1. Introduction

1. **Internet&&Web**
   - **Internet:**是在一个通信网络中连接的计算机的大规模集合，通过装置连接起来，相互之间可以通信
   - **World wide web（web）：**是一组软件和协议的集合，Internet 中绝大部分甚至全部计算机都安装了这些软件和协议，用户通过 web 来使用 Internet。Web 中的信息单元包括：页面，文档和资源。包括 web 服务器，web 客户机或浏览器。

     特点：

     图形化，易于导航（页面之间相互连接）

     与平台无关

     分布式

     动态

     交互

     Web 客户机程序:　浏览器——获取和显示服务器端传回的文档

     Web 传输协议：　最常用的 HTTP

2. **TCP/IP:**传输控制协议，1982 年成为计算机网络连接的标准协议，它允许某台计算机中的程序通过 Internet 直接和另一台计算机中得程序进行通讯。

   TCP/IP 由四个层次组成：网络接口层、网络层、传输层、应用层。

   OSI（Open System Interconnect）是传统的开放式系统互连参考模型，是一种通信协议的 7 层抽象的参考模型，其中每一层执行某一特定任务。该模型的目的是使各种硬件在相同的层次上相互通信。这 7 层是：物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。

3. **Lamp:linux-apache-mysql-php**

   Linux+Apache+Mysql+Perl/PHP/Python 一组常用来搭建动态网站或者服务器的开源软件，本身都是各自独立的程序，但是因为常被放在一起使用，拥有了越来越高的兼容度，共同组成了一个强大的 Web 应用程序平台。随着开源潮流的蓬勃发展，开放源代码的 LAMP 已经与 J2EE 和.Net 商业软件形成三足鼎立之势，并且该软件开发的项目在软件方面的投资成本较低，因此受到整个 IT 界的关注。从网站的流量上来说，70%以上的访问流量是 LAMP 来提供的，LAMP 是最强大的网站解决方案.

   Lamp system:redhat,suse,debian,FreeBSD,solaris,yellow dog linux,mac os

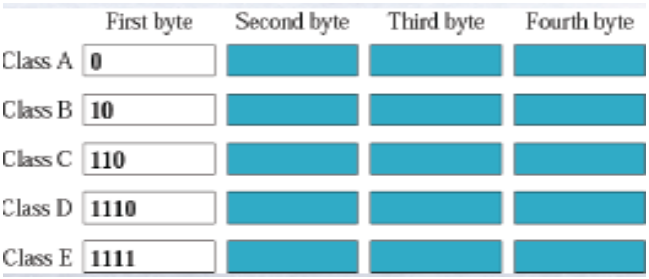
   Mysql:快速，免费，稳定

4. **“WISA,” Windows-IIS-SQL Server-ASP (and**

   now, ASP.Net).

5. **IP**
   - IP 是英文 Internet Protocol（网络之间互连的协议）的缩写，中文简称为“网协”，也就是为计算机网络相互连接进行通信而设计的协议，用来唯一标识 Internet 中的节点。

- IP 地址具有唯一性，根据用户性质的不同，可以分为 5 类。
  A 类保留给政府机构 1.0.0.1---126.255.255.254
  B 类分配给中等规模的公司：128.0.0.1---191.255.255.254
  C 类分配给任何需要的人：192.0.0.1---223.255.255.254
  D 类用于组播，E 类用于实验，各类可容纳的地址数目不同。224.0.0.1---239.255.255.254
  E 类地址，用于实验：240.0.0.1---255.255.255.254



- IP 地址是一个唯一的 32 位数字
  IPV6 128 位

6. **DNS**
   - **定义：**DNS（Domain Name Server）是一个分布式数据库，本地负责控制整个分布式数据库的部分段，每一段中得数据通过客户/服务器模式在整个网络上均可存取。负责将域名转化为 IP 地址。
   - **DNS 数据库结构：**倒立的树状结构，根用“”空字符串表示，每隔一节点就是一个域，每个域可以划分为多个子域，叶节点代表主机。
     每个域分别由不同的组织进行管理。每个组织都可以将它的域再分成一定数量的子域并将这些子域委托给其他组织进行管理，域既包括主机又能包括它的子域。域名被用做 DNS 数据库中的索引，域包含所有域名在该域的主机。
   - **它是由解析器和域名服务器组成的。**
     域名服务器：是指保存有该网络中所有主机的域名和对应 IP 地址，并具有将域名转换为 IP 地址功能的服务器。
     域名解析：将域名解析为对应的 IP 地址的过程。
   - **工作原理：**
     1．客户机将域名查询请求发送到本地 DNS 服务器，服务器在本地数据库中查找客户机要求的映射。
     2．如果不能在本地找到客户机查询的信息，将客户机请求发送到根域名服务器。根域名服务器负责解析客户机请求的根域部分，它将包含下一级域名信息的服务器的地址返回给客户机的 DNS 服务器。
     3．客户机的 DNS 服务器利用根域名服务器解析的地址访问下一级 DNS 服务器，得到维护再下一级域名的 DNS 服务器的地址。
     4．按照上述方法递归地逐级接近查找目标，最后在维护目标域名的 DNS 服务器上找到相应

的 IP 地址信息。

5．客户机的本地 DNS 服务器将查询结果返回客户机。

6．客户机利用从本地 DNS 服务器查询得到的 IP 地址访问目标主机。

7. Web 服务器

1) **定义**：web 服务器是可以向发出请求的浏览器提供文档的程序。是一种被动程序，只有当 Internet 上运行的其他计算机中得浏览器发出请求时服务器才响应。

2) **工作原理**：

- web 浏览器通过向服务器发送 URL 来启动与服务器之间的通讯，一个 URL 可以指定：1）存储在服务器中得某个数据文件的地址，该文件会发送给客户机 2） 客户机要求执行的位于服务器中得某个程序，程序执行结果返回客户机。
- **Web 服务器**：监控主机的通信端口，通过该端口接收 HTTP 命令，运行该命令指定的操作。HTTP 命令包含一个 URL，其中包含主机名称。
- **Web 服务文档结构：1）文档根目录**：存储服务器能直接访问的为客户机提供的 web 文档 2）**服务器根目录**：存储服务器端程序和软件

3) 常用

Apache：当前最广泛的 web 服务器，开源免费，通过配置文件（http.conf）进行控制。

IIS:**windows 平台下，通过窗口管理程序来控制服务器参数**

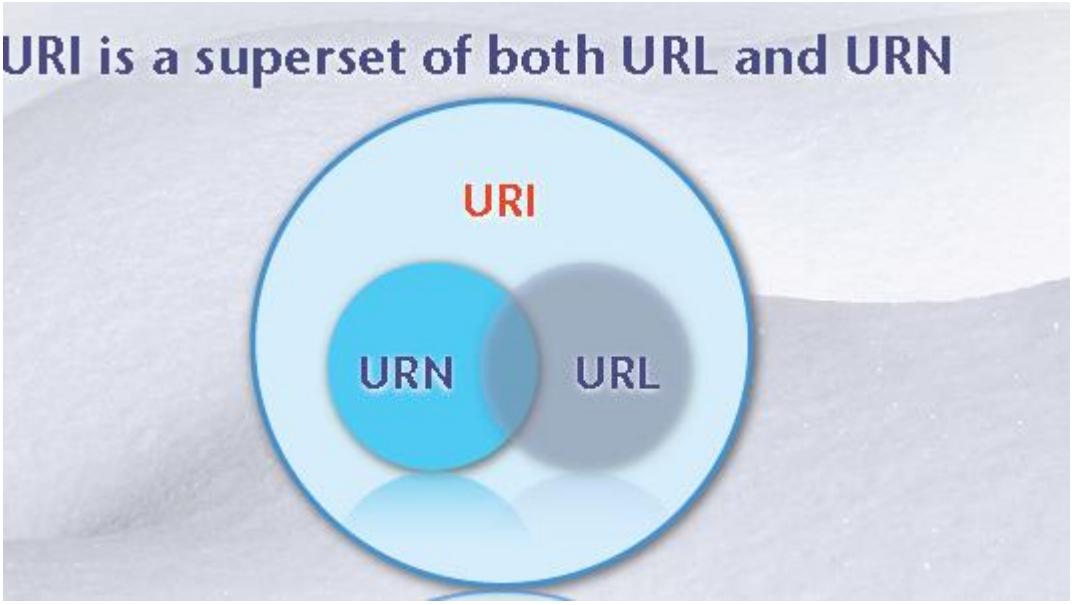
8. URI,URL,URN

- **URI**

  Web 上可用的每种资源 - HTML 文档、图像、视频片段、程序等 - 由一个通用资源标志符（Uniform Resource Identifier, 简称"URI"）进行定位。

  以某种统一的（标准化的）方式标识资源的简单字符串。

  组成：存放资源的主机名，片段标识符，相对 URI



- **URL** Uniform Resource Location

  统一资源定位符

  URL 是 Internet 上用来描述信息资源的字符串，主要用在各种 WWW 客户程序和服务器程序上，特别是著名的 Mosaic。采用 URL 可以用一种统一的格式来描述各种信息资源，包括

文件、服务器的地址和目录等。

格式：scheme：object-address

协议，主机地址或 IP，端口号，资源具体地址

Scheme：协议 http，ftp，telnet，mailto 等

Object-address：//完全限定域名/文档路径

路径：目录名称和文件名称组成的一组序列，通过“/”分隔，不能有空格，分号，冒号，&符号。可以使完整路径或部分路径

- **URN：Uniform Resource Name，统一资源名称。**
- **关系：URL,URN 是 URI 的子集**

Web 上地址的基本形式是 URI,它代表统一资源标识符。有两种形式：

URL：目前 URI 的最普遍形式就是无处不在的 URL 或统一资源定位器。

URN：URL 的一种更新形式，统一资源名称(URN, Uniform Resource Name)不依赖于位置，并且有可能减少失效连接的个数。但是其流行还需假以时日，因为它需要更精密软件的支持。

