL是对可以从因特网上得到的资源的位置和访问方法的一种简洁的表示。**
- URL给资源的位置提供一种抽象的识别方法，并用这种方法给资源定位。
- 只要能够对资源定位，系统就可以对资源进行各种操作，如存取、更新、替换和查找其属性。
- URL相当于一个文件名在网络范围的扩展。因此**URL是与因特网相连的机器上的任何可访问对象的一个指针。**

第二节 静态网页的表达及其技术

2、统一资源定位符URL

问：如何标志一台计算机上的任意一个文件？有大量计算机呢？

- URL也称为网址或链接（包括位置和方法）。
- 资源定位，系统就可以对资源进行各种操作(如存取)。
- 把网络通信看成一个黑箱，对网络资源的操作和对本地的操作没有区别。
- **URL相当于一个文件名在网络范围的扩展（+路径）。**
- 每个在网上可访问的对象都有 URL。
  - 对象：文本文件、图片、音频以及视频文件等。

第二节 静态网页的表达及其技术

2、统一资源定位符URL

- URL的一般形式
  - 由以冒号隔开的两大部分组成，并且在URL中的字符对大写或小写没有要求。
- URL 的一般形式是：

<协议>://<主机>:<端口>/<路径>

ftp —— 文件传送协议 FTP

http —— 超文本传送协议 HTTP

News —— USENET 新闻

第二节 静态网页的表达及其技术

2、统一资源定位符URL

- URL的一般形式
  - 由以冒号隔开的两大部分组成，并且在URL中的字符对大写或小写没有要求。
- URL 的一般形式是：

<协议>://<主机>:<端口>/<路径>

<主机> 是存放资源的主机在因特网中的域名

第二节 静态网页的表达及其技术

2、统一资源定位符URL

- URL的一般形式
  - 由以冒号隔开的两大部分组成，并且在URL中的字符对大写或小写没有要求。
- URL 的一般形式是：

<协议>://<主机>:<端口>/<路径>

有时可省略

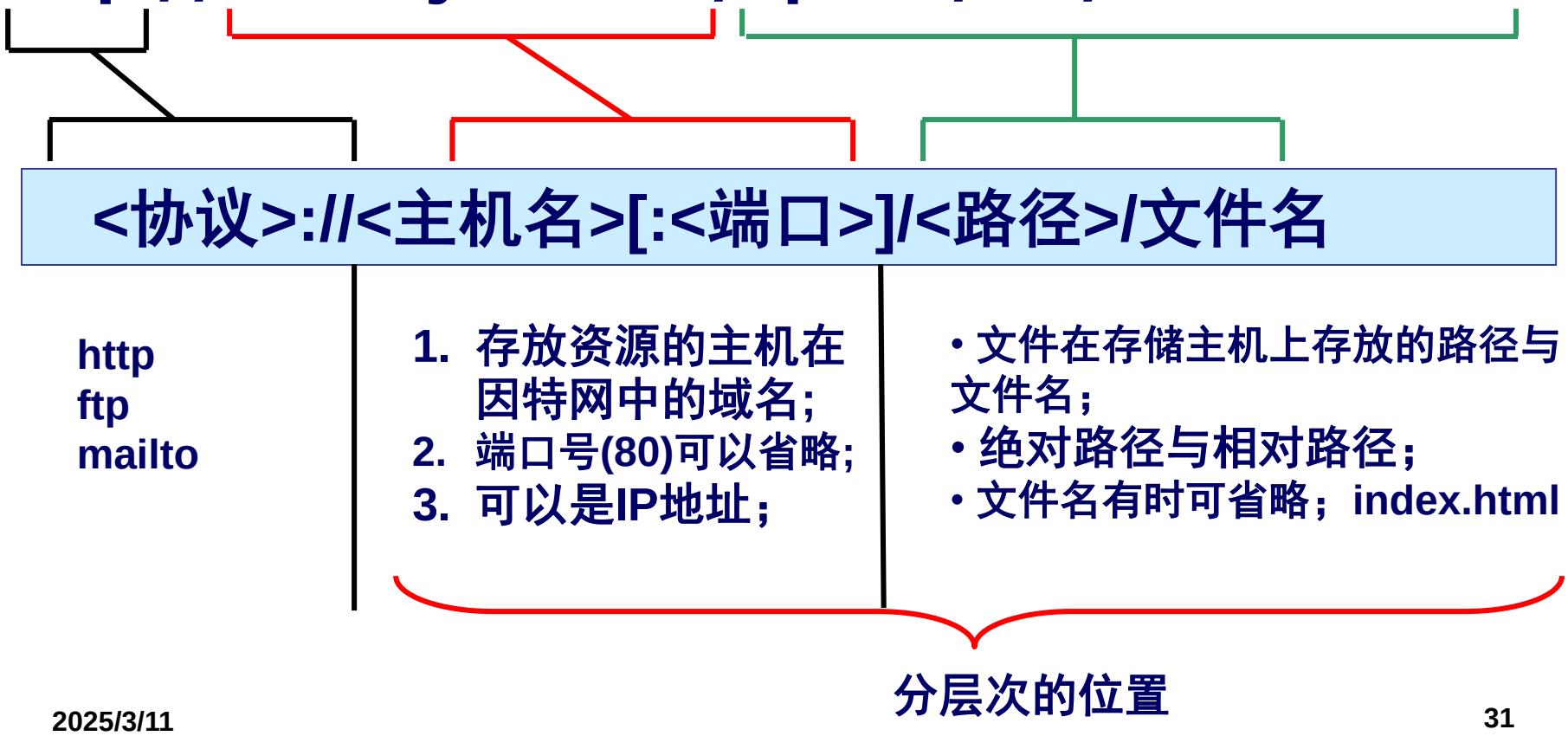
例： http://come.tju.edu.cn/upload/file/12723516.rar

http://come.tju.edu.cn/jxsz/xysz/T/201301/t20130112_169539.htm

第二节 静态网页的表达及其技术

2、统一资源定位符URL

http://come.tju.edu.cn/upload/file/12723516.rar



第二节 静态网页的表达及其技术

2、统一资源定位符URL <协议>://<主机>:<端口>/<路径>

- URL目前支持的主要协议有http、ftp、mailto、telnet、news等。
- 主机全名以双斜杠“//”打头，为资源所在的服务器名或IP地址，如www.microsoft.com或者207.46.19.30。
- 端口号一般设置为默认值（WEB端口号为80、FTP端口号为21、TELNET端口号为23等），它并不是URL的必须项。
- 目录路径的层次以正斜杠“/”分隔。

第二节 静态网页的表达及其技术

2、统一资源定位符URL

- URL最大的缺点：当信息资源的存放地点发生变化时，必须对URL作相应的改变。
- 新的信息资源表示方法:
  - 通用资源标识URI (Universal Resource Identifier)
  - 统一资源名URN（Uniform Resource Name）
  - 统一资源引用符URC（Uniform Resource Citation

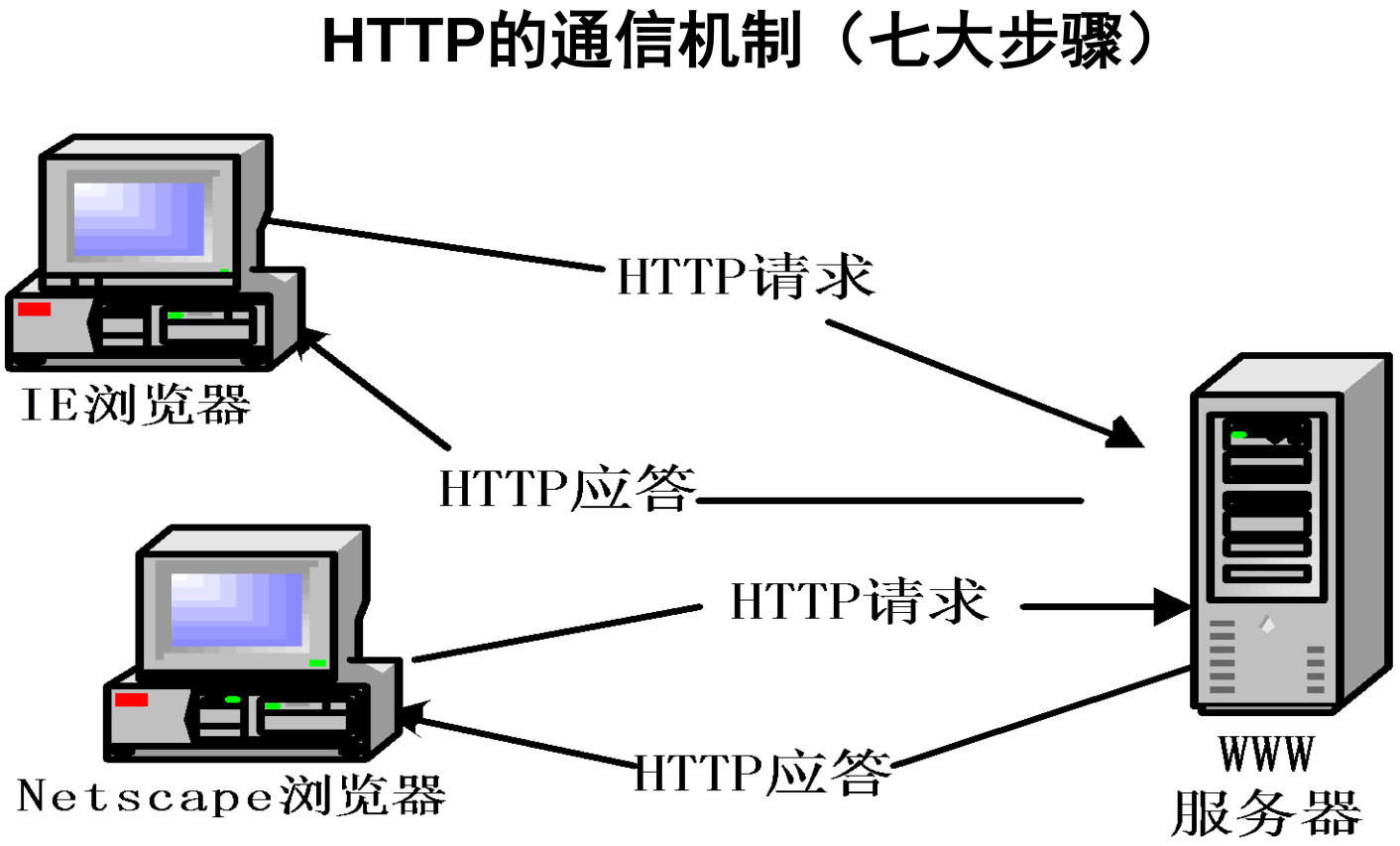
第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **Hyper-Text Transfer Protocol，服务器与浏览器进行沟通的语言，在Internet上用于传输文档。**
- 浏览器与服务器间的通信模式实际上是一种Client/Server模式，因此HTTP协议也是Client程序和Server程序进行通信的一种语言，它建立在TCP/IP协议之上，是一种**请求/应答式协议**。
- HTTP协议是用于从Web服务器传输超文本到本地浏览器的传送协议，**不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分，以及哪部分内容首先显示**（如文本先于图形）等。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP



第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **建立TCP连接**：在HTTP工作开始之前，Web浏览器首先要通过网络与Web服务器建立连接，该连接是通过TCP来完成的，该协议与IP协议共同构建Internet，即著名的TCP/IP协议族，因此Internet又被称作是TCP/IP网络。HTTP是比TCP更高层次的应用层协议，根据规则，只有低层协议建立之后才能，才能进行更高层协议的连接，因此，首先要建立TCP连接，一般TCP连接的端口号是80。
- **浏览器向Web服务器发送请求命令**：一旦建立了TCP连接，Web浏览器就会向Web服务器发送请求命令，例如：GET/sample/hello.jsp HTTP/1.1。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **浏览器发送请求头信息**：浏览器发送其请求命令之后，还要以头信息的形式向Web服务器发送一些别的信息，之后浏览器发送一空白行来通知服务器，它已经结束了该头信息的发送。
- **Web服务器应答**：客户机向服务器发出请求后，服务器会回送应答，HTTP/1.1 200 OK，应答的第一部分是协议的版本号和应答状态码。
- **Web服务器发送应答头信息**：正如客户端会随同请求发送关于自身的信息一样，服务器也会随同应答向用户发送关于它自己的数据及被请求的文档。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **Web服务器向浏览器发送数据**：Web服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束，接着，它就以Content-Type应答头信息所描述的格式发送用户所请求的实际数据。
- **Web服务器关闭TCP连接**：一般而言，一旦Web服务器向浏览器发送了请求数据，它就要关闭TCP连接；然而若浏览器/服务器在其头信息加入了Connection:keep-alive这行代码，TCP连接在发送后将仍然保持打开状态，浏览器可继续通过相同的连接发送请求；保持连接节省了为每个请求建立新连接所需的时间并节约了网络带宽。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- 在通信的时候，服务器端程序预先要绑定好相关的TCP协议和用于标识自已身份的端口号码，并进行TCP连接，使自已处于侦听状态，然后客户端程序才能和它进行正常通信。
- 在HTTP中，规定Web服务器绑定的标准端口号是80，但也可使用其它的端口号。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **HTTP请求**：浏览器向Web服务器发出请求，实际就是向服务器传递一个数据块即请求信息。
- HTTP请求信息由3部分组成：