9. 万维网联盟 万维网联盟（World Wide Web Consortium，W3C）

万维网联盟是国际著名的标准化组织。1994 年成立后，至今已发布近百项相关万维网的标准，对万维网发展做出了杰出的贡献。

10. web1.0,2.0,3.0

Web X.0 并不是一个具体的事物，而是一个阶段，是促成这个阶段的各种技术和相关的产品服务的一个称呼。

- **Web1.0,**是以编辑为特征,网站提供给用户的内容是网站编辑进行编辑处理后提供的，用户阅读网站提供的内容。这个过程是网站到用户的单向行为，web1.0 时代的代表站点为新浪，搜狐，网易三大门户。主要是静态网页。

  主要是指互联网一开始就有的那些运营模式，例如：信息发布网站，门户网站等。特点是：信息由网站的运行维护者发布，用户一般只能查看和评论。

  网络资源从信息生产者到信息客户的单向流程。

- **Web2.0** 则是以加强了网站与用户之间的互动，网站内容基于用户提供，网站的诸多功能也由用户参与建设，实现了网站与用户双向的交流与参与,web2.0 不同于 web1.0 的**最大之处在于它的交互性**。这个时期的典型代表有：博客中国、亿友交友、联络家等。

  注重积用户之力来创造内容，网站的运行维护者只是提供一个平台，而主要内容是用户来创造，用户是主角，网站不再是主角。

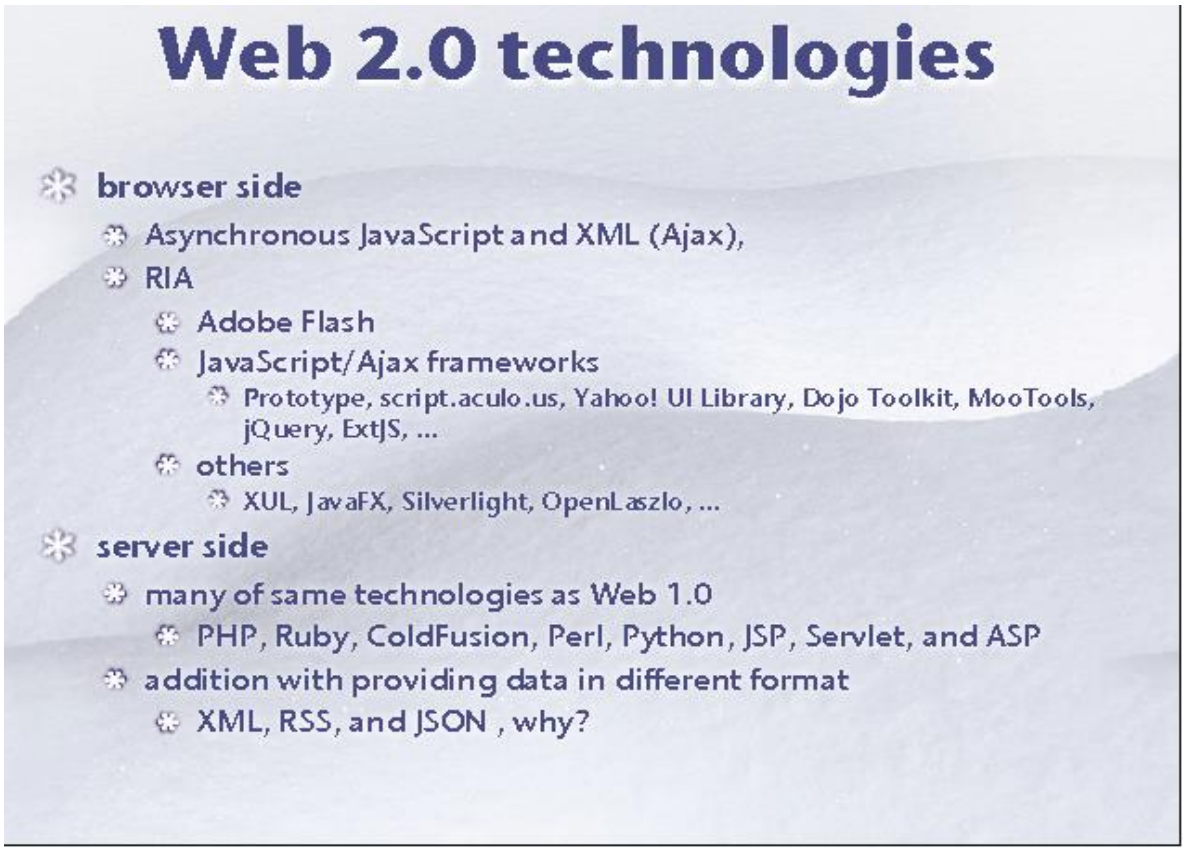
  所有人是网络资源的生产者和信息客户。

联系：无论是 web1.0 还是 web2.0，它们都是在窄带互联网这样一个大环境下，基于内容与交互的信息模式，只是 web2.0 在表现形式上比 web1.0 丰富一些，加强了用户的参与度。

- **Web3.0：**随着网络基础设施的建设和信息技术的深入发展，互联网会更深层次的进入到人们的日常生活。web3.0 是以主动性（Initiative）、数字最大化（max-Digitalizative）、多维化（multi-dimension）等为特征的，以服务为内容的第三代互联网系统。

  web 3.0 最明显的特征就是主动性，即强调网站对用户需求的主动提取，并加以分析处理，然后给出用户所需要的资源。

GoogleCEO 埃里克施密特：web3.0 是一系列组合在一起的应用，对于个人用户来讲，互联网将更具有管理性，互联网将由一系列标准化的组件拼接起来。智能语义程序介入网络资源流程，更有针对性的发送信息和获取信息。



2. HTTP

2.1. 概念

1. Web 页面是对象构成，web 对象通过 URL 定位
2. hypertext transport protocol;
3. HTTP 定义：一种详细规定了浏览器和万维网服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议

4. **特点**

支持客户/服务器模式。

简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有 GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于 HTTP 协议简单，使得 HTTP 服务器的程序规模小，因而通信速度很快。

灵活：HTTP 允许传输任意类型的数据对象。正在传输的类型由 Content-Type 加以标记。

无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客

户的应答后，即断开连接。采用这种方式可以节省传输时间。

无状态：HTTP 协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

应用层协议,基于 TCP/IP 协议

5. **Request For Comments（RFC）**是一系列以编号排定的文件。文件收集了有关互联网相关信息，以及 UNIX 和互联网社区的软件文件。目前 RFC 文件是由 Internet Society（ISOC）赞助发行。基本的互联网通信协议都有在 RFC 文件内详细说明。RFC 文件还额外加入许多的论题在标准内，例如对于互联网新开发的协议及发展中所有的记录。因此几乎所有的互联网标准都有收录在 RFC 文件之中。

6. **结构:**

   Client sends request

   Server respose request

   支持多请求-响应

7. **HTTP 延迟影响因素：**

   The TCP connection setup handshake 执行三向沟通连接法

   TCP slow-start congestion control 低速启动算法

   Nagle's algorithm for data aggregation Nagel 数据压缩算法

   TCP's delayed acknowledgment algorithm TCP 认证算法

   for piggybacked acknowledgments

   TIME_WAIT delays and port exhaustion

2.2. HTTP 请求

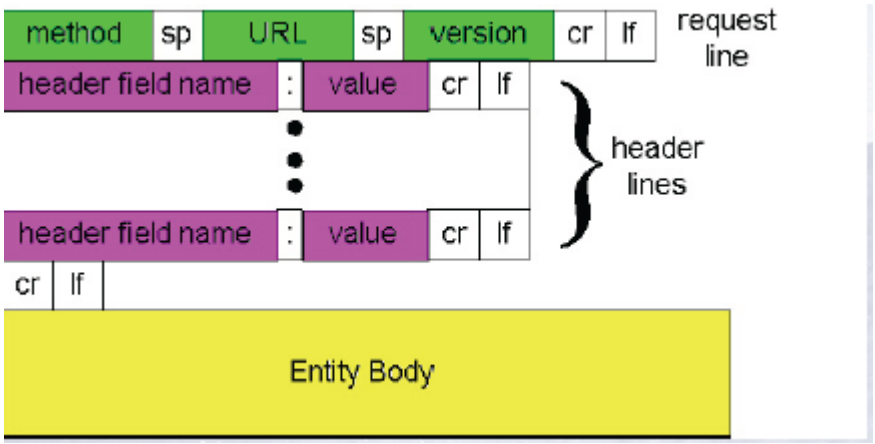
1. **通用格式：**

   HTTP 方法 URL 当中的域名部分 HTTP 版本

   头部字段

   空行

   消息主体



2. **请求方法**

OPTIONS 返回服务器针对特定资源所支持的 HTTP 请求方法。也可以利用向 Web 服务器发送'*'的请求来测试服务器的功能性。

HEAD : retrieve meta-information about the URI 返回指定文档的头部信息

GET：返回指定的文档内容

POST：提交数据进行处理，数据被包含在请求体中，POST 请求会导致新资源的建立或已有资源的修改。 常用于发送表单，或发送请求执行某个程序处理表单

PUT 向指定资源位置上传其最新内容。

DELETE 请求服务器删除 Request-URI 所标识的资源。

TRACE 回显服务器收到的请求，主要用于测试或诊断。

2.3. HTTP 响应

1. 通用格式

状态行

响应头部字段

空行

响应主体

2. 返回的状态码

✓ **Codes**

1xx! Informational

2xx! Success

3xx! Redirection 重定向

4xx! Client Error

5xx! Server Error

Common Status Codes

- ✓ **200 ! OK**
- ✓ **301! Moved Permanently**
- ✓ **400 ! Bad Request**
- ✓ **401 ! Unauthorized**
- ✓ **403 ! forbidden**
- ✓ **404 ! Not Found**
- ✓ **500 ! Internal Server Error**
- ✓ **100 ! Continue**
- ✓ **101 ! switching protocols**

Success status codes

200 OK
201 Created
202 Accepted
203 Non-Authoritative Information
204 No Content
205 Reset Content
206 Partial Content

Redirection status codes

300 multiple choices
301 Moved Permanently
302 Found
303 See Other
304 Not Modified
305 Use Proxy
307 Temporary Redirect

Client Error Status Codes

400 Bad Request
401 Unauthorized
403 Forbidden
404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
409 Conflict
410 Gone

Client Error Status Codes (cont.)

411 Length Required
412 Precondition Failed
413 Request Entity Too Large
414 Request URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed

Server Error Status Codes

500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported

2.4. HTTP 连接

2.4.1. 非持续连接——HTTP1.0

在使用 HTTP/1.0 的情况下，如果打开一个包含一个 HTML 文件和 10 个内联图象对象的网页时，HTTP 就要建立 11 次 TCP 连接才能把文件从服务机传送到客户机。使用一次 TCP 连接传送一个对象的效率比较低，这体现在下列几个方面：

(1) **每次 TCP 连接必需要建立和断开。**客户机和服务机建立一次连接需要执行三向沟通连接法(three-way handshake)，服务机在对象递送之后要断开 TCP 连接。在建立和断开连接时要占用 CPU 的资源。如果使用一次连接代替 11 次连接的话，占用客户机和服务机的 CPU 时间可大大减少。

(2) **对每次连接，客户机和服务机都必须分配发送和接收缓存**。这就意味着要影响客户机和服务机的存储器资源，这同样要占用 CPU 的时间。

(3) **对由大数量对象组成的文件，TCP 的低速启动算法(slow start-up algorithm)会限制服务机向客户机传送对象的速度。**使用 HTTP/1.1 之后，大多数对象都可以尽最大的速率传送。
非持久连接每个对象的发送和接收都会有一个 HTTP 延迟，共有两个 HTTP 延迟。

2.4.2. 持续连接

持久连接：http1.1 版本中：当发送请求并响应之后，服务器和客户端浏览器之间依然保持连接，文件中的所有对象都可在相同的 TCP 连接上传送。

1. 不带流水线 without pipelineing
那么客户只在收到前一个请求的响应后才发出新的请求。这种情况下，web 页面所引用的每个对象(上例中的 10 个图像)都经历 1 个 RTT 的延迟，用于请求和接收该对象。与非持久连接 2 个 RTT 的延迟相比，不带流水线的持久连接已有所改善。
缺点：服务器送出一个对象后开始等待下一个请求，而这个新请求却不能马上到达。这段时间服务器资源便闲置了。

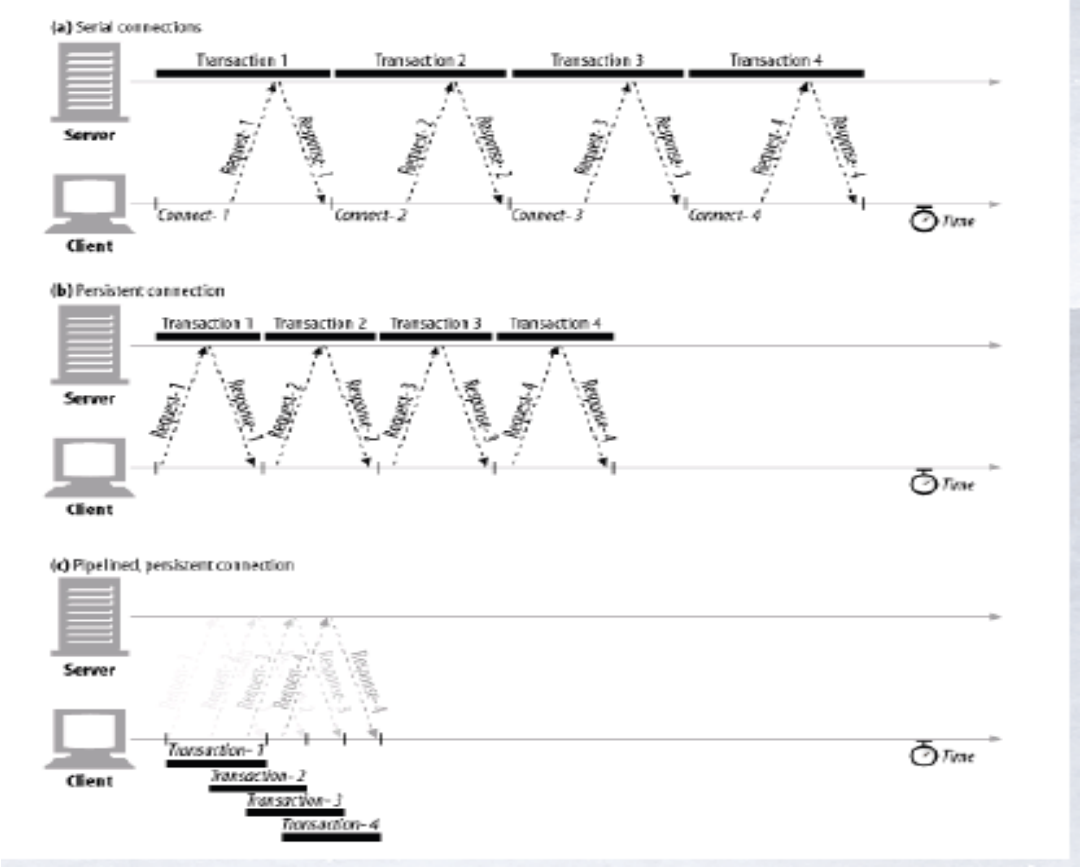
2. 带流水线——HTTP1.1
HTTP/1.1 的默认模式使用带流水线的持久连接。HTTP/1.1 也允许在客户机接收到服务机的消息响应之前发送多个消息请求，这叫做流水线式请求(pipelined request)。

- **可以连续发送请求和响应对象。**HTTP 客户每碰到一个引用就立即发出一个请求，HTTP 客户可以一个接一个紧挨着发出各个引用对象的请求。服务器收到这些请求后，也可以一个接一个紧挨着发出各个对象。如果所有的请求和响应都是紧挨着发送的，那么

所有引用到的对象一共只经历 1 个 RTT 的延迟(而不是像不带流水线的版本那样，每个引用到的对象都各有 1 个 RTT 的延迟)。

- **带流水线的持久连接中服务器空等请求的时间比较少**。
- **降低缓启动延迟**。与非持久连接相比，持久连接(不论是否带流水线)除降低了 1 个 RTT 的响应延迟外，缓启动延迟也比较小。其原因在于既然各个对象使用同一个 TCP 连接，服务器发出第一个对象后就不必再以一开始的缓慢速率发送后续对象。相反，服务器可以按照第一个对象发送完毕时的速率开始发送下一个对象。

HTTP/1.1 都有非持续连接(non-persistent connection)和持续连接(persistent connection)功能。
HTTP/1.1 的默认设置是持续连接。
HTTP/1.0 的默认设置是非持续连接



2.5. HTTPS

1. **定义**

它是由 Netscape 开发并内置于其浏览器中，用于对数据进行压缩和解压操作，并返回网络上传送回的结果。HTTPS 实际上应用了 Netscape 的安全套接字层（SSL）作为 HTTP 应用层的子层。（HTTPS 使用端口 443，而不是像 HTTP 那样使用端口 80 来和 TCP/IP 进行通信。）SSL 使用 40 位关键字作为 RC4 流加密算法，这对于商业信息的加密是合适的。HTTPS 和 SSL 支持使用 X.509 数字认证，如果需要的话用户可以确认发送者是谁。

也就是说它的主要作用可以分为两种：一种是建立一个信息安全通道，来保证数据传输的安全；

另一种就是确认网站的真实性。

2. HTTPS 和 HTTP 的区别

一、https 协议需要到 ca 申请证书，一般免费证书很少，需要交费。

二、http 是超文本传输协议，信息是明文传输，https 则是具有安全性的 ssl 加密传输协议。

三、http 和 https 使用的是完全不同的连接方式，用的端口也不一样，前者是 80，后者是 443。

四、http 的连接很简单，是无状态的；HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 http 协议安全。

3. HTTPS 解决的问题

1）信任主机问题

采用 https 的服务器必须从 CA （Certificate Authority）申请一个用于证明服务器用途类型的证书。该证书只有用于对应的服务器的时候，客户端才信任此主机。银行