  请求方法 URL 协议/版本+ 请求头+ 请求正文

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- HTTP请求报文的一些方法

方法（操作）	意义
OPTION	请求一些选项的信息
GET	请求读取由 URL所标志的信息
HEAD	请求读取由 URL所标志的信息的首部
POST	给服务器添加信息（例如，注释）
PUT	在指明的 URL下存储一个文档
DELETE	删除指明的 URL所标志的资源
TRACE	用来进行环回测试的请求报文
CONNECT	用于代理服务器

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- 一个HTTP请求的例子：

```
GET /sample.jsp HTTP/1.1
Accept: image/gif. image/jpeg, */*
Accept-Language: zh-cn
Connection: Keep-Alive
Host: localhost
User-Agent: Mozila/4.0(compatible; MSIE5.01;
Window NT5.0)
Accept-Encoding: gzip, deflate

username=jinqiao&password=1234
```

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- 请求方法 URL 协议/版本：第一行GET /sample.jsp HTTP/1.1中，“GET”表示请求方法，“/sample.jsp”表示URL，“HTTP/1.1”代表协议和协议的版本。
  - 根据HTTP标准，HTTP请求可以使用多种请求方法，例如HTTP1.1支持7种请求方法：GET、POST、HEAD等。在Internet应用中最常用的是GET和POST。
  - URL完整地指定了要访问的网络资源，通常只要给出相对于服务器的根目录的相对目录即可，因此总是以“/”开头。
  - 协议版本声明了通信过程中所使用的HTTP的版本。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- 请求头(Request Header)：包含许多有关的客户端环境和请求正文的有用信息。例如请求头可以声明浏览器所用的语言，请求正文的长度等。
  - Accept: image/gif. image/jpeg.*/*
  - Accept-Language: zh-cn
  - Connection: Keep-Alive
  - Host: localhost
  - User-Agent: Mozila/4.0(compatible; MSIE5.01; Windows NT5.0)
  - Accept-Encoding: gzip, deflate.

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **请求正文**：请求头和请求正文之间是一个空行，这个行非常重要，它表示请求头已经结束，接下来的是请求正文。
  - 请求正文中可以包含客户提交的查询字符串信息：username=jinqiao&password=1234。
  - 在上例的HTTP请求中，请求的正文只有一行内容，但在实际应用中，HTTP请求正文可以包含更多的内容。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **HTTP应答**：HTTP应答与HTTP请求相似，也由3个部分构成：

  协议状态版本代码描述+ 响应头+ 响应正文

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- 一个HTTP应答的例子：

```
HTTP/1.1 200 OK
Server: Apache Tomcat/5.0.12
Date: Mon, 6 Oct 2003 13:23:42 GMT
Content-Length: 112

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

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **协议状态代码描述**：HTTP响应的第一行(HTTP/1.1 200 OK)类似于HTTP请求的第一行，它表示通信所用的协议是HTTP1.1，服务器已经成功的处理了客户端发出的请求（200表示成功）。
  - HTTP应答码：也称为状态码，Web服务器在处理完HTTP命令请求后，在给浏览器发回的HTTP响应中，含有一个应答码，它反映了Web服务器处理HTTP请求的状态，表示响应的结果。
  - HTTP应答码由3位数字构成，其中首位数字定义了应答码的类型。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- **HTTP应答码：**
  - 1XX——信息类（Information）：表示收到Web浏览器请求，正在进一步的处理中（一般不出现，暂时保留不用 ）。
  - 2XX——成功类（Successful）：表示用户请求被正确接收、理解和处理，例如：200 OK是HTTP中标准的成功代码。
  - 3XX——重定向类（Redirection）：表示请求没有成功，客户须采取进一步的动作，例如：若Web服务器决定将HTTP命令请求临时转向到另一个不同的Web服务器，则返回307状态码，表示命令转向。
  - 4XX——客户端错误（Client Error）：表示客户端提交的请求有错误，例如：当服务器解析HTTP的请求命令失败时，它将返回400状态码，404 NOT Found，意味着请求中所引用的文档不存在。
  - 5XX——服务器错误（Server Error）：表示服务器不能完成对请求的处理，例如：若返回501，则表示Web服务器不支持此命令。

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- 响应头（Response Header）：响应头也和请求头一样包含许多有用的信息，例如服务器类型、日期时间、内容类型和长度等。
  - Server: Apache Tomcat/5.0.12
  - Date: Mon, 6 Oct 2003 13:13:33 GMT
  - Content-Type: text/html
  - Last-Moified: Mon, 6 Oct 2003 13:23:42 GMT
  - Content-Length: 112

第二节 静态网页的表达及其技术

3、超文本传送协议 HTTP

- 响应正文：响应正文就是服务器返回的HTML页面，响应头和正文之间也必须用空行分隔。
  - <html>
  - <head>
  - <title>HTTP响应示例</title>
  - </head>
  - <body>
  - Hello HTTP!
  - </body>
  - </html>

第二节 静态网页的表达及其技术

4、静态网页的表达

- **静态网页的定义：**是指网页的显示内容对访问者而言是单向的、固定不变的，即访问者不能通过自己的操作来改变网页的显示内容，若要更新显示内容，必须由网站管理员修改存放在Web服务器上的HTML文件（动态网页的内容对于访问者来说是双向的、动态变化的，访问者可以通过自己的操作获取不同的显示内容，并能有限度的更新Web服务器上的内容）。
- **静态网页的特点：**内容固定不变，制作工艺比较简单。

第二节 静态网页的表达及其技术

4、静态网页的表达

- 静态网页技术是动态网页技术的基础，大多数动态网页是通过在静态网页中插入相关的程序，或动态生成静态网页的方式实现的。
- 静态网页的制作：利用相关的工具（如文档编辑工具、可视化的专用工具Frontpage、集成开发环境中的应用工具Dreamweaver）进行编辑。

第二节 静态网页的表达及其技术

Web服务器的核心就是向请求浏览器传送文件，它并不对文件进行任何处理。这是“静态”名称的核心思想。

4、静态网页的表达

- 静态网页的体系结构
  - 事先编辑好的静态网页的内容（HTML文件、图形图像和声音等多媒体文件）存放在WEB服务器上；
  - 用户要访问网页，通过浏览器向WEB服务器发出HTTP请求；
  - WEB服务器识别请求后将相应内容通过HTTP送回浏览器并由其解释和显示。

第二章 商务表达层及其技术

- 第一节 商务表达层的功能与实现
- 第二节 静态网页的表达及其技术
- **第三节 标记语言**
- 第四节 动态网页与客户端脚本
- 第五节 服务器脚本

第三节 标记语言

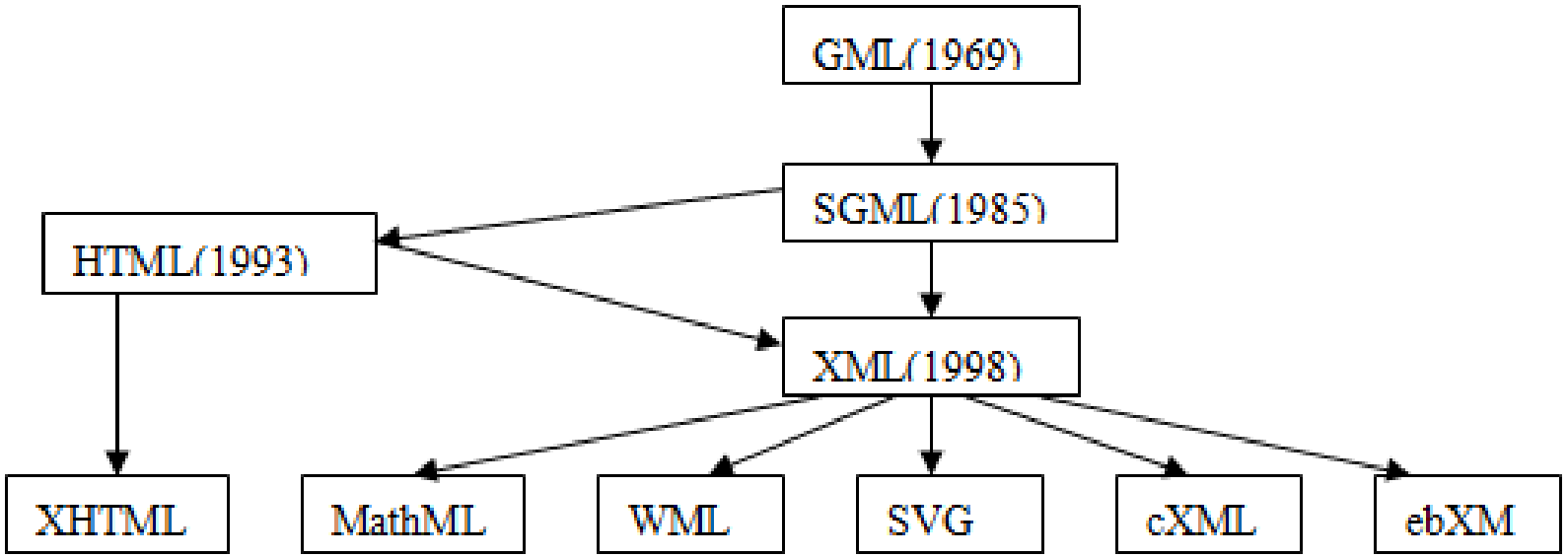
1、通用标记语言(SGML)

- 常用的标记语言
  - SGML (Standard Generalized Markup Language, 标准的通用标记语言)，它是一种元语言(Meta Language)，利用它可以定义其它无数种语言。
  - HTML (Hypertext Markup Language，超文本标记语言)，它是SGML的一个子集，用来制作静态网页。
  - XML (eXtensible Markup Language，可扩展的标记语言)，它也是SGML的一个子集，用来定义数据结构，用于在电子商务系统中传输大规模的数据。

第三节 标记语言

1、通用标记语言(SGML)

- 标记语言的家谱图



几种标记语言

第三节 标记语言

2、 HTML语言

- 与程序设计语言相比，HTML缺少编程语言所需的最基本的变量定义、流程控制等功能，它是通过一系列的标记和属性对超文本的语义进行描述，这些描述经浏览器解释后才成为日常见到的Web页面。
- 与SGML最大的不同的是，SGML没有信息指明数据看起来是什么样子，但通过HTML却可以辨别出文件的架构。

第三节 标记语言

2、 HTML语言

- 特点
  - Html3.2只定义了70种标记。
  - 平台兼容、标记固定、文本文件。
  - 具有超链接。
  - Web服务器不处理标记，由客户端浏览器解释。
  - 丰富的多媒体显示、各种布局处理。

第三节 标记语言

2、 HTML语言

- 缺点
  - HTML更多的关注Web浏览器如何在页面上安排文本、图象和按钮等，过多地考虑外观使其缺乏对结构化数据的表示能力；无法描述数据内容，这正是数据检索、电子商务所必需的；对数据表现的描述能力不够，如不能描述矢量图形、科学符号。
  - HTML还有一些诸如链路丢失后不能自动纠正、下载的内容太多、搜索不方便、时间长等缺点。
  - HTML中有限的标记不能满足众多Web应用的需要，缺乏可扩展性。

第三节 标记语言

2、 HTML语言

- HTML文件及其显示
  - HTML文件实际上是由HTML语言组成的一种纯文本文件，可以通过普通的记事本来进行编辑。
- HTML典型结构
  - <HTML>
  - <HEAD>
  - 此处为头部文件，元素属性及内容
  - </HEAD>
  - <BODY>
  - HTML文件的正文写在这里
  - </BODY>
  - </HTML>

第三节 标记语言

2、 HTML语言

- HTML常用标记(不区分大小写)
  - <html><head><body>
  - <title><style><script>
  - <A HREF = “URL”> HTML超链接
  - <IMG SRC = “URL”> 图像嵌入
  - <FORM> 表单对象（如用户注册页）
  - <INPUT> 表单中的部件（如文本框、按钮）
  - <TABLE> 表格对象（格式控制、表格）
  - <FRAMESET> 框架（HTML页面嵌套、页面格式控制）

第三节 标记语言

2、 HTML语言

- HTML常用标记

标记名	说明
<html>	网页的开始与结束
<head>	网页的头部
<body>	网页的体部，格式为：<body bgcolor=# background="image-URL" bgproperties=FIXED text=# link=# alink=# vlink=# leftmargin=* topmargin=*>
<meta>	<meta http-equiv="Content-Type" content="text/html;charset=#">，用于在HTML中设置MIME字符集等信息 <meta name="Generator" contect="editplus">，用以说明生成工具（如Microsoft FrontPage 4.0） <meta name="KEYWords" contect="cnbruce,cnrose">，向搜索引擎说明网页的关键词 <meta name="Description" contect="cnbruce's blog">，告诉搜索引擎站点的主要内容 <meta name="Author" contect="cnbruce">，告诉搜索引擎站点制作的作者
<title>	网页窗口的标题
<style>	在文档中包含样式
<base>	相对路径转为绝对路径
<script>	包含脚本
<a href="URL">	锚点链接，如： 登庐山 ，用<a name="name">定义一个标签
<hr>	标尺线
<font	字体大小，如今天天气真好！
<b>	字体加粗
<i>	斜字体
<u>	下划线字体
<sup>	上标

第三节 标记语言

2、 HTML语言

HTML例子

- 例1：<A HREF = “http://www.tju.edu.cn”>天津大学</A>
  （见Sample1.html）
- 例2：HTML页面显示（见Sample2.html）