2）通讯过程中的数据泄密和篡改

1. **一般意义上的 https，就是服务器有一个证书。**
   a) 主要目的是保证服务器就是他声称的服务器，这个跟第一点一样。
   b) 服务端和客户端之间的所有通讯，都是加密的。
   i. 具体讲，是客户端产生一个对称的密钥，通过服务器的证书来交换密钥，即一般意义上的握手过程。
   ii. 接下来所有的信息往来就都是加密的。第三方即使截获，也没有任何意义，因为他没有密钥，当然篡改也就没有什么意义了。
2. **少许对客户端有要求的情况下，会要求客户端也必须有一个证书。**
   a) 这里客户端证书，其实就类似表示个人信息的时候，除了用户名/密码，还有一个 CA 认证过的身份。因为个人证书一般来说是别人无法模拟的，所有这样能够更深的确认自己的身份。
   b) 目前少数个人银行的专业版是这种做法，具体证书可能是拿 U 盘（即 U 盾）作为一个备份的载体。

3. 网络机器人

3.1. 定义

网络爬虫（又被称为网页蜘蛛，网络机器人，在 FOAF 社区中间，更经常的称为网页追逐者），是一种按照一定的规则，自动的抓取万维网信息的程序或者脚本。另外一些不常使用的名字还有蚂蚁，自动索引，模拟程序或者蠕虫。

3.2. 必要性

传统的搜索引擎的局限性

1. 不同领域、不同背景的用户往往具有不同的检索目的和需求，通用搜索引擎所返回的结果包含

大量用户不关心的网页

2. 通用搜索引擎的目标是尽可能大的网络覆盖率，有限的搜索引擎服务器资源与无限的网络数据资源之间的矛盾将进一步加深
3. 万维网数据形式的丰富和网络技术的不断发展，图片、数据库、音频/视频多媒体等不同数据大量出现，通用搜索引擎往往对这些信息含量密集且具有一定结构的数据无能为力，不能很好地发现和获取
4. 通用搜索引擎大多提供基于关键字的检索，难以支持根据语义信息提出的查询

3.3. 爬虫

1. **传统爬虫**

   传统爬虫从一个或若干初始网页的 URL 开始，获得初始网页上的 URL，在抓取网页的过程中，不断从当前页面上抽取新的 URL 放入队列,直到满足系统的一定停止条件

2. **聚焦爬虫**

聚焦爬虫是一个自动下载网页的程序，它根据既定的抓取目标，有选择的访问万维网上的网页与相关的链接，获取所需要的信息。与通用爬虫(general?purpose web crawler)不同，聚焦爬虫并不追求大的覆盖，而将目标定为抓取与某一特定主题内容相关的网页，为面向主题的用户查询准备数据资源。

工作流程：聚焦爬虫的工作流程较为复杂，需要根据一定的网页分析算法过滤与主题无关的链接，保留有用的链接并将其放入等待抓取的 URL 队列。然后，它将根据一定的搜索策略从队列中选择下一步要抓取的网页 URL，并重复上述过程，直到达到系统的某一条件时停止。另外，所有被爬虫抓取的网页将会被系统存贮，进行一定的分析、过滤，并建立索引，以便之后的查询和检索；对于聚焦爬虫来说，这一过程所得到的分析结果还可能对以后的抓取过程给出反馈和指导。

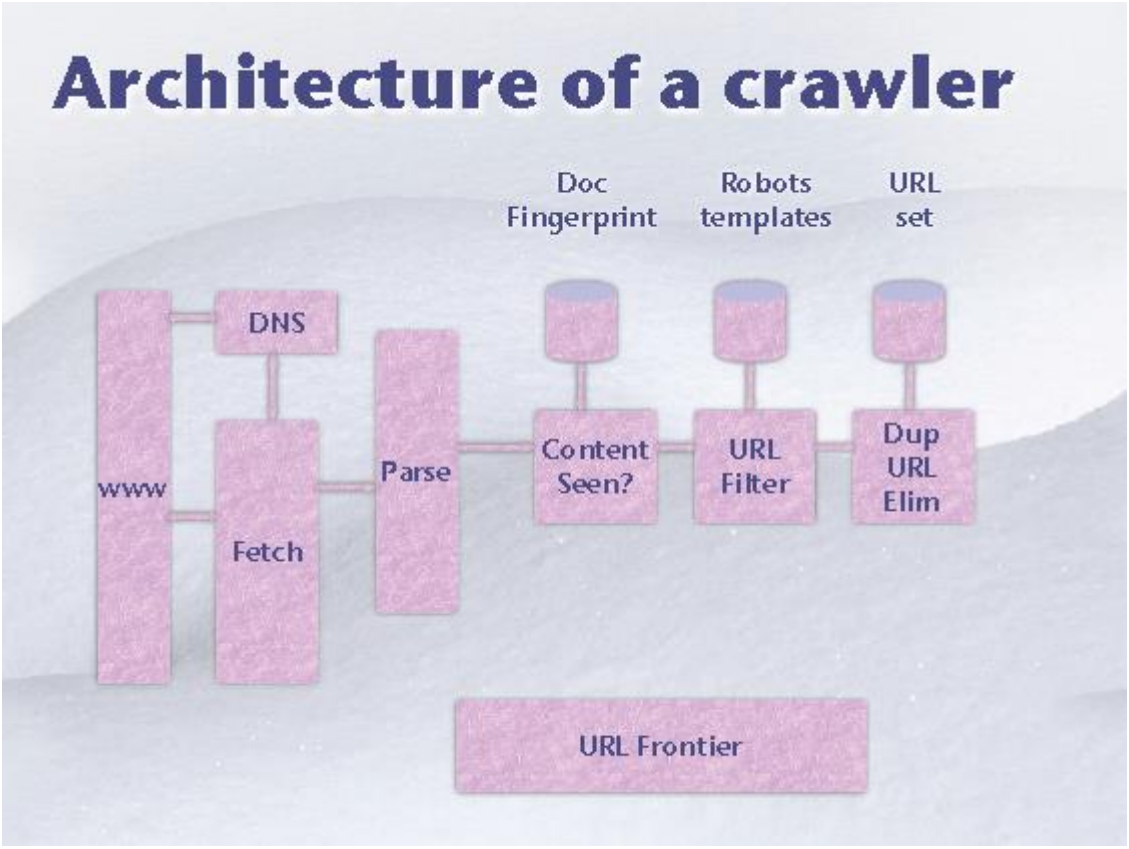
解决问题：

(1) 对抓取目标的描述或定义：决定网页分析算法与 URL 搜索策略如何制订的基础。

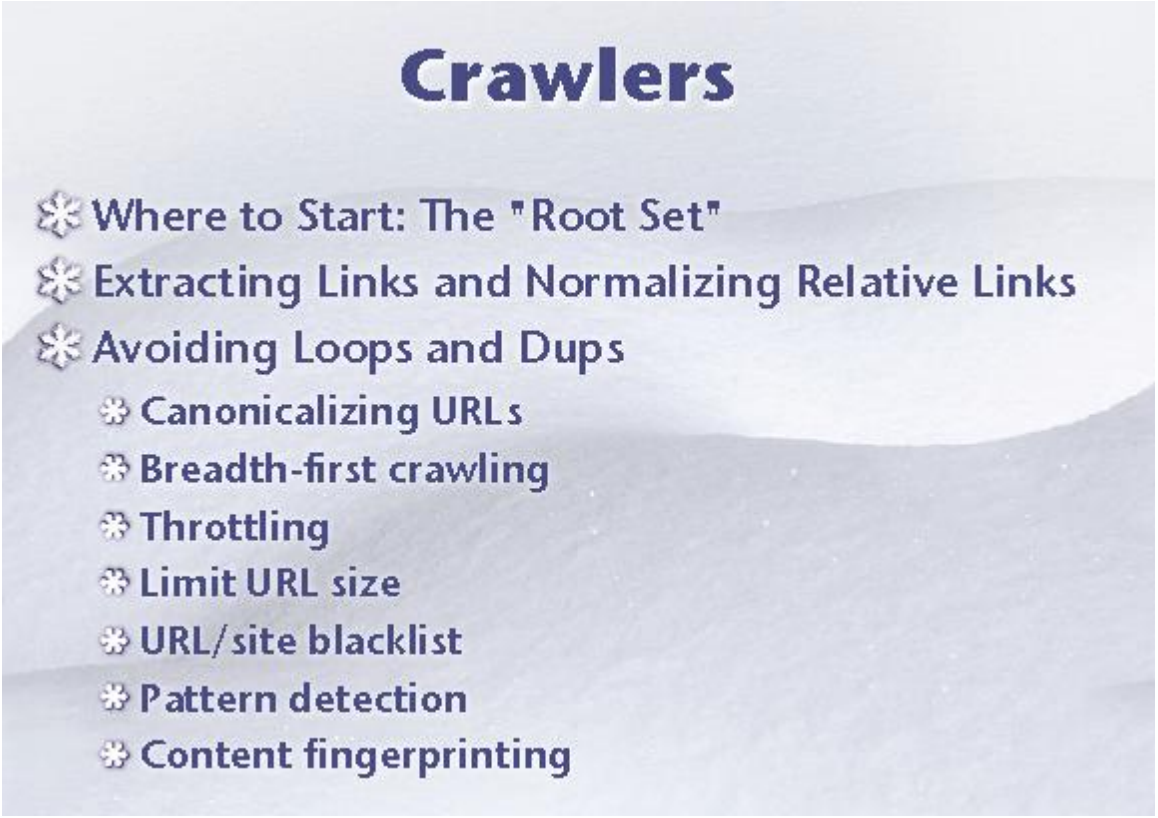
(2) 对网页或数据的分析与过滤；

(3) 对 URL 的搜索策略：网页分析算法和候选 URL 排序算法是决定搜索引擎所提供的服务形式和爬虫网页抓取行为的关键所在。

3. **体系结构**



URL Frontier：存放爬虫要抓取的 URL 的集合
DNS：域名解析，将域名转换为响应的 IP 地址，
Fetch:使用 HTTP 协议抓取 URL
Parse：解析页面，文本和连接被分离
Content seen：检测相同的网页内容是否出现在另一 URL 中
URL filter：判断解析出的 URL 是否应该放入 URL frontier 中；标准化 URL
Dup URL Dlim：去除重复的 URL



网站会给出 URL/robots.txt，说明哪些 URL 是可以被抓取的，连接限制

4. HTML/XHTML

结构表现行为
基本语法，常用标记
重构（为何重构，使用 web 标准）
Html5

4.1. HTML

1. 超文本标记语言，即 HTML（Hypertext Markup Language），是用于描述网页文档的一种标记语言。

HTML 是一种规范，一种标准，它通过标记符号来标记要显示的网页中的各个部分。
HTML 之所以称为超文本标记语言，是因为文本中包含了所谓“超级链接”点——URL 指针,可以点击浏览其他网页。

2. 版本

超文本标记语言（第一版）——在 1993 年 6 月作为互联网工程工作小组（IETF）工作草案发布（并非标准）：

HTML 2.0——1995 年 11 月作为 RFC 1866 发布，在 RFC 2854 于 2000 年 6 月发布

之后被宣布已经过时

HTML 3.2——1996 年 1 月 14 日，W3C 推荐标准

HTML 4.0——1997 年 12 月 18 日，W3C 推荐标准

HTML 4.01（微小改进）——1999 年 12 月 24 日，W3C 推荐标准

ISO/IEC 15445:2000（“ISO HTML”）——2000 年 5 月 15 日发布，基于严格的 HTML 4.01 语法，是国际标准化组织和国际电工委员会的标准。

3. 特点

1 简易性，HTML 版本升级采用超集方式，从而更加灵活方便。

2 可扩展性，HTML 语言的广泛应用带来了加强功能，增加标识符等要求，HTML 采取子类元素的方式，为系统扩展带来保证。

3 平台无关性。虽然 PC 机大行其道，但使用 MAC 等其他机器的大有人在，HTML 可以使用在广泛的平台上，这也是 WWW 盛行的另一个原因。

4. 结构

超文本标记语言标准的 HTML 文件都具有一个基本的整体结构，即 HTML 文件的开头与结尾标志和 HTML 的头部与实体 2 大部分。有 3 个双标记符用于页面整体结构的确认。

```
<html>
    <head></head>
<body></body>
</html>
```

4.2. XHTML

1. 概述

- XHTML 是 The Extensible HyperText Markup Language(可扩展超文本标识语言)的缩写。HTML 是一种基本的 WEB 网页设计语言，XHTML 是一个基于 XML 的标记语言，看起来与 HTML 有些相象，只有一些小的但重要的区别，XHTML 就是一个扮演着类似 HTML 的角色的 XML，所以，本质上说，**XHTML 是一个过渡技术，结合了部分 XML 的强大功能及大多数 HTML 的简单特性。**
- 2000 年底，国际 W3C 组织(World Wide Web Consortium)组织公布发行了 XHTML 1.0 版本。XHTML 1.0 是一种在 HTML 4.0 基础上优化和改进的的新语言，目的是基于 XML 应用。XHTML 是一种增强了的 HTML,XHTML 是**更严谨更纯净**的 HTML 版本。它的可扩展性和灵活性将适应未来网络应用更多的需求。XML 虽然数据转换能力强大，完全可以替代 HTML，但面对成千上万已有的基于 HTML 语言设计的网站，直接采用 XML 还为时过早。因此，在 HTML4.0 的基础上，用 XML 的规则对其进行扩展，得到了 XHTML。所以，建立 XHTML 的目的就是实现 HTML 向 XML 的过渡。目前国际上在网站设计中推崇的 WEB 标准就是基于 XHTML 的应用（即通常所说的 CSS+DIV）。
- 跟 CSS（Cascading Style Sheets，层叠式样式表）结合后，XHTML 能发挥真正的威力；这使实现样式跟内容的分离的同时，又能有机地组合网页代码，在另外的单独文件中，还可以混合各种 XML 应用，比如 MathML、SVG。

2. **XHTML 与 HTML**

- **Html**

  语法比较松散，写起来比较容易，但是兼容性不好
- **XHTML**

  1. 所有的标记都必须要有一个相应的结束标记

2. 所有标签的元素和属性的名字都必须使用小写
3. .所有的 XML 标记都必须合理嵌套
4. .所有的属性必须用引号""括起来
5. .把所有<和&特殊符号用编码表示

任何小于号（<），不是标签的一部分，都必须被编码为<

任何大于号（>），不是标签的一部分，都必须被编码为>

任何与号（&），不是实体的一部分的，都必须被编码为&

注：以上字符之间无空格。

6. 给所有属性赋一个值

<input type="checkbox" name="shirt" value="medium" checked="checked">

7. 不要在注释内容中使“--”

“--”只能发生在 XHTML 注释的开头和结束，也就是说，在内容中它们不再有效。

<!--这里是注释==========这里是注释-->

8. 图片必须有说明文字——alt 说明

3. XHMTL 现行规范

1. XHTML 1.0 Transitional - 过渡型，标识语法要求较宽松

<!DOCTYPEhtml PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

要求非常宽松的 DTD，它允许你继续使用 HTML4.01 的标识(但是要符合 xhtml 的写法)。

2. XHTML 1.0 Strict - 严格型，标识要求达到以上 XHTML 相比于 HTML 的所有改动

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

要求严格的 DTD，你不能使用任何表现层的标识和属性，例如

。

3. XHTML 1.0 Frameset - 框架集定义

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

专门针对框架页面设计使用的 DTD，如果你的页面中包含有框架，需要采用这种 DTD

4. XHTML 1.1 - 模块化的 XHTML

5. XHTML 2.0 - 完全模块化可定制化的 XHTML，正在开发中，参考 http://w3.org/

4.3. 基本标记

- <title>
- <meta>包含文档的相关信息，属性 schema，name，content，http-equiv
  - ✓ SEO（搜索引擎优化）name 设置为 description 或者 keywords

name 设置为 robots，告诉搜索引擎不能索引该页面或者跟踪页面中得任何连接以索引页面

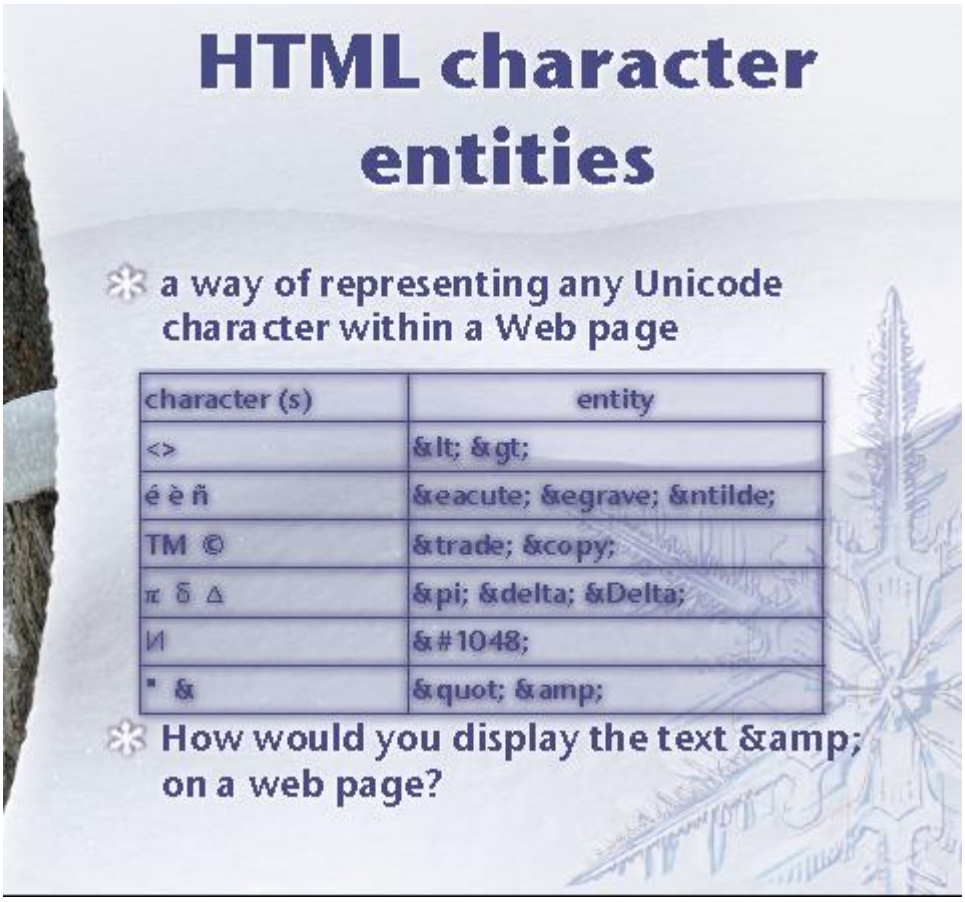
  - ✓ http-equiv content 可以设置 HTTP 头的信息，设置 cookie，指定作者姓名，设置字符编码，设置默认样式表，默认脚本语言，设置 schema 属性。P456
- 块级元素：* address - 地址