```
<html>
<head>
  <title> 我的HTML页面 </title>
 </head>
 <body>
  <table>
    <tr>
      <td>Hello World</td>
    </tr>
  </table>
 </body>
</html>
```

第三节 标记语言

2、 HTML语言

- HTML的编制方法
  - 记事本/UltraEdit
  - 专门的网页制作工具集成IDE工具
    - Frontpage/Dreamweaver
    - VB.net; JBuilder; Eclipse/MyEclipse

第三节 标记语言

2、 HTML语言

- HTML语言的缺陷
  - 语义性差 - 擅长布局/外观，缺乏信息表达能力
  - 可扩展性差 - 标记数目有限，即使增加标记定义，也无法满足用户的各种显示需求。
  - 内部结构乱，语法上容易出错 – 各种标记混合在一起，不容易读写，且容易出错
  - 交互性差 – 虽然可为用户提供良好的界面，但是与计算机的交互性差，无法从用户处获得有用信息，如搜索引擎的问题。

第三节 标记语言

3、 XML语言

- XML（Extensible Makeup Language，扩展标记语言）是SGML的一个优化子集，是一种通用的数据格式表示语言。它是在SGML的基础上，去除SGML中过于复杂的东西，保留其作为元标记语言的优点，根据实际需要，定义需要的标记。
  - 比如可以按照数学定理来自定义标记，这些自定义的标记放在DTD（Document Type Definition, 文档类型定义）或Scheme（模式）中加以描述。
  - 与SGML一定需要DTD不同，XML中的DTD可有可无，视使用情况而定。

第三节 标记语言

3、 XML语言

- 与HTML一样，XML也是一种标记语言，带标记的元素是XML文档的构造块，这种元素可以有若干属性，并可以包含零到多个元素。
- XML基于文本格式，开发跨平台，采用结构化数据表示形式，数据表示和数据显示相分离，在B2B和B2C型的电子商务系统应用中发挥重要作用。

第三节 标记语言

3、 XML语言

- XML的特点
  - 可扩展性：允许使用者创建和使用自己定义的标记而不是HTML的有限词汇表，其作为电子商务和供应链集成很重要的标记语言，作为数据交换、信息共享的基础；
  - 用户界面和结构化数据分离：用户界面分离于数据，许多先进功能能在XML环境下更容易实现；
  - 自描述性：XML文档通常包括一个文档声明（DTD），因而是自描述的；不仅人可以读懂，计算机也可以处理，XML的数据方式真正做到了独立于应用系统，并且数据可以重用，XML文档被看作是文档的数据库化和数据的文档化；
  - 简明性：与SGML相比，XML规范不到SGML的1/10，简单易懂，有SGML20%的复杂性，80%的功能，且比SGML易学易用易实现。

第三节 标记语言

3、 XML语言

- XML与HTML的主要区别
  - XML的最大特征在于：XML描述文档的结构和含义，不描述页面上元素的格式。XML文档标记仅描述文档内容，不描述文档外观。
  - HTML语言中包含的内容和普通文本是混在一起的，而且标记所表示的内容是预先由标准确定的不能自己定义。
  - XML不是HTML的替代品，两者是不同用途的语言。

第三节 标记语言

3、 XML语言

- XML与HTML的主要区别

XML与HTML综合比较

比较内容	HTML	XML
可扩展性	不具有扩展性	元标记语言，可用于定义新的标记语言
侧重点	如何表现信息	如何结构化描述信息
语法要求	不要求标记的嵌套、配对等，不要求标记之间具有一定顺序	严格要求嵌套、配对，遵循DTD的树形结构
可读性/可维护性	难于阅读、维护	结构清晰，便于阅读和维护
数据与显示的关系	内容与显示方式整合为一体	内容描述与显示方式分离
保值性	不具有保值性	具有保值性
编辑与浏览工具	IE浏览器等	XMLSpy等

第三节 标记语言

3、 XML语言

- XML文档的结构（树形结构）
  - 计算机
    - 类型----个人机
    - 制造商----联想
    - 识别符
      品种----台式机
      型号----同喜500p3
    - 主频--------667
      ----单位-----MHZ
    - 内存--------64
      ---单位-----MB
    - 硬盘--------10
      ---单位-----GB
    - 单价--------7999
      ---单位-----元

第三节 标记语言

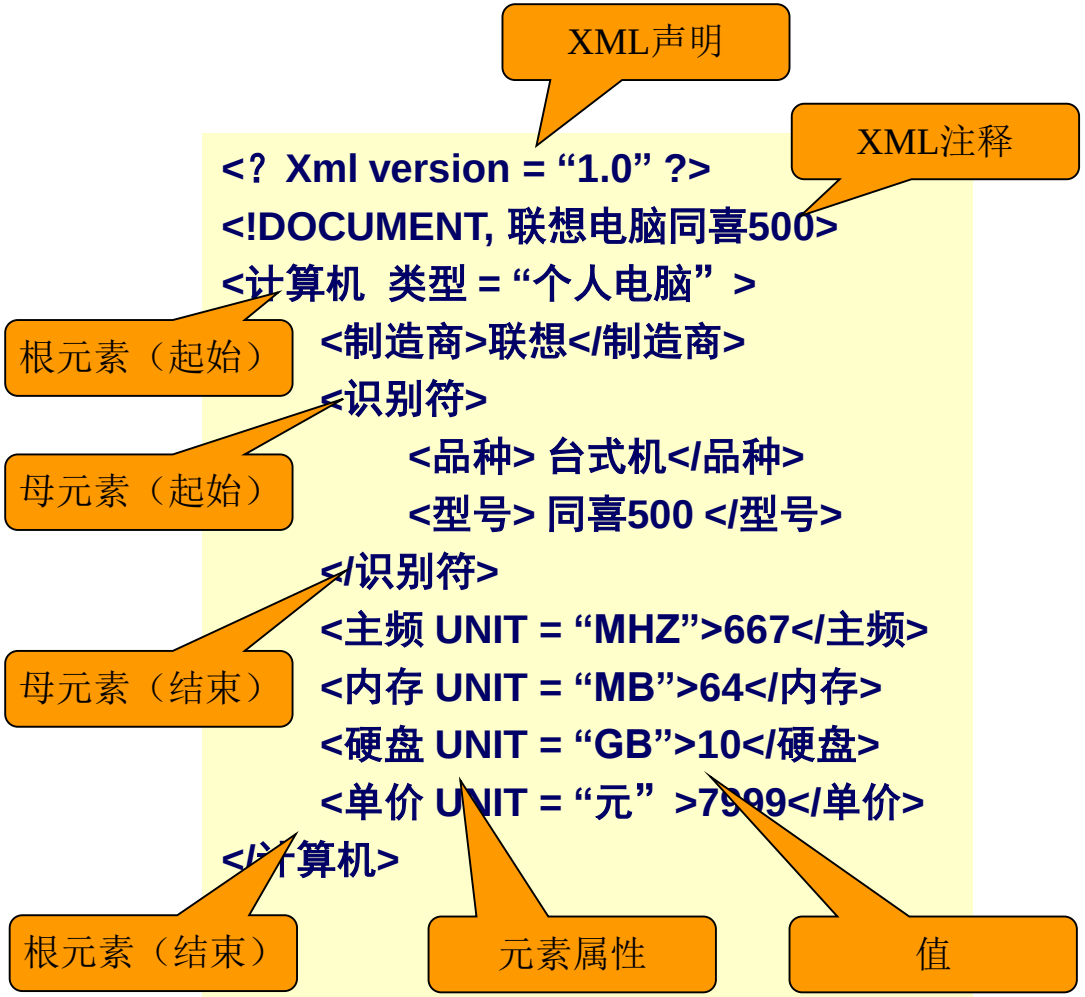
3、 XML语言

- XML和HTML的结构基本相同，分为：元素、属性、值。不同之处在于HTML只允许文档制作者使用固定的标记，而XML允许文档制作者通过创建新标记，更准确的描述数据。
- 下例为两者的文档结构比较。

第三节 标记语言

3、 XML语言

```
<HTML>
<TITLE>联想电脑</TITLE>
<BODY>
<UL>
<LI>联想
<LI>台式机
<LI>同喜500
<LI>667MHZ
<LI>64MB
<LI>10GB
<LI>7999
</UL>
</BODY>
</HTML>
```



第三节 标记语言

3、 XML语言

XML文档的组成

- 在基本文档结构上主要由3部分组成：
  - XML声明
    - <? xml version = “1.0” encoding = “GB2312” >
  - XML中的标记(TAGS)
    - 元素+属性两部分构成
  - XML中的注释和处理命令
- XML良构：一个XML文档如果具有正确配对的首尾标记、正确的元素嵌套、并且没有错位或遗漏的字符等，那么它就是一个良构的XML文档。

第三节 标记语言

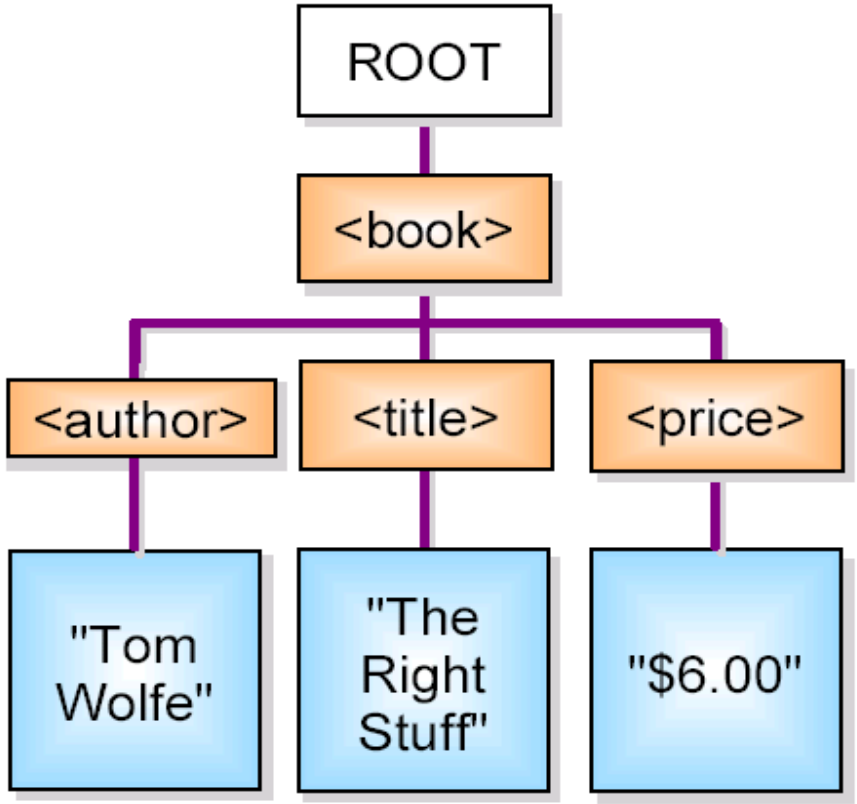
3、 XML语言

XML文档的组成

```
<book>
  <author>
    Tom Wolfe
  </author>
  <title>
    The Right Stuff
  </title>
  <price>
    $6.00
  </price>
</book>
```

=

XML文档主体结构



第三节 标记语言

3、 XML语言

XML文档的组成

- XML声明

<?xml version="1.0" encoding="gb2312"?>

  - 必须放在文件的开始
  - 一般包含版本号（1.0）、文字编码等。

第三节 标记语言

3、 XML语言

XML文档的组成

- 注释

<!-- 表示注释的开始 --> 表示注释的结束

  - XML解析器将忽略其中的所有数据
  - 可以放在XML声明之后的任何位置
  - 可以写成一行或多行
  - 不能放在某个标记之间
  - 作用
    - 对某代码行的功能等内容进行注释
    - 保留临时或没有完成的代码行

第三节 标记语言

3、 XML语言

XML文档的组成

- 处理指令

以“<?”开头、“?>”以结尾

- 允许XML文件中包含由应用（css、xs、Javascript等）来处理的指令。例如：
- <?xml-stylesheet type="text/css" href="style.css"?>
- <?xml-stylesheet type=“text/xsl” href=“style.xsl”?>

第三节 标记语言

3、 XML语言

XML标记

- XML标记的界定：

  1对尖括号 —— < 与 >，以“<”开始，以“>”结束

- XML标记的分类：
  - 开始标记（首标记）　如：<message>
  - 结束标记（尾标记）　如：</message>
- XML标记的基本结构：标记名、属性与值
  - 如：<note date= " 08/08/2008 " >
  - 标记名：具有一定含义，能够描述数据的内容、区分大小写。
  - 属性与值：属性是对元素信息的补充，值是属性的量化。

第三节 标记语言