  * blockquote - 块引用

* center - 举中对齐块
* dir - 目录列表
* div - 常用块级容易，也是 css layout 的主要标签
* dl - 定义列表
* fieldset - form 控制组
* form - 交互表单
* h1 - 大标题
* h2 - 副标题
* h3 - 3 级标题
* h4 - 4 级标题
* h5 - 5 级标题
* h6 - 6 级标题
* hr - 水平分隔线
* isindex - input prompt
* menu - 菜单列表
* noframes - frames 可选内容，（对于不支持 frame 的浏览器显示此区块内容
* noscript - 可选脚本内容（对于不支持 script 的浏览器显示此内容）
* ol - 排序表单
* p - 段落
* pre - 格式化文本
* table - 表格
* ul - 非排序列表

- 内联元素有: * a - 锚点
  * abbr - 缩写
  * acronym - 首字
  * b - 粗体(不推荐)
  * bdo - bidi override
  * big - 大字体
  * br - 换行
  * cite - 引用
  * code - 计算机代码(在引用源码的时候需要)
  * dfn - 定义字段
  * em - 强调
  * font - 字体设定(不推荐)
  * i - 斜体
  * img - 图片
  * input - 输入框
  * kbd - 定义键盘文本
  * label - 表格标签
  * q - 短引用
  * s - 中划线(不推荐)
  * samp - 定义范例计算机代码
  * select - 项目选择
  * small - 小字体文本

* span - 常用内联容器，定义文本内区块
* strike - 中划线
* strong - 粗体强调
* sub - 下标
* sup - 上标
* textarea - 多行文本输入框
* tt - 电传文本
* u - 下划线
* var - 定义变量

- 

option tags can be set to be initially selected

```
<select name="favoritecharacter[]" size="3" multiple="multiple">
  <option>Jerry</option>
  <option>George</option>
  <option>Kramer</option>
  <option>Elaine</option>
  <option selected="selected">Newman</option>
</select>
```

HTML

Kramer
Elaine
Newman
提交查询

output

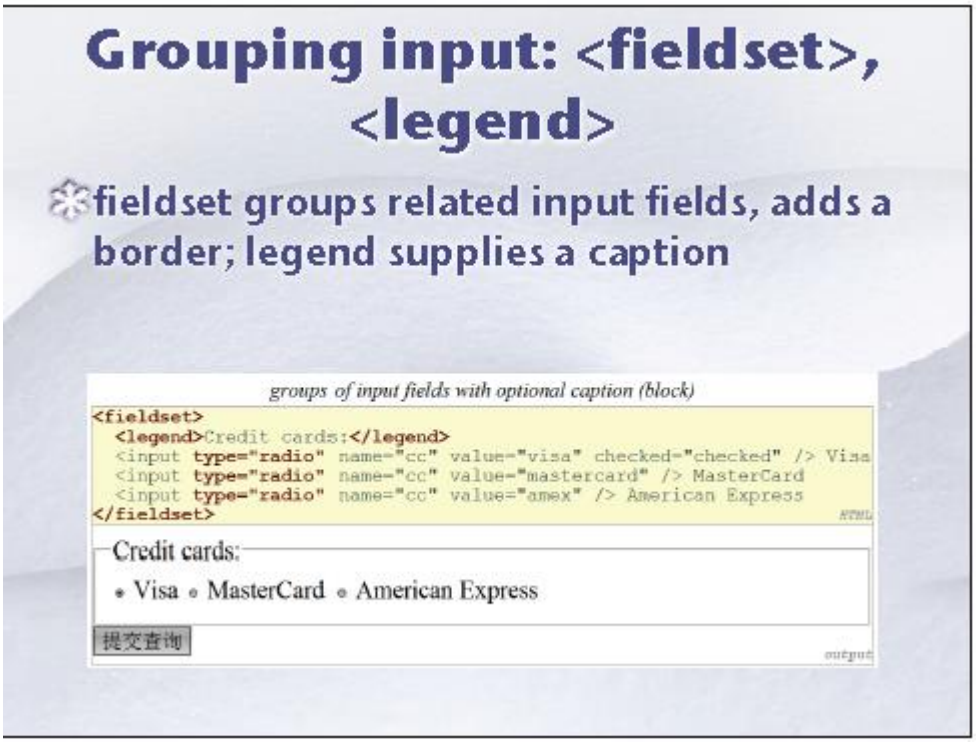
Option groups: <optgroup>

```
<select name="favoritecharacter">
  <optgroup label='Major Characters'>
    <option>Jerry</option>
    <option>George</option>
    <option>Kramer</option>
    <option>Elaine</option>
  </optgroup>
  <optgroup label='Minor Characters'>
    <option>Newman</option>
    <option>Susan</option>
  </optgroup>
</select>
```

HTML

Jerry
提交查询

output

- 

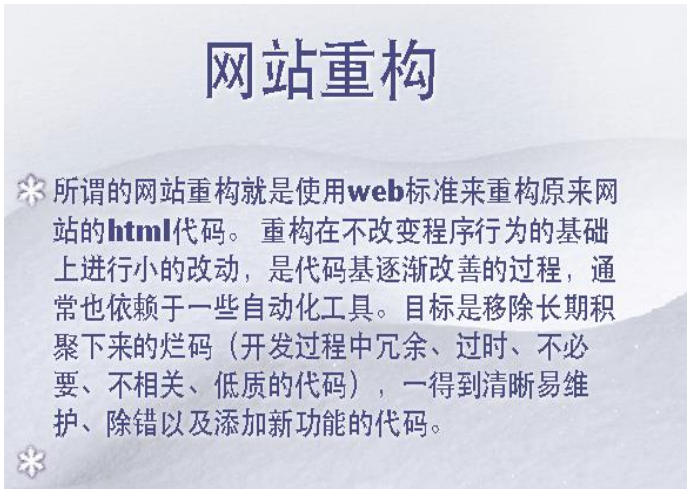
5. 重构

5.1. 重构原因

- 难以辨认的代码
- 缓慢的页面加载速度
- 不同浏览器不同呈现
- Google 搜索中排在 17 页之后
- 网页需要危险或者非标准技术的支持
- 公司主页突然显示 Pwned by elite doodz

5.2. 概念

1. 定义


网站重构
所谓的网站重构就是使用web标准来重构原来网站的html代码。 重构在不改变程序行为的基础上进行小的改动，是代码基逐渐改善的过程，通常也依赖于一些自动化工具。目标是移除长期积聚下来的烂码（开发过程中冗余、过时、不必要、不相关、低质的代码），一得到清晰易维护、除错以及添加新功能的代码。

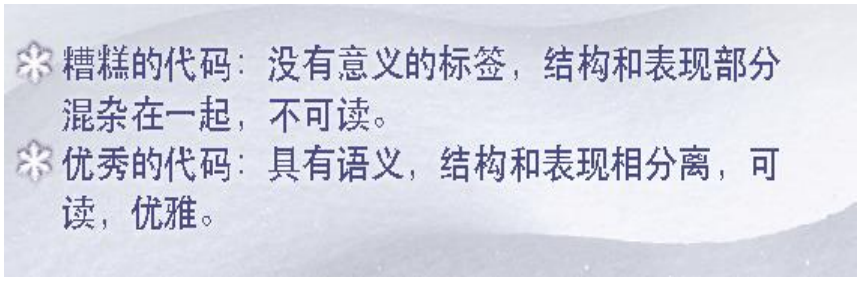
2. **Web 标准**

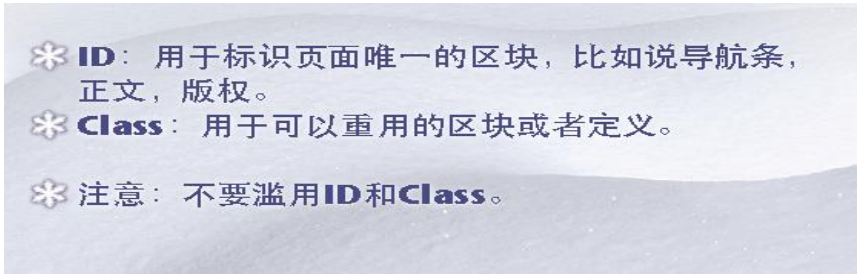

Web标准的定义
Web标准就是结构化的语言(XHTML和XML)，解释性语言(CSS)，对象模型(DOM)和脚本语言(ECMAScript)。更形象的理解就是结构，表现和行为相分离。


为什么要使用Web标准
易于协同开发和维护
代码量减少，降低了带宽成本
提高网站易用性和用户体验
与未来浏览器和手持设备兼容
搜索引擎优化(SEO)
便于改版
加快网页解析的速度