```
<book category="CHILDREN">
<title>Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
```

3、 XML语言

XML元素

- XML元素是XML文档中的数据单元，用来表示、存放和组织数据。
- XML元素的组成：
  - 开始标记
  - 文本数据内容（元素的值）
  - 结束标记
- 元素的基本结构：

```
<元素名  属性名="属性值">
    文本数据内容（元素的值）
</元素名>
```

第三节 标记语言

```
<book category="CHILDREN">
<title>Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
```

3、 XML语言

XML元素

- 元素名：对应于数据的字段名
- 元素的值：对应于数据的记录
- 元素可包含其他元素、文本或者两者的混合。
- 元素也可以拥有属性。

第三节 标记语言

3、 XML语言

XML元素

- 元素嵌套
- XML的标记呈层次结构，所以XML中元素嵌套是完全包含关系，不能重叠或交叉。嵌套关系有：
  - 父子关系：一个元素中嵌套了一个或多个元素，这个元素即称为母元素，而包含的元素称为子元素。
  - 兄弟关系：具有相同母元素的子元素之间的关系。

第三节 标记语言

3、 XML语言

XML元素

- XML元素的分类（以元素嵌套关系进行分类）：
  - 根元素
  - 母元素
  - 子元素
- 最高层的母元素称为根（root）元素。
- 每个XML文档有且只有一个根元素，其它元素都包含在这个根元素中。

第三节 标记语言

3、 XML语言

XML元素

- 元素名的命名规则：
  - 元素名：不能为空，可以含字母、数字以及其他的字符；但不能不能包含空格。
  - 首字符：可以以字母或下划线（-）或冒号（:）开头；但不能以字符 “xml”（或者 XML、Xml）开始。
  - 不能以数字或者标点符号开始。
  - 其它字符：多个字母、数字、破折号、下划线、句号。
  - 区分大小写。

第三节 标记语言

3、 XML语言

XML属性

- 属性提供有关元素的额外信息
- XML 属性值必须加引号
  - 属性值必须被引号包围，单引号和双引号均可使用。
  - <person sex="female">
  - <person sex='female'>
- 一个元素可以有多个属性，它的基本格式为：
  - <元素名 属性名1="属性值1" 属性名2="属性值2">
- 特定的属性名称在同一个元素标记中只能出现一次。

第三节 标记语言

3、 XML语言

XML属性

- XML 元素 vs. 属性

```
<person sex="female">
<firstname>Anna</firstname>
<lastname>Smith</lastname>
</person>
```

```
<person>
<sex>female</sex>
<firstname>Anna</firstname>
<lastname>Smith</lastname>
</person>
```

- 子元素及属性均可提供相同的信息。

第三节 标记语言

3、 XML语言

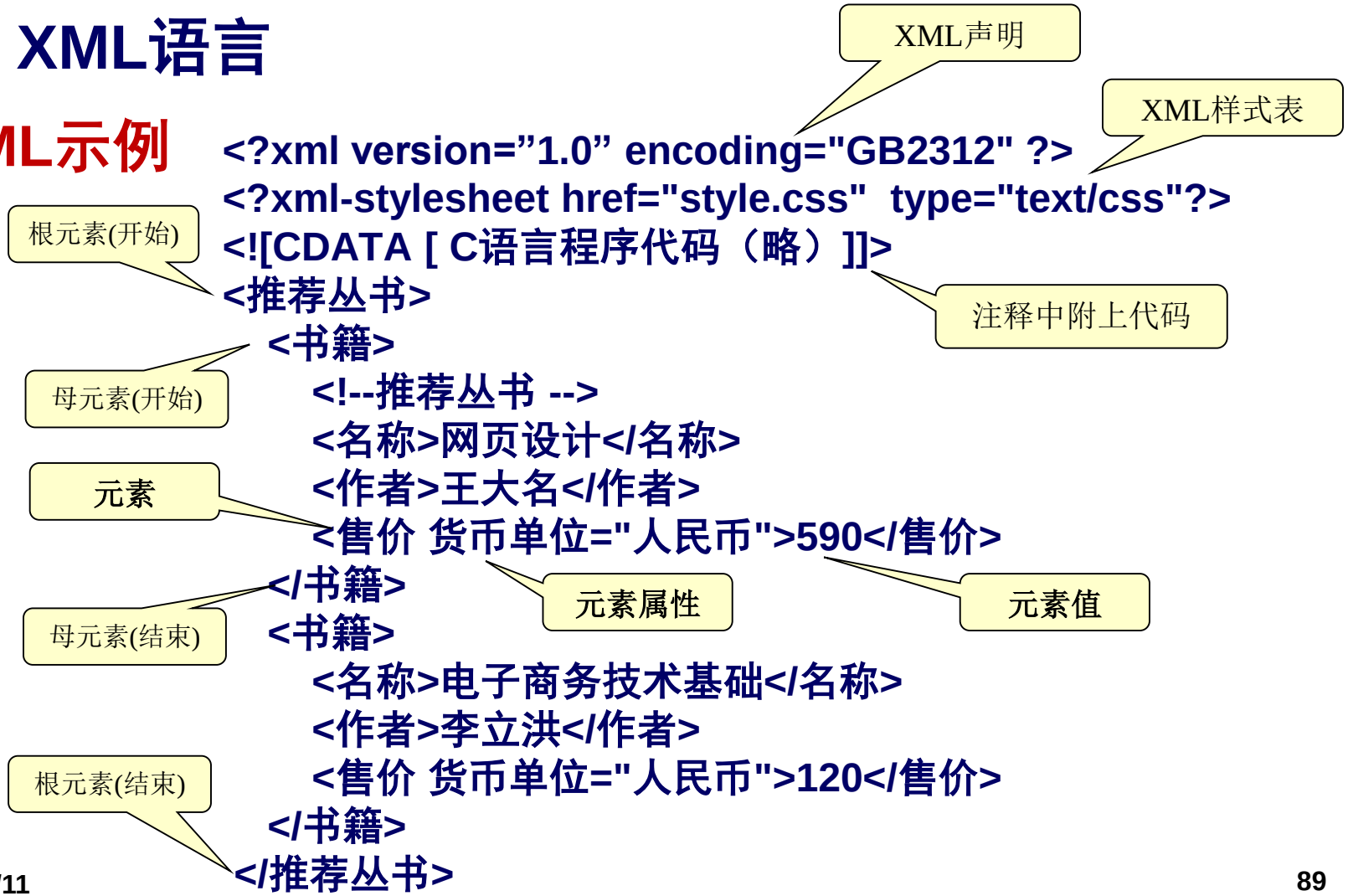
XML属性

- 因使用属性而引起的一些问题：
  - 属性无法包含多个值（子元素可以）；
  - 属性无法描述树结构（子元素可以）；
  - 属性不易扩展（为未来的变化）；
  - 属性难以阅读和维护。
- 因此属性通常提供不属于数据组成部分的信息。例如：
  - <file type="gif">computer.gif</file>
- 文件类型与数据无关，但对需要处理这个元素的软件来说却很重要。

第三节 标记语言

3、 XML语言

XML示例




```
<?xml version=”1.0” encoding="GB2312" ?>
<?xml-stylesheet href="style.css"  type="text/css"?>
<![CDATA [ C语言程序代码（略）]]>
<推荐丛书>
   <书籍>
      <!--推荐丛书 -->
      <名称>网页设计</名称>
      <作者>王大名</作者>
      <售价 货币单位="人民币">590</售价>
   </书籍>
   <书籍>
      <名称>电子商务技术基础</名称>
      <作者>李立洪</作者>
      <售价 货币单位="人民币">120</售价>
   </书籍>
</推荐丛书>
```

第三节 标记语言

3、 XML语言

<书籍>
<名称>电子商务技术基础
<作者>
</名称> 无名氏
</作者>

编写XML要求

- 文档必须以XML声明开始。
  - 如：<?xml version="1.0" encoding="GB2312"?>
- 在XML文件中有且只能有一个根元素。
- 所有的元素都要配对：有起始和结束标记。
  -
<hr>无对应结束标记，则写成
<hr/>。
- 标记之间不得交叉，嵌套必须正确。
  - XML中规定，所有的元素排列必须是严谨的树状结构。
- 所有属性值都得含有引号。
- 大小写有区别。

第三节 标记语言

3、 XML语言

XML编写原则

- 所有的元素都要配对；
- 标记之间不应交叉；
- 所有属性值都得用引号；
- 区别大小写；
- 空白字符不相同。

第三节 标记语言

3、 XML语言

XML中的命名空间（namespace）

- 为什么用？
  - 有着相同标记定义的 XML 文件合并或处理时会发生冲突
  - 避免名称相同的元素（来自不同的文件、含义不同）造成理解和处理上的混淆
  - 命名空间的目的是去除元素标记名字的模糊性
- 作用：通过给元素/属性加上命名空间，可以唯一标识一个元素和属性，从而避免因名称上的冲突带来的问题。
- 声明的一般形式
  - xmlns: name = “ URI ”
  - 例：xmlns: river = “http://www.myserver.com”
- 作用范围：只能作用在自己所在的目录树。

命名空间前缀

命名空间标识

声明关键字

第三节 标记语言

3、 XML语言

XML中的命名空间（namespace）

XML 文档1携带着某个表格中的信息：

```
<table>
  <tr>
  <td>Apples</td>
  <td>Bananas</td>
  </tr>
</table>
```

XML 文档2携带有关桌子的信息（一件家具）：

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

假如这两个 XML 文档被一起使用，由于两个文档都包含带有不同内容和定义的 <table> 元素，就会发生命名冲突。

XML 解析器无法确定如何处理这类冲突。

第三节 标记语言

3、 XML语言

XML中的命名空间（namespace）

XML文档1使用命名空间

- XML文档1携带着某个表格中的信息：
- **<table xmlns="http://www.w3.org/TR/html4">**
- **<tr>**
- **<td>Apples</td>**
- **<td>Bananas</td>**
- **</tr>**
- **</table>**

第三节 标记语言

3、 XML语言

XML中的命名空间（namespace）

XML文档2使用命名空间

- XML文档2携带着有关一件家具的信息：
- **<table xmlns="http://www.w3school.com.cn/furniture">**
- **<name>African Coffee Table</name>**
- **<width>80</width>**
- **<length>120</length>**
- **</table>**

第三节 标记语言

3、 XML语言

XML数据类型的定义（DTD和XML Schema）

- XML文档是一种结构化的标记文档，创建XML文档之前，首先要确定其元素和结构，然后根据结构的定义，填入实际的内容，形成一个XML文档。
- XML这样的结构文件有两种定义方式（均由W3C开发）
  - 文档类型定义（DTD, Document Type Definition）
  - 模式定义（Schema）
- XML必须与DTD或Schema相比较，如果XML文档中有任何信息不符合它们的规定，则不会通过有效性检查。如果一个XML文档在结构和内容上符合某个DTD或Schema的规定，那么该XML文档对于这个DTD或Schema来说就是有效的（Valid）。

第三节 标记语言

3、 XML语言

XML数据类型的定义（DTD和XML Schema）

- DTD定义：是一套关于标记符的语法规则，它定义了可用在文档中的元素、属性和实体，以及这些内容之间的相互关系。

XML文档

```
<?xml version="1.0" encoding="GB2312" ?>
<?xml-stylesheet href="style.css"  type="text/css"?>
<推荐丛书>
    <书籍>
        <!--  推荐丛书 -->
        <名称>网页设计</名称>
        <作者>张凡</作者>
        <售价 货币单位="人民币">590</售价>
    </书籍>
    <书籍>
        <名称>电子商务技术基础</名称>
        <作者>李立洪</作者>
        <售价 货币单位="人民币">120</售价>
    </书籍>
</推荐丛书>
```

DTD文档

```
<?xml version="1.0" encoding="GB2312" ?>
<!ELEMENT 推荐丛书 (书籍*)>
<!ELEMENT 书籍 (名称,作者+,售价+)>
<!ELEMENT 名称(#PCDATA)>
<!ELEMENT 作者(#PCDATA)>
<!ELEMENT 售价(#PCDATA)>
<!ATTLIST 售价 货币单位(人民币,新台币,港币,美元)人民币>
```

第三节 标记语言

3、 XML语言

XML数据类型的定义（DTD和XML Schema）

- DTD
  - “()”：表示把括号内的元素或数据类型合并为一个单位
  - “,”：表示把元素按次序排列，用逗号排列的元素在XML文档中出现的顺序要求严格按照顺序排列
    - <!ELEMENT 联系人(姓名,电话)>
      说明联系人中含两个子元素，在XML文档中须将姓名在前，电话在后
    - <!ELEMENT 联系人(姓名 电话)>
      无逗号则编写次序无要求
  - “+”：表示该元素出现一次或多次，但不能不出现
    <!ELEMENT 联系人(姓名,电话+)>
  - “*”：表示该元素出现零次或多次。
    <!ELEMENT 联系人(姓名,电话*)>
  - “|”：表示在符号作用范围内的元素只能出现一个，而且只能出现一次。
    <!ELEMENT 联系人(姓名,(电话|EMALL))>
  - “?”：表示元素可选，即不出现或出现一次。
    <!ELEMENT 联系人(姓名,(电话|EMALL),地址？)>

第三节 标记语言

3、 XML语言

XML数据类型的定义（DTD和XML Schema）

- DTD的作用：XML解析器可以根据DTD及时确认收到的资料格式是否正确无误。
- 关联XML与DTD的方法：

外接book.dtd

内嵌写入

外接方式	内嵌方式
<?xml version="1.0" encoding="GB2312" ?> <?xml-stylesheet href="style.css" type="text/css"?> <!DOCTYPE 推荐丛书 SYSTEM "book.dtd" > <推荐丛书> <书籍> …… </书籍> </推荐丛书>	<?xml version="1.0" encoding="GB2312" ?> <?xml-stylesheet href="style.css" type="text/css"?> <!DOCTYPE 推荐丛书 [<!ELEMENT 推荐丛书 (书籍*)> <!ELEMENT 书籍 (名称,作者+,售价+)> ……]> <推荐丛书> <书籍> …… </书籍> </推荐丛书>

第三节 标记语言

3、 XML语言

XML数据类型的定义（DTD和XML Schema）

- DTD的作用：XML解析器可以根据DTD及时确认收到的资料格式是否正确无误。
- 关联XML与DTD的方法：
  - 内嵌方式
    - DTD包含在XML源文件里面，预先包装在含有以下句法构造的DOCTYPE定义当中：
    - <!DOCTYPE root-element [element-declarations]>

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to      (#PCDATA)>
<!ELEMENT from    (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body    (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

定义了类型注释的文件

第三节 标记语言

3、 XML语言

XML数据类型的定义（DTD和XML Schema）

- DTD的作用：XML解析器可以根据DTD及时确认收到的资料格式是否正确无误。
- 关联XML与DTD的方法：
  - 外接方式
    - DTD不在XML文件内部，在含有以下句法构造的DOCTYPE声明中预先包装进去：
    - <!DOCTYPE root-element SYSTEM "filename">

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

对应的包含了DTD的“note.dtd”文件为：

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

第三节 标记语言

3、 XML语言

XML数据类型的定义（DTD和XML Schema）

- Schema: 作用与DTD相同(.xsd扩展名)，可由XMLSpy等工具直接创建。
- Schema与DTD比较其优点
  - DTD与XML是不同的语法编码方式，XML Schema使用的是一种XML语言。
  - DTD中都是全局声明，每个元素唯一名字只能声明一次，而XML Schema 既有全局声明也有局部声明。
  - DTD不能对给定的元素属性的数据类型进行定义，而XML Schema允许详细定义。

第三节 标记语言

3、 XML语言

XML数据的表现与样式链接

- 利用XML语言和DTD文档，可以将数据、文件资料准确完整的表示出来，但由于XML文档具有“用户界面和结构化数据分离”的特点，XML文档中缺少显示格式信息，XML文档还不能像用户所希望的那样在IE中表现出来。
- 因此，XML不涉及显示效果，所以要表现出效果，还需要有相关的文件提供有关显示和处理XML文档的信息，这种文件就是样式文件（*.CSS）或样式转换文件（*.XSL）。
- 使用方法：在XML文件中放入一个PI指令语句
  - <?xml-stylesheet href=“style.css” type = “text/css” ?>
  - <?xml-stylesheet href=“style.xsl” type = “text/xsl” ?>

第三节 标记语言

CSS的语法由三部分组成：
selector {property: value}（选择器{属性:值}）
例：h1,h2,h3,h4,h5,h6 {color: green}
演示了将所有的标题元素组合起来，它们的颜色都变为绿色。

3、 XML语言

XML数据的表现与样式链接

- 层叠样式表（Cascading Style Sheet，CSS）从根元素开始到该树的最后一个元素遍历XML文档树而显示整个XML文档，它按元素在树中出现的顺序遍历该树；它提供给每个元素添加一种样式的手段。
- 样式：定义了元素怎样去显示，它一般存储在样式表中，利用外部样式表（所有主流浏览器都支持样式表）可以提高工作效率，而外部样式表就存储在CSS文件中。
- 样式表的工作原理：样式表将元素的显示属性保存在一个外部的.css文件中，外部样式表能够实现仅仅通过编辑一个单独的CSS文档就改变所有出现在WEB中的外观和布局，其原理就是一动多变。

第三节 标记语言

3、 XML语言

XML数据的表现与样式链接

- XSL(eXtensible Style Language)和XSLT(eXtensible Style Language Transformation)是用于XML文本的转换和格式化的标准语言。
- 使用XSL，用一个样式单文件XSLT，XML文本可转换成多种格式的输出，如HTML, WML, PDF, Postscript, PlainText等。
- 在服务器端/客户端可以应用XSL解析器解析XML文档为HTML文件形式。

第三节 标记语言

3、 XML语言

XML的用途

- 数据存储
  - 如学生信息数据的存储，数据库表 => XML文档格式存放
- 数据转换
  - 如在Oracle与SQL Server之间，XML与EXCEL之间不兼容系统之间交换数据（或进行表格型数据的复制）
- 数据共享
  - 提供了一种与软硬件无关的数据共享方法，是一种能够被不同的应用程序读取的数据文件。不同的应用程序能够像对待数据库一样，将XML文档作为数据源来处理。
  - 数据共享可以通过XML编程接口的DOM（文档对象模型）或SAX（Simple API for XML）实现，可以通过DOM或SAX对XML进行检查、修改、删除和重组。

第二章 商务表达层及其技术

- 第一节 商务表达层的功能与实现
- 第二节 静态网页的表达及其技术
- 第三节 标记语言
- **第四节 动态网页与客户端脚本**
- 第五节 服务器脚本

第四节 动态网页与客户端脚本

1、 动态网页

- 动态页面的核心技术在于提供更加丰富的用户交互能力，特别是将业务数据处理、存储与Web页面进行集成，能够将来自用户的数据用于业务处理，并存储到企业数据库中，也能够将处理结构和企业数据库显示在用户的网页上。
- “动态”的重点不是如何获得数据（数据层技术），也不是如何构建复杂的业务逻辑（逻辑层技术），而是实现用于与应用程序的通信，以及如何将所获得数据显示在网页中，即目标是通过网页编程，对信息的表达上。
- 上述技术的区别，反应了电子商务网站和电子商务系统设计思路与技术基础的不同。

第四节 动态网页与客户端脚本

1、 动态网页

- 动态网页与静态网页的区别
  - 静态网页主要通过标记语言完成信息的组织和显示，无商务逻辑处理及与用户进行交互的能力；动态网页首先也是用来表现数据的，但由于在网页中加入了一些脚本程序，使得动态网页具备了商务逻辑处理的功能，并能和用户进行交互。
  - 动态网页交互可在客户端实现，也可在服务器端实现：在客户端实现时需要使用客户端脚本，如JavaScript，Applet等；而在服务器端实现时则需要使用服务器端脚本，如JSP、ASP、PHP等。

第四节 动态网页与客户端脚本

1、 动态网页

- 动态网页实际上不是独立存在于服务器上的网页文件，只有当用户请求时服务器才返回一个完整的网页。
- 动态网页内容的生成可在客户端完成（通过客户端脚本），也可在服务器端完成（通过服务器端脚本）。

第四节 动态网页与客户端脚本

1、 动态网页

- 动态网页的特征
  - 以数据库技术为基础，内容随客户访问要求的改变而改变；
  - 能与用户进行交互，根据不同客户请求来生成不同的页面内容；
  - 可以实现如用户注册、用户登录、在线调查、用户管理、订单管理等功能，是连接商务表达层和商务逻辑层的桥梁；
  - 可含有动画的内容；
  - 网页URL的后缀不是.htm、.html、.shtml、.xml等静态网页的常见形式，而是以.asp、.jsp、.php、.perl、.cgi等形式为后缀，且在动态网页网址中通常有一个标志性的符号“？”。

第四节 动态网页与客户端脚本

1、 动态网页

- 两大类脚本程序
  - 客户端脚本- VBScript, JavaScript, Applet等
  - 服务器端脚本 - JSP, PHP, ASP/ASP.NET等
- 脚本语言
  - 完成特殊功能的小“程序段”，不像将一段的程序（c/c++/java）被编译，而是逐行解释执行的。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

- 优势
  - 用户界面与业务逻辑的分离
  - 和用户交互时与服务器通信很少
  - Web页面可以离线浏览
  - 不需要很多服务器资源
- 支持这种结构的技术
  - Java Applet/可下载的Java程序
  - JavaScript
  - VBScript （仅IE支持）
  - ActiveX控件（仅IE支持）

第四节 动态网页与客户端脚本

2、 客户端脚本技术

VBScript（Microsoft VB Scripting Edition）

- 是基于Visual Basic 语言的脚本，主要由PWS/IIS Web服务器所支持。
- 程序结构

```
Dim 参数列表;
Sub
   程序段;
End Sub
```

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- 是一种直接嵌入在HTML文件中，基于对象和事件并具有安全性能的解释性脚本语言。它是一种客户端脚本语言，最早出现在Netscape 2.0上，主要由Java的语法派生而来。
- JavaScript的特点
  - 简单——小程序，由浏览器编译执行
  - 动态——直接对用户输入做出响应
  - 跨平台——只依赖于浏览器
  - 节省交互时间

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- JavaScript与Java区别
  - 概念和结构完全不同：Java是一种程序语言，而javascript是一种描述语言，它们使用的技术相差甚大。
  - 针对不同的目的：Java（由Sun Microsystems开发）是与C和C++同类的语言，它的功能更强大，结构也更复杂，可用来开发软件以及手机、游戏等项目；而JavaScript则多用于网页或类似于网页的编程，可用于服务器端也可用于客户端。
  - 基于对象（ObjectBased）和面向对象的区别：Java是一种真正的面向对象的语言，即使是开发简单的程序也必须设计对象；JavaScript是一种脚本语言，它可以用来制作与网络无关的、与用户交互作用的复杂软件，它是基于对象和事件驱动的编程语言，本身提供了非常丰富的内部对象供设计人员使用。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- JavaScript与Java区别(续)
  - 解释和编译：Java的源代码在传递到客户端执行之前，必须经过编译，因而客户端上必须具有相应平台上的仿真器或解释器，它可以通过编译器或解释器实现独立于某个特定的平台编译代码的束缚；JavaScript是一种解释性编程语言，其源代码在发往客户端执行之前不需经过编译，而是将文本格式的字符代码发送给客户端由浏览器解释执行。
  - 强变量和弱变量：Java采用强类型变量检查，即所有变量在编译之前必须作声明；JavaScript中的变量声明采用弱类型，即变量在使用前不需作声明，而是解释器在运行时检查其数据类型。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- JavaScript与Java区别(续)
  - 代码格式不一样：Java是一种与HTML无关的格式，必须通过像HTML中引用外媒体那样进行装载，其代码以字节代码的形式保存在独立的文档中；JavaScript的代码是一种文本字符格式，可以直接嵌入HTML文档中，并且可动态装载。
  - 嵌入方式不一样：在HTML文档中，JavaScript使用<script>...</script>来标识，而Java使用<applet>...</applet>来标识。
  - 绑定方式不一样：Java采用静态绑定，而JavaScript采用动态绑定。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- **JavaScript的用途**
  - JavaScript为HTML使用者提供了一种程序工具，可以在HTML页面放入一小段JavaScript代码；
  - JavaScript可以为HTML页面添加动态内容；
  - JavaScript可以被事件触发，即JavaScript可被设置为当指定事件触发时执行，例如当一个页面加载完成时或是浏览者点击了某个HTML元素；
  - JavaScript可以获取或写入HTML元素；
  - JavaScript可以校验数据，即在表单提交前对表单数据进行校验，可以减轻服务器的负担；
  - JavaScript可以检测浏览者的浏览器类型并为其加载相应的页面；
  - JavaScript可以创建cookies，它可以在浏览者的计算机上存储和处理信息。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- **JavaScript中的数据类型与变量**
  - **JavaScript有六种数据类型**：Number（数值数据）、String（字符串）、Object（对象） 以及 Boolean（布尔值）、Null（空） 和Undefined（未定义的数据类型）。
  - **Number**：包括整数和浮点数。整数可以为正数、0或者负数；浮点数可以包含小数点, 也可以使用科学记数法。
  - **String**：字符串，用单引号或双引号来说明。
  - **Object**： JavaScript 中的重要组成部分，含有方法和属性。在JavaScript中，主要存在三种类型的对象：一是由浏览器根据页面的内容自动生成的对象，如表单(Form)对象、框架(Frame)对象、链接对象等；二是内置对象，如Date、Math、window、document等；三是用户自定义的对象。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- JavaScript中的数据类型与变量(续)
  - Boolean：代表某种可能，其值为true或false。这是两个特殊的值，不能用1 和0来表示。
  - Null：表示没有任何值。
  - Undefined：表示一个变量被创建后，在未赋值以前所具有的值 。例如: var x; var y = true; 则y变量就成为一个Boolean 类型的变量，而x变量在赋值之前的类型暂时为Undefined。
  - 数据类型转换示例（与java有所不同）：
    - var x=”123.0578”; //x为字符串
    - x=parseFloat(x).toFixed(3); //小数点保留3位
    - x=Math.round(parseFloat(x)*100); //四舍五入取整
    - x=Number(“123”); //x的值为整数
    - JavaScript的大部分语句/语法与Java类似。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- JavaScript中对象的定义和使用方式

用户自定义Dog类