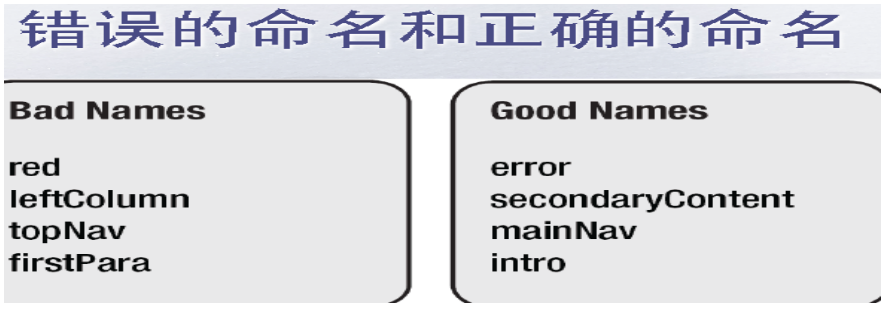
5.3. 重构方法

5.3.1. 糟糕的代码和优秀的代码





5.3.2. 错误的命名和正确的命名

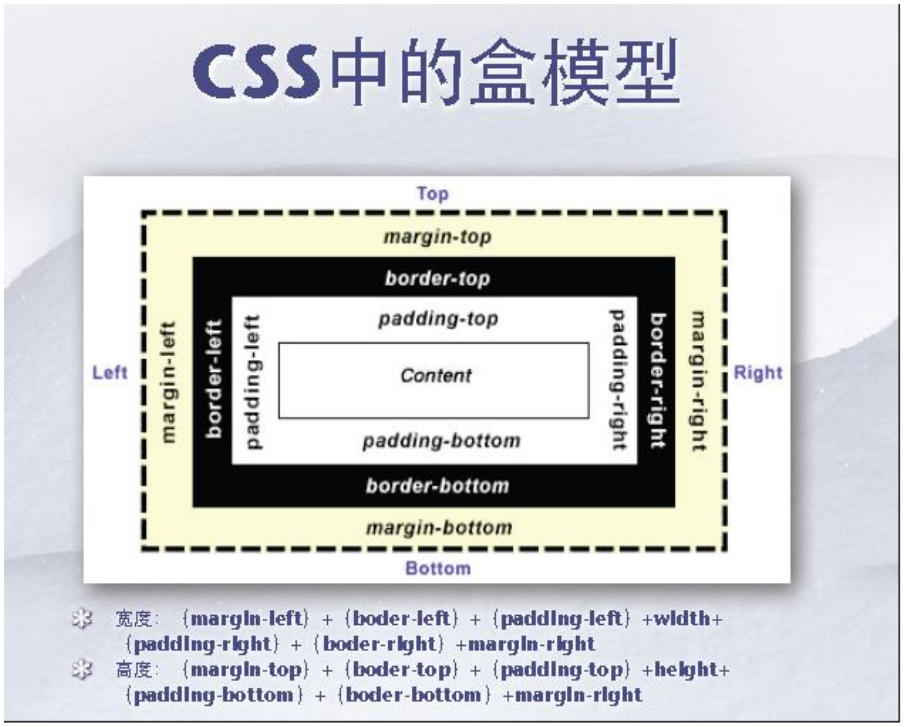


5.3.3. 块级标签和内联标签

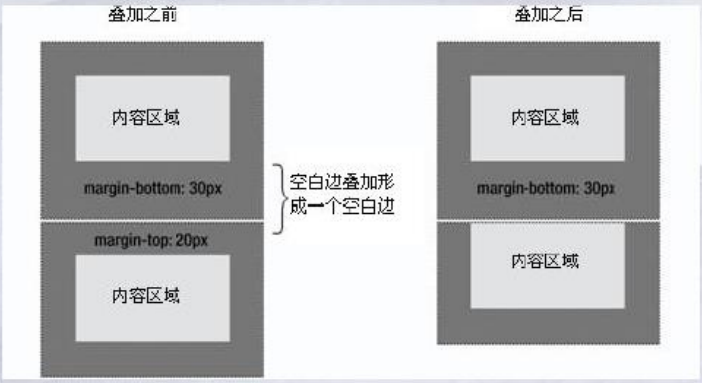
5.3.4. 浏览器模式和文档类型声明

5.3.5. CSS 布局问题


CSS中的布局
CSS中的盒模型
IE和FF的盒模型解析差异
三种定位机制：普通流、绝对定位和浮动


CSS中的盒模型
Top
margin-top
border-top
padding-top
Left
margin-left
border-left
padding-left
Content
padding-right
border-right
margin-right
Right
padding-bottom
border-bottom
margin-bottom
Bottom
宽度：(margin-left) + (boder-left) + (padding-left) +width+ (padding-right) + (boder-right) +margin-right
高度：(margin-top) + (boder-top) + (padding-top) +height+ (padding-bottom) + (boder-bottom) +margin-right

元素的顶空白边与前面元素的底空白边发生叠加

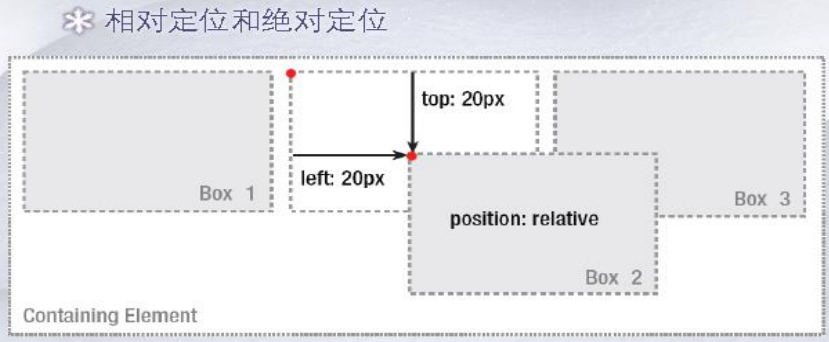


元素的顶空白边与底空白边发生叠加

- 普通流

默认的定位。各标签在页面上的位置是由标签在**html**代码中的位置和自身的属性（块级标签或者内联标签）决定的。

相对定位和绝对定位

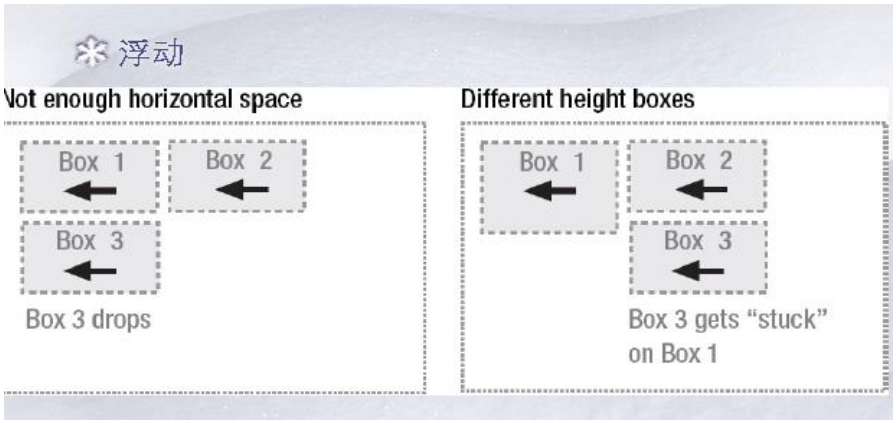


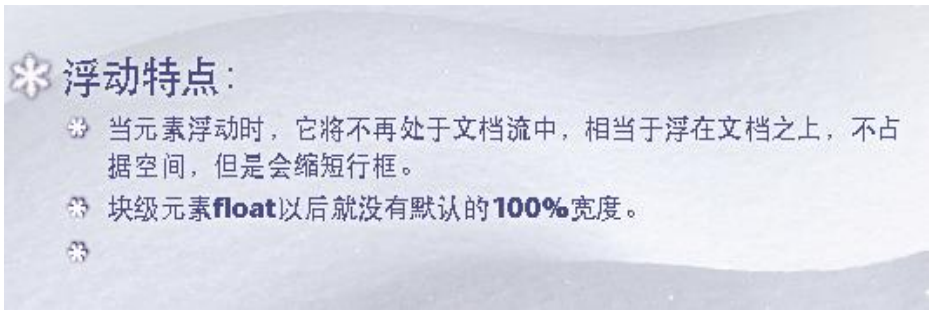
相对定位和绝对定位

相对定位和绝对定位特点：

- 相对定位无论是否移动，元素仍然占据原来的空间；
- 绝对定位不占据空间，绝对定位的标签位置相对于最近已定位的父标签，如果不存在，则相对于**body**；
- **IE**中如果相对于右**(right)**和底部**(bottom)**设置绝对定位，需要确定相对定位的框设置了尺寸。