```
<script language="javascript">
function Dog(name,breed,color,sex) {
    this.name=name;
    this.breed=breed;
    this.color=color;
    this.sex=sex;
    this.eat=eatfun;   //eat方法等同于eatfun方法
}
function eatfun(x) {document.write(x);}
theDog = new Dog("Gabby","Lab","chocolate","girl");
theDog.eat("has ate!");
</script>
```

用户自定义函数

创建用户自定义对象theDog

为自定义对象theDog的eat方法赋值

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- HTML中嵌入JavaScript的三种主要方式
  - 放置在标记对<script></script>之间
  - 放置在<script>标记的src属性指定外部文件中
  - 放置在事件处理程序中，该事件处理程序由onclick或onmouseover这样的HTML属性值指定
- JavaScript可放置在HTML中的位置
  - <head></head>头文件中
  - <body></body>体文件中
  - 由src属性所指定的外部URL文件中

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- 利用JavaScript制作动态网页
- 例1：
  - <html>
  - <body>
  - <script type="text/javascript">
  - document.write("Hello World!")
  - </script>
  - </body>
  - </html>

上述代码将在HTML页面上输出文字：Hello World!

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- 利用JavaScript制作动态网页
  - 通过 <script> 标签在HMTL页面中插入JavaScript（使用type 属性定义所要插入的脚本语言）。
  - <script type="text/javascript"> 和 </script> 分别标记了JavaScript代码的开始和结束。
  - document.write 是用于页面输出的标准JavaScript命令，将document.write语句写入到<script type= "text/javascript">和</script>标签之间，浏览器就可以将它识别为JavaScript命令并执行。
  - 注意：如果没有使用<script>标签，浏览器将把document.write("Hello Word!")语句识别为普通文本并将其全部输出。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

JavaScript语言

- 利用JavaScript制作动态网页
- 例2：

```
<script type="text/javascript">
//You will receive a different greeting based on what day it is. Note that
Sunday=0,Monday=1, Tuesday=2, etc.
var d=new Date()
theDay=d.getDay()
switch (theDay)
{case 5: document.write("Finally Friday")
  break
case 6: document.write("Super Saturday")
  break
case 0: document.write("Sleepy Sunday")
  break
default: document.write("I'm looking forward to this weekend!")
}
</script>
```

第四节 动态网页与客户端脚本

2、 客户端脚本技术

Java Applet

- Java Applet是Java语言中编写包含在网页中的“小应用程序”，它不能独立运行，必须嵌入在一个HTML文件中，由浏览器解释后作为网页的一部分来执行，需要浏览器中有Java虚拟机的运行。
- JavaApplet与Java 应用程序（Java Application）不同点：
  - JavaApplication需要客户机装有Java虚拟机才能运行，但是只需在客户端安装一次就可多次运行；
  - JavaApplet必须嵌入HTML网页才能运行，且每次都必须从站点重新下载运行（除非客户端有高速缓存）；
  - JavaApplication应用程序比JavaApplet程序大得多，功能强得多，例如RealPlayer；
  - JavaApplet程序都是通过继承Java类库中的JApplet父类来实现的。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

Java Applet

- 利用Applet实现动态网页
  - 当浏览器浏览到一个含有Applet小程序的HTML文件，嵌入的Applet程序代码会自动地从服务器上下载到客户的浏览器中，交给浏览器中的Java虚拟机执行，由此实现人机交互，并能进行必要的商务逻辑处理，实现与后台服务器的交互。
  - 能执行含有Applet网页的程序通常被称为Applet容器，因此，浏览器是一种Applet容器。
  - IE浏览器，在初始状态下不是Applet容器，需要下载安装Applet插件，真正支持Applet动态网页的浏览器是Netscape7。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

Java Applet

- 编写Applet动态网页的步骤
  - 第一步，用Java开发工具编写一个符合Applet规范的Java源程序，源程序文件的扩展名应为.java。
  - 第二步，利用Java编译工具将此Java源程序编译成一个扩展名为.class的类文件。
  - 第三步，编写一个HTML网页，在此网页中，利用“<applet>”标识嵌入此.class的类文件，最后将网页文件和Applet类文件一起发布到网站上。

第四节 动态网页与客户端脚本

2、 客户端脚本技术

Java Applet

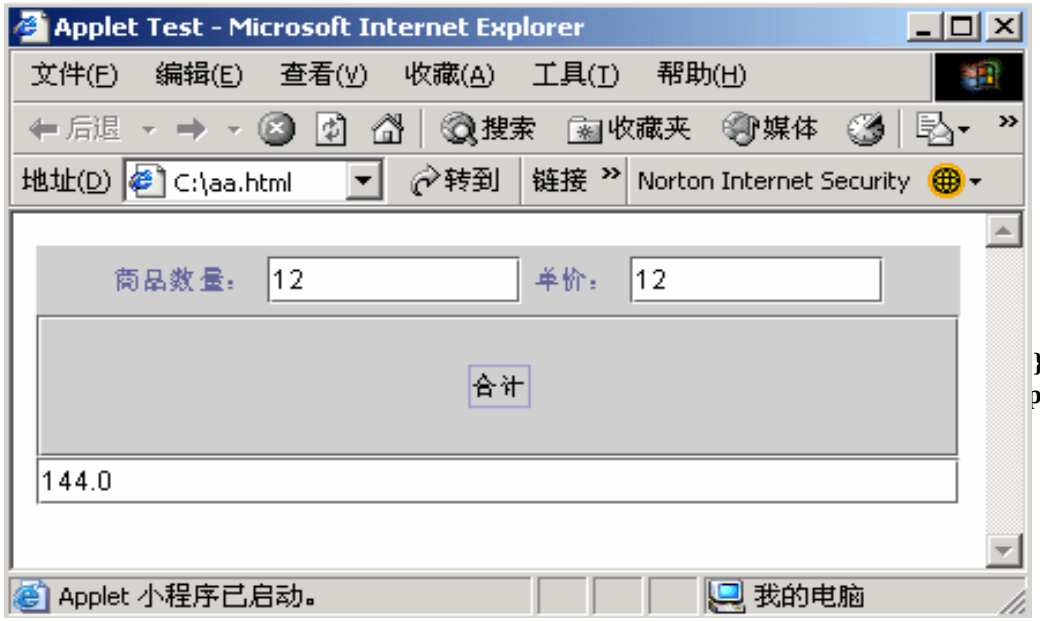
- 例：利用Applet实现对商品总金额进行合计（p80　例2-7）
  - 步骤1：利用记事本编写一个Java源程序；
  - 步骤2 ：利用java的编译工具对源文件CalTest.java进行编译 ；
    - 在命令窗口下输入javac CalTest.java命令，不过要预先配置好jdk1.4的环境变量
  - 步骤3：利用记事本编写一个HTML文件(aa.html)
    - <html>
    - <head>
    - <title> Applet Test </title>
    - </head>
    - <applet code=CalTest.class width=400 height=120 >
    - </applet>
    - </html>
  - 步骤4：使用appletviewer或浏览器浏览aa.html网页

第四节 动态网页与客户

2、 客户端脚本技术

Java Applet

- 例：利用Applet实现对商品总



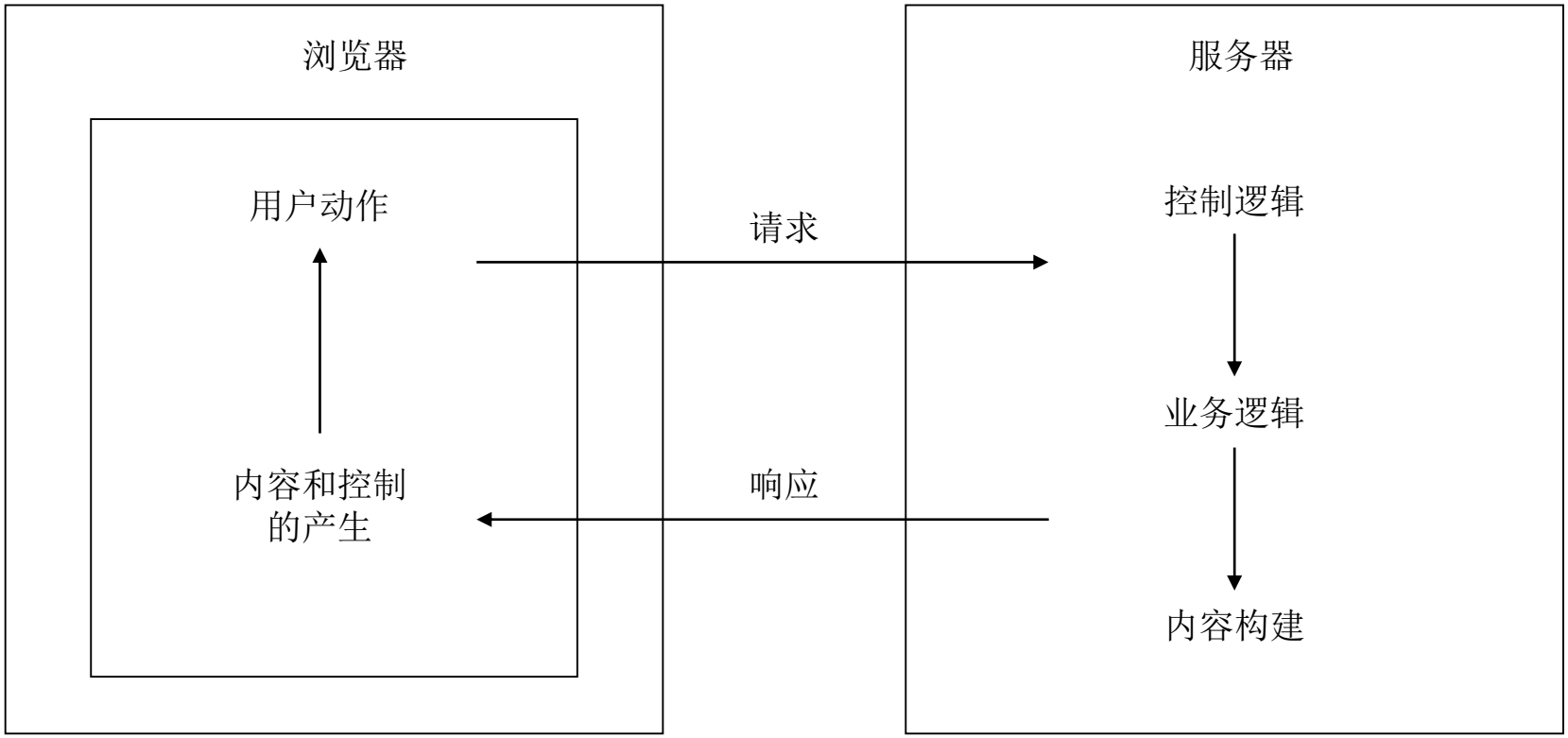

```
//source file name : CalTest.java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class CalTest extends JApplet{
  public   Container cont1;
  public   JLabel  lb1,lb2;
  public   JTextField t1,t2,t3;
  public   JButton b1;
  public   MyEvent1 listen1;
  public   JPanel mypanel;
  public void start()
  {
    mypanel=new JPanel();
    mypanel.setLayout(new FlowLayout());
    cont1=getContentPane();
    cont1.add(mypanel,BorderLayout.NORTH);
    lb1=new JLabel("商品数量：");
    lb2=new JLabel("单价：");
    t1=new JTextField(10);
    t2=new JTextField(10);
    t3=new JTextField(10);
    b1=new JButton("合计");
    listen1=new MyEvent1();
    b1.addActionListener(listen1);
    mypanel.add(lb1);
    mypanel.add(t1);
    mypanel.add(lb2);
    mypanel.add(t2);
    cont1.add(b1,BorderLayout.CENTER);
    cont1.add(t3,BorderLayout.SOUTH);
  }
 private class MyEvent1 implements ActionListener{
    public void actionPerformed(ActionEvent event)
    {
      if(event.getSource()==b1)  //press button1
      {
        double x;
        x=Double.parseDouble(t1.getText())*Double.parseDouble(t2.getText());
        t3.setText(String.valueOf(x));
      }
    }
  }
}
```

第二章 商务表达层及其技术

- 第一节 商务表达层的功能与实现
- 第二节 静态网页的表达及其技术
- 第三节 标记语言
- 第四节 动态网页与客户端脚本
- **第五节 服务器脚本**

第五节 服务器脚本

1、服务器端体系结构



第五节 服务器脚本

2、服务器端脚本

- 服务器端的脚本是指能够在服务器上运行的脚本，其脚本形式与客户端的脚本没有大的差别，也是将脚本直接嵌入到网页中。但不同的是，在网页被传送到浏览器之前，服务器会预先运行网页中的服务器端脚本，运行产生的结果合并到HTML中。
- 与客户端脚本相比优势：
  - 不存在浏览器兼容的问题，所有浏览器收到的都是标准的HTML代码；
  - 可以很容易地访问服务器上的资源；
  - 在存取敏感数据时，较为安全，脚本直接在服务器上执行，身份等敏感信息不可能暴露到客户端上；
  - 简化了客户端的装载，脚本的执行由Web服务器承担。
- 常用的服务器端脚本有JSP、PHP、ASP、CGI、Servlet等。

第五节 服务器脚本

3、PHP

- PHP(Personal Hypertext Preprocessor，个人超文本预处理器)是一种易于学习和使用的服务器端的脚本语言，它将自己的标记嵌入在HTML文件中，其语法大部分是从C、JAVA、PERL语言中借用过来的，并形成了自己的独有风格;
- PHP安装方便，简单易学;
- 在PHP中提供了标准的数据库接口，可以连接多种类型的数据库。

第五节 服务器脚本

4、ASP

- ASP（Active Service Page，活动服务页），是微软开发的一种服务器端的脚本技术，它没有提供自己专门的编程语言，而是允许用户使用包括VBScript、JavaScript等在内的许多已有的脚本语言来编写ASP的应用程序。
- ASP吸收了许多流行的技术，如IIS、ActiveX、VBScript、ODBC等，是一种发展较为成熟的Web应用程序开发技术；其核心功能支持对象组件，通过使用ASP的组件和对象，调用ActiveX控件，可实现强大的商务功能。

第五节 服务器脚本

5、JSP

- JSP（Java Service Pages，Java 服务器页面）是由Sun公司于1999年6月推出的基于Java Servlet以及整个java体系的Web开发技术。利用这一技术可以建立先进、安全和跨平台的动态网站。
- JSP和ASP有许多相似之处：
  - 两者都提供动态网页的技术运行环境，都能实现网页与组件的分离，都能替代CGI，使得动态网页的制作变得简单与快捷。
  - 但两者来源于不同的技术规范与组织， ASP只能运行于Windows平台，JSP可以运行在大多数的服务器平台上，这其中也包括Windows系统，符合“write once，run anywhere”（一次编写，多平台运行）的java标准，能很好地实现平台和服务器的独立性。

第五节 服务器脚本

5、JSP

- JSP是基于Java的技术，用于创建可支持跨平台及跨Web服务器的动态网页。
- JSP使用的是类似于HTML的标记和Java代码片段Scriptlet（HTML标记用于信息显示，Java程序代码用于逻辑处理）。
- JSP代码的执行，需要在Web服务器上安装JSP引擎，从www.java.sun.com下载JSWDK (Java Server Web Development Kit)并安装，安装后执行startserver命令即可启动JSP服务，输入http://localhost:8080可打开默认页面。

第五节 服务器脚本

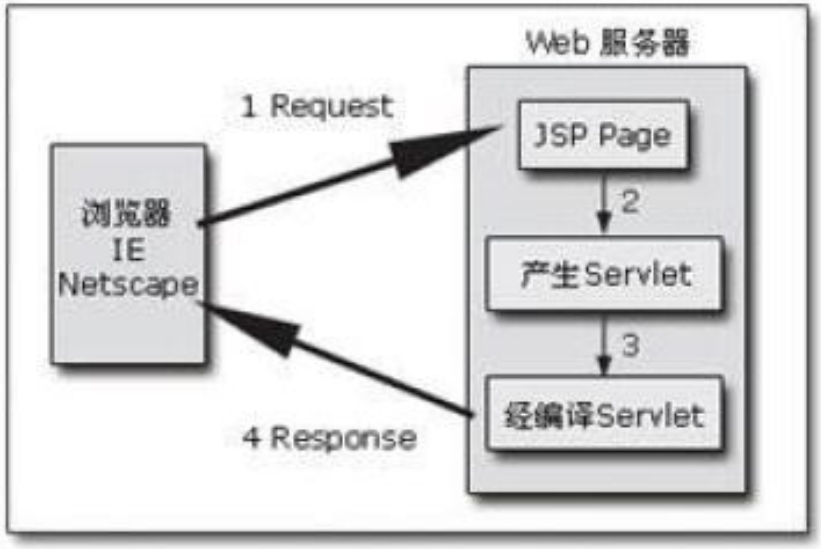
5、JSP

- JSP技术的特点
  - 一次编写，到处执行(Write Once, Run Anywhere)。
  - 搭配可重复使用的跨平台组件，如JavaBean和EJB等。
  - 采用标签化页面开发。Web网页开发人员不一定都是熟悉Java 语言的程序员。JSP技术能够将许多功能封装起来，成为一个自定义的标签，这些功能是完全根据XML的标准来制订的，即JSP技术中的标签库。因此Web页面开发人员可以运用自定义好的标签来达成工作需求，而无须再写复杂的Java语法，让Web页面开发人员能快速开发出动态内容网页。
  - JavaServer Page 技术是J2EE 架构中的一部分，主要负责前端显示经过复杂运算后之结果内容，而分散性的对象系统则是主要依赖EJB等实现。

第五节 服务器脚本

5、JSP

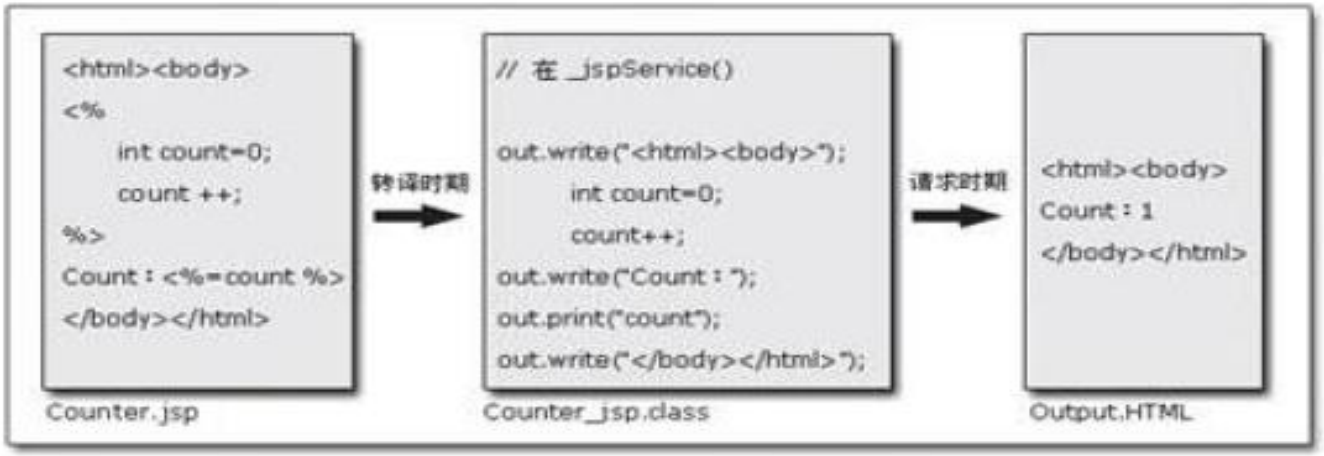
- JSP的执行过程
  - 客户端发出Request (请求)；
  - JSP Container (Tomcat)将JSP 转译成Servlet 的源代码；
  - 将产生的Servlet源代码经过编译后，并加载到内存执行；
  - 把结果Response (响应)至客户端。



第五节 服务器脚本

5、JSP

- JSP的执行过程
  - 在执行JSP 网页时，通常可分为两个时期。
    - 转译时期(Translation Time)：JSP网页转译成Servlet 类。
    - 请求时期(Request Time)：Servlet类执行后，响应结果至客户端。



第五节 服务器脚本

5、JSP

- JSP的基本语法与内部对象
  - 5大元素+8大内部对象

元素	格式
指示符	<%@ direcive %>
声明	<%! declaration %>
表达式	<%= expression %>
代码段	<% code fragment %>
注释	<%-- comment --%>

表 2-7 JSP 中的内部隐含对象

对象	Servlet 类
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
out	javax.servlet.jsp.JspWriter
page	javax.servlet.jsp.JspPage
pageContext	javax.servlet.jsp.PageContext
session	javax.servlet.http.HttpSession.
application	javax.servlet.ServletContext
config	javax.servlet.ServletConfig

第五节 服务器脚本

5、JSP

- JSP的5大元素
  - 指识符Directive：JSP指令的形式为<%@ directive %>，它们只是告诉引擎如何处理其余JSP页面，并不直接产生任何可见输出；JSP中两个最重要的指令是include和page。
    - include指令：其语法格式为<%@ include file="filename" %>，用于在JSP文件中插入文件（JSP文件、HTML文件或JAVA代码等）并解析被插入文件中的JSP语句。

    注意：插入的文件中不能使用<html>，<head>，<body>等标记，否则会影响原文件中的标记并导致错误。

    - page指令：用于定义JSP文件中的全局属性，即无论指令放在JSP文件的哪个地方，它的作用范围都是整个JSP页面，其语法格式见后页。

第五节 服务器脚本

5、JSP

- JSP的5大元素
  - 指识符Directive
    - page指令：

```
<%@ page
[language="java" ]
[extends="package.class" ]
[import="{package.class | package.*}, ..." ]
[session="true | false" ]
[buffer="none | 8kb | sizekb" ]
[autoFlush="true | false" ]
[isThreadSafe="true | false" ]
[info="text" ]
[errorPage="relativeURL" ]
[contentType="mimeType [ ;charset=characterSet ]" | "text/html ;
            charset=ISO-8859-1" ]
[isErrorPage=“true | false” ]
%>
```

第五节 服务器脚本

5、JSP

- **JSP的5大元素**
  - **指识符Directive**
    - **page指令：属性说明**
    - **language="java"：**声明脚本语言的种类，暂时只能用"java"；
    - **extends="package.class"：**标明JSP编译时需要加入的Java Class的全名，但是得慎重使用，它会限制JSP的编译能力；
    - **import="{package.class | package.* }, ..."：**声明需要导入的Java包的列表，这些包就作用于程序段，表达式，以及声明（ java.lang.* 、 javax.servlet.* 、 javax.servlet.jsp.* 和 javax.servlet.http.* 包在JSP编译时已经导入了，所以不需要再指明了）；
    - **session="true | false"：**设定客户是否需要HTTP Session，如果它为true,那么Session是有用的；如果它有false，那么就不能使用session对象，以及定义了scope=session的<jsp:useBean>元素；缺省值是true。

第五节 服务器脚本

5、JSP

- **JSP的5大元素**
  - **指识符Directive**
    - **page指令：属性说明**
    - **buffer="none | 8kb | sizekb"**：buffer的大小被out对象用于处理执行后的JSP对客户浏览器的输出，缺省值是8kb；
    - **autoFlush="true | false"**：设置如果buffer溢出，是否需要强制输出，如果其值被定义为true（缺省值）则输出正常；如果它被设置为false，如果这个buffer溢出，就会导致一个意外错误的发生；
    - **isThreadSafe="true | false"**：设置JSP文件是否能多线程使用，缺省值是true，也就是说JSP能够同时处理多个用户的请求；如果设置为false，一个JSP只能一次处理一个请求；
    - **info="text"**：一个文本在执行JSP将会被逐字加入JSP中；
    - **errorPage="relativeURL"**：设置处理异常事件的JSP文件；
    - **isErrorPage="true | false"**：设置此页是否为出错页，如果被设置为true，就能使用exception对象；
    - **contentType="mimeType [;charset=characterSet]" | "text/html;charset=ISO-8859-1"**：设置MIME类型，缺省MIME 类型是: text/html，缺省字符集为 ISO-8859-1。

第五节 服务器脚本

5、JSP

- JSP的5大元素
  - 声明Declaration：JSP声明的形式为<%! declaration %>，用来定义页面级变量以保存信息，或者定义JSP页面的其余部分可能需要的支持方法；声明一定要以分号（;）结束，因为任何内容都必须是有效的Java语句，例如<%! int i=0; %>，声明了一个整型变量i，其初始值为0。
  - 表达式 Expressions：JSP表达式的形式为<%= expression %>，用来包含一个符合JSP语法的表达式，一般可以用来输出变量的值；表达式没有分号，除非在加引号的字符串部分使用分号，例如<%= i %>、<%= "Hello;" %>。

第五节 服务器脚本

5、JSP

- JSP的5大元素
  - 代码段Code Fragment/脚本段Scriptlet：JSP脚本段的形式为<% code fragment %>，用来包含有效的JAVA程序段，这些Java代码在Web服务器响应请求时就会运行；在脚本片段周围可能是原始的HTML或XML语句，在这些地方，代码片段可以使设计者创建条件执行代码，或要用到另外一段代码的代码。例如以下的代码组合使用表达式和代码片段，显示H1、H2、H3和H4标记中的字符串“Hello”：

```
<% for (int i=1; i<=4; i++) { %>
    <H<%=i%>>Hello</H<%=i%>>
<% } %>
```

  - 注意：代码片段并不局限于一行源代码。

第五节 服务器脚本

5、JSP

- JSP的5大元素
  - 注释Comments：JSP注释的形式为<%-- comment --%>，如果使用HTML注释，用户在查看页面源代码时就可以看到；如果不想让用户看到注释就应该将其嵌入“<%--”和“--%>”标记中，例如<%-- comment for server side only --%>。

第五节 服务器脚本

5、JSP

- JSP的内部隐含对象

JSP中可用的隐含对象包括：

- request请求对象：当客户端请求一个JSP网页时，JSP引擎会将客户端的请求信息包装在这个request对象中，请求信息的内容包括请求的标题头(Header)、信息(如浏览器的版本信息、语言和编码方式等)，请求的方式(如HTTP方法：GET、POST、PUT等)，请求的参数名称、参数值和客户端的主机名称等；可以通过这个对象来取得有关客户端的请求信息或者获取请求对象传递过来的参数值。

第五节 服务器脚本

5、JSP

- JSP的内部隐含对象
  - response响应对象：代表对客户端的响应，组织传回客户端的数据，如回应的Header、回应本体（HTML的内容）以及服务器端的状态码等信息。
  - pageContext页面上下文对象：代表当前页面运行的一些属性，常用的方法包括findAttribute、getAttribute和getAttributeScope等。
  - session会话对象：代表服务器与客户端所建立的会话，可以使用它来保存某个特定客户端（访问者）一次访问的一些特定信息，或者跟踪访问者的访问路径，从中挖掘有用的信息，如访问者的兴趣爱好、访问目的。

第五节 服务器脚本

5、JSP

- JSP的内部隐含对象
  - application应用程序对象：提供应用程序在服务器中运行时的一些全局信息，用于说明代码片段的运行环境；与session对象相比，application对象是所有客户共享的，而session对象则是每个客户专用的；该对象一旦创建就将一直保持，除非服务器关闭。
  - out输出对象：传送响应的输出流，与response的不同之处在于通过out发送的内容将是浏览器要显示的内容，是文本一级的，它可以直接向客户端写一个由程序动态生成的HTML文件。

第五节 服务器脚本

5、JSP

- JSP的内部隐含对象
  - config配置对象：提供存取servlet class的初始参数及有关Server环境的信息，config对象的范围是page。
  - page页面对象：JSP网页本身，代表了正在运行的JSP文件产生的类对象。
  - exception例外对象：代表有错的网页中未被捕获的例外，用于获得错误信息。

第五节 服务器脚本

5、JSP

- JSP的内部隐含对象
  - JSP中隐含对象的作用：在JSP代码片段中，可以利用隐含对象与JSP页面的代码片段执行环境产生互动；在JSP脚本片段里，可使用内置隐含对象进入执行JSP代码的代码片段。
  - 一般来说，应尽量少访问内置隐含对象，但是在一某些情况下，访问隐含对象是可被接受的。
  - 内置隐含对象使用实例：
    - 不用表达式，直接进入“Out”隐含对象，将某些内容输出到响应中<% out.println("Hello"); %>；
    - 可以从请求对象获取参数值<% String name=request.getParameter("name"); out.println(name); %>。

第五节 服务器脚本

JSP脚本示例1

指示符
指明脚本语言和字符编码

声明
声明字符串变量msg

代码段
具体的程序代码

表达式
显示变量值

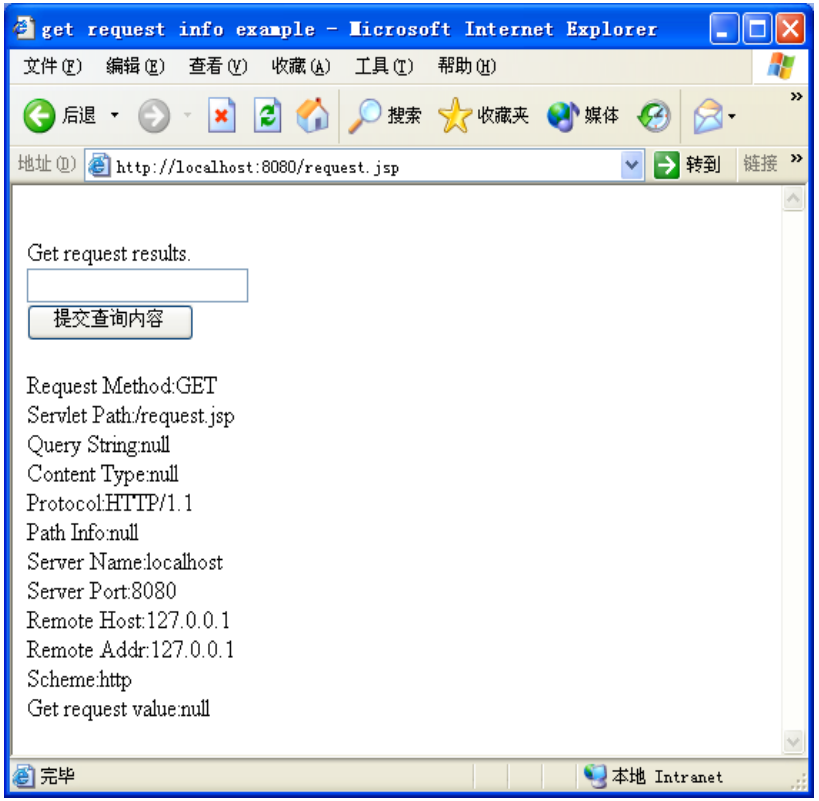
注释部分

```
<%@page language="java" contentType="text/html;charset=gbk"%>
<HTML>
 <head >
  <title>Hello World!</title>
 </head>
<body bgcolor="#FFFFFF">
  <%!String msg="一个简单的JSP";%>
  <%
    out.println("Hello World!");
    out.println("Hello China!");
    out.println(msg);
  %>
  <br>
  <%=msg%>
  <%--显示变量值--%>
</body>
</HTML>
```

第五节 服务器脚本

JSP脚本示例2 - Request对象使用（p90-91）

```
<html>
<head><title>get request info example </title> </head>
<body>
 <form action="req1.jsp">
   <br>Get request results.
   <br><input type="text" name="myname">
   <br><input type="submit" name="get value">
 </form>
Request Method:<%=request.getMethod()%>
<br>Servlet Path:<%=request.getServletPath()%>
<br>Query String:<%=request.getQueryString()%>
<br>Content Type:<%=request.getContentType()%>
<br>Protocol:<%=request.getProtocol()%>
<br>Path Info:<%=request.getPathInfo()%>
<br>Server Name:<%=request.getServerName()%>
<br>Server Port:<%=request.getServerPort()%>
<br>Remote Host:<%=request.getRemoteHost()%>
<br>Remote Addr:<%=request.getRemoteAddr()%>
<br>Scheme:<%=request.getScheme()%>
<br>Get request
value:<%=request.getParameter("myname")%>
</body>
</html>
```



第五节 服务器脚本

6、三种常用服务器脚本语言的比较

- JSP与ASP的比较
  - Sun公司的JSP和Microsoft的ASP在技术方面有许多相似之处。
    - 两者都为动态网页的技术；
    - 双方都能够替代CGI技术，使网站的开发时程能够大大缩短；
    - 在性能上有较高的表现；
    - 更重要的是，两者都能够提供组件设计的功能，通过组件设计，将网页中逻辑处理部分交由组件负责处理(ASP使用COM组件、JSP使用JavaBean 组件)，而和网页上的排版、美工分离。
  - 尽管JSP技术和ASP在许多方面都很相似，但仍然存在很多不同之处，本质区别在于两者来源于不同的技术规范组织。
  - 以下比较两大技术的不同点和各自带来的优势。

第五节 服务器脚本

6、三种常用服务器脚本语言的比较

- JSP与ASP的比较
  - 平台和服务器的弹性（JSP>ASP）
    - JSP（JVM+UNIX）> ASP（Windows+IIS+ASP的体系结构）
    - WEB网页程序员未来在开发电子商务平台时，不需要再考虑客户厂商的操作系统平台，可更专心于系统功能的开发。厂商在使用JavaServer Pages技术开发的系统平台时，不再需要担心未来在扩充软硬件时，是否产生不兼容的问题。
  - 语法结构（JSP>ASP）
    - ASP语法结构上，是以“<%”和“%>”作为标记符号，而JSP也是使用相同标记符号作为程序的区段范围的。但不同的是，标记符号之间所使用的语言：ASP为JavaScript或VBScript；而JSP为Java。Java易扩充，优于VBScript语言。
    - Java使程序员的工作变得容易、简单。例如：当ASP应用程序在Windows NT系统可能会造成系统当机，由于JSP是在JVM上执行程序，且提供强大的异常事件处理机制，因此，不会因为程序撰写的疏忽，而导致服务器操作系统的损毁。
    - Java语言提供防止直接存取内存的功能，存取内存产生的错误，通常也正是造成服务器损毁的最主要原因之一。
    - Java语言是一个有严谨规范、有系统组织的语言。

第五节 服务器脚本

6、三种常用服务器脚本语言的比较

- JSP与ASP的比较
  - 开放的开发环境（JSP>ASP）
    - J2EE成为业界标准（IBM+Oracle+BEA+Apache）
    - ASP仅靠Microsoft的平台支持
  - 语法的延展性（JSP>ASP）
    - ASP和JSP都使用标签与Scripting Language来制作动态WEB 网页，JavaServer Pages 2.0新规范中，能够让程序员自由扩展JSP标签来应用。
    - JSP开发者能自定义标签库，能充分利用与XML兼容的标签技术的功能，减低对Java语法的依赖，也可以利用XML强大的功能，做到数据、文件格式的标准化。
  - 执行性能表现（JSP>ASP）
    - JSP除了在一开始加载的时间会比较久外，之后的表现优于ASP。原因在于：JSP在一开始接受到请求时，会产生一份Servlet实体，它会先被暂存在内存中，当再有相同请求时，这实体会产生一个线程来服务它。如果过了一段时间都不再用到此实体时，Container会自动将其释放。而ASP在每次接收到请求时，都必须要重新编译。因此，JSP的执行比每次都要编译执行的ASP快，尤其是程序中存在循环操作时，JSP的速度要快1到2倍。
    - ASP这部分的缺陷随ASP.NET的出现有所改观，性能表现上有很大的突破。

第五节 服务器脚本

6、三种常用服务器脚本语言的比较

- 三种服务器脚本的开发选择
  - 跨平台性/多种Web服务器和数据库支持
    - JSP/PHP可以跨平台并且支持多种Web服务器和数据库，而ASP只能支持Windows系统+IIS+Win32平台下的数据库。
  - 开发周期/维护难度
    - ASP/PHP开发周期短，维护容易，而JSP中等。
  - 安全性/开放性
    - ASP安全性不好，JSP/PHP安全性较好，PHP开源。
  - 组件支持（大型网站构建）
    - JSP通过JavaBean组件可实现复杂的逻辑控制。
    - ASP通过COM/DCOM组件实现复杂的逻辑控制。
    - PHP组件支持方面还比较差，不适合大型系统。

第五节 服务器脚本

7、商务表达平台体系结构综合比较

特　性	客户端应用	客户端脚本	服务器端脚本
导航、定位	效果好	比较强、稍显简陋	能力差
客户端资源	大	较多	少
服务器资源	少（适度依赖）	较高	高
响应时间	快（第一次下载最大）	较快、下载时间长	长
信息传输	少（无布局信息）	较少	大量
安全性	少量控制	极少	较安全
兼容性	少量问题	许多问题	最好
应用特点	用户群稳定、界面业务复杂	用户范围广、交互多	用户范围广、交互少
应用举例	企业内部网应用	网页浏览	掌上电脑

本章小结

- 商务表达层的任务、功能和三种实现方式
- 标记语言（重点是HTML，XML的基本编写方法）
- XML的用途、XML开发步骤
- 静态网页与动态网页的区别
- 客户端脚本程序与服务器端脚本的区别
- JavaScript语言与Java的区别与联系
- 客户端脚本（JavaScript、Java Applet）
- 服务器端脚本（ JSP）
- 三种服务器端脚本（ASP/PHP/JSP）的比较与选择

Thank You !