浮动

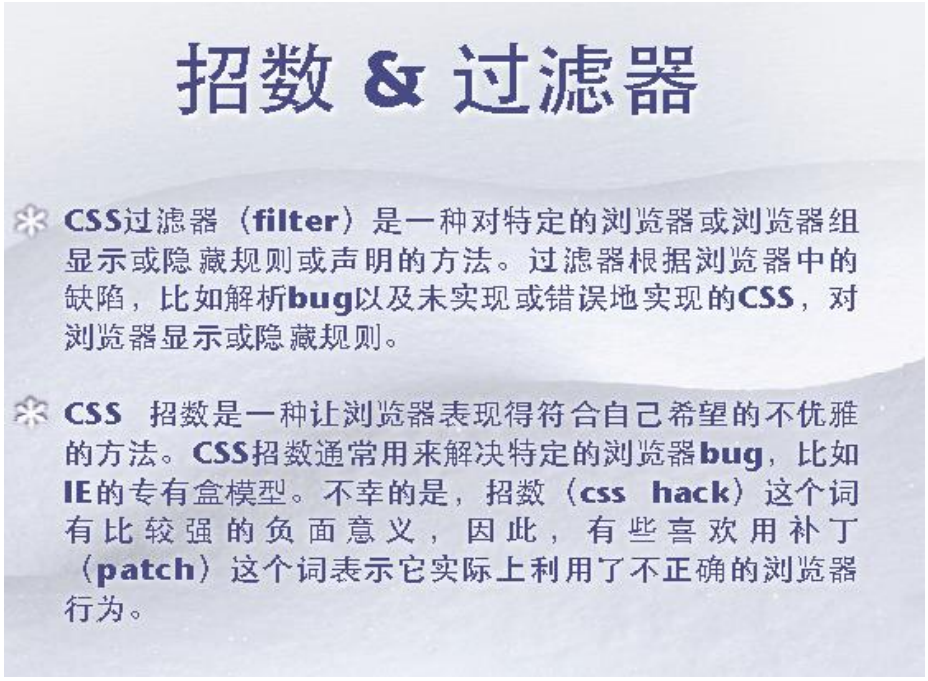


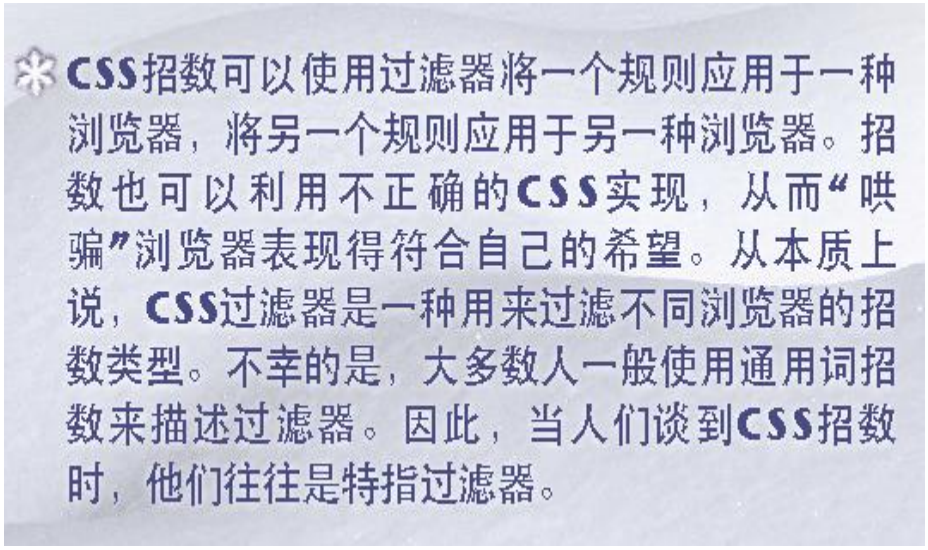


Float 定位一定要设置 width

5.3.6. css 调试技巧

5.3.7. 招数和过滤器


招数 & 过滤器
CSS过滤器（filter）是一种对特定的浏览器或浏览器组显示或隐藏规则或声明的方法。过滤器根据浏览器中的缺陷，比如解析bug以及未实现或错误地实现的CSS，对浏览器显示或隐藏规则。
CSS 招数是一种让浏览器表现得符合自己希望的不优雅的方法。CSS招数通常用来解决特定的浏览器bug，比如IE的专有盒模型。不幸的是，招数（css hack）这个词有比较强的负面意义，因此，有些喜欢用补丁（patch）这个词表示它实际上利用了不正确的浏览器行为。


CSS招数可以使用过滤器将一个规则应用于一种浏览器，将另一个规则应用于另一种浏览器。招数也可以利用不正确的CSS实现，从而“哄骗”浏览器表现得符合自己的希望。从本质上说，CSS过滤器是一种用来过滤不同浏览器的招数类型。不幸的是，大多数人一般使用通用词招数来描述过滤器。因此，当人们谈到CSS招数时，他们往往是特指过滤器。

过滤器 - IE的有条件注释

- IE的有条件注释是一种专有的（因此是非标准的）、对常规XHTML注释的Microsoft扩展。顾名思义，有条件注释使你能够根据条件（比如浏览器版本）显示代码块。尽管是非标准的，但是有条件注释对于其他所有浏览器作为常规注释出现，因此本质上是无害的。有条件注释在Windows上的IE5中首次出现，并且得到了Windows浏览器所有后续版本的支持。
- IE的有条件注释及其有效，而且非常容易记住。主要的缺点是这些注释需要放在HTML页面中，而不是放在CSS中。这样，当你不需要这些东西，或者有所更改的时候，就需要维护很多的地方。

只有IE才能识别

- 只有IE5版本以上

```
<!--[if IE]>
<style type="text/css">
div.content{ color:red;}
</style>
<![endif]-->
```

只有特定版本才能识别

- 只有IE7版本才能识别

```
<!--[if IE 7]>
<style type="text/css">
div.content{ color:red;}
</style>
<![endif]-->
```

只有不是特定版本才能识别

- 除了IE7版本其他版本IE5或以上版本才能识别

```
<!--[if !IE 7]>
<style type="text/css">
div.content{ color:red;}
</style>
<![endif]-->
```

只有低于特定版本的才能识别

- 只有低于IE7版本的IE才能识别

```
<!--[if lt IE 7]>
<style type="text/css">
div.content{ color:red;}
</style>
<![endif]-->
```

等于或者低于特定版本的才能识别

- 只有低于或等于IE7版本的IE才能识别

```
<!--[if lte IE 7]>
<style type="text/css">
div.content{ color:red;}
</style>
<![endif]-->
```


网页重构－微博的改版
HTTP请求数的权衡
对CSS的处理优化
对DOM结构的优化处理
对图片的优化处理
微博3.0截图
新版微博截图

5.3.8. 优化方法


对CSS的处理优化
提取公用模块或公用元素，并反复调用
尽可能少的使用css images
尽量使用CSS3等新技术
鼠标滑上效果改用伪类实现

对DOM结构的优化处理

- bigpipe模式重构并优化垃圾代码
- 尽量减少代码体积
- 首页中杜绝Table布局和iframe

对图片的优化处理

- 图片的存储格式
- css sprites
- 大图片、小图片
- 在已知宽高的图片标签内，直接指定宽高

良好的XHTML规则

- 以正确的DOCTYPE和名字空间开始
- 使用META内容元素声明你的内容编码语言
- 用小写字母写所有元素和属性名称
- 给所有属性值加引号
- 给所有属性赋一个值
- 关闭所有标签
- 用空格和斜杠关闭空标签
- 不要再注释内容中使用“--”
- 把所有符号编码化

6. HTML5

6.1. 概念

1. HTML5 是 HTML 标准的下一个版本

6.2. 与 html4 区别

1. **HTML5 标准还在制定中**

   HTML5 实际上还之是一个还没有真正完成的标准，HTML5 距离真正的标准化还有相当长的一段时间。万维网联盟 W3C 对外表示 HTML5 标准计划在 2014 年完成。而 HTML4 已经有 10 多年了，但它仍是当前正式的标准的事实没有改变。

   另一方面，HTML5 仍处在早期阶段，以后的修改会不断的出现。你必须考虑到这些，因为你在网站上使用的这些新增加或修改的网页元素会每年都出现一些变化，你需要不停的更新升级你的网站，这可不是你希望的。这就是目前为止，你最好在产品里使用 HTML4，只在实验里使用 HTML5 的原因。
2. **简化的语法**

   HTML5 添加了许多新的语法特征，其中包括<video>,<audio>, 和<canvas>元素，同时整合了 SVG 内容。这些元素是为了更容易的在网页中添加和处理多媒体和图片内容而添加的。其它新的元素包括<section>, <article>, <header>, 和<nav>,是为了丰富文档的数据内容。新的属性的添加也是

为了同样的目的。同时也有一些属性和元素被移除掉了。一些元素，像<a>, <cite>和<menu>被修改，重新定义或标准化了。

3. **<canvas> 标记取代 Flash**

Canvas 元素是 HTML5 的一部分，允许脚本动态渲染位图像。Canvas 由一个可绘制地区 HTML 代码中的属性定义决定高度和宽度。JavaScript 代码可以访问该地区，通过一套完整的绘图功能类似于其他通用二维 API，从而使动态生成的图形。其一些可能的用途，包括使用 Canvas 构造图形，动画，游戏和图片

4. **新的<header>和<footer>标记**

使用新标记替换掉 标记是 HTML5 在语义方面的主要成就。这 在 HTM4L 里被广泛使用，但是它所表达的语义太弱，在声明网页组织结构里不同的区块的任务面前它毫无用武之地。

新的 HTML5 标记 - 例如 <article>，<aside>，<nav>，<figure>，<header>，<footer> - 会更有用和更方便，这些标记能够让你指明网页不同区域的用途，清楚的显示网站的结构。SEO 搜索引擎更易优化

5. **新的 <section> 和<article> 标记**

HTML5 中引入的新的<section> 和 <article> 标记可以让开发人员更好的标注页面上的这些区域。

在同一个页面中可以使用多个<footer>元素，即可以用作页面整体的页脚，也可以作为一个内容区块的结尾，例如，我们可以将<footer>直接写在<section>或是<article>中

6. **新的 <menu> 和<figure> 标记**

新的<menu>标记可以被用作普通的菜单，也可以用在工具条和右键菜单上，虽然这些东西在页面上并不常用。

类似的，新的 <figure> 标记是一种更专业的管理页面上文字和图像的方式。当然，你可以用样式表来控制文字和图像，但使用 HTML5 内置的这个标记更适合。

7. **新的 <audio> 和 <video> 标记**

新的<audio> 和 <video> 标记可能是 HTML5 中增加的最有用处的两个东西了。正如标记名称，它们是用来嵌入音频和视频文件的。

如下是一个<audio>标记示例：

```
<audio  autoplay="autoplay"  controls="controls">

<source  src="music.ogg"

<source  src="music.mp3"  />

</audio>
```

8. **全新的表单设计**

```
源码

<form id="login">
```

```
<h1>Log In</h1>

<fieldset id="inputs">

<input id="username" type="text" placeholder="Username" autofocus required>

<input id="password" type="password" placeholder="Password" required>

</fieldset>

<fieldset id="actions">

<input type="submit" id="submit" value="Login">

<a href="">Forgot your password?</a><a href="">Register</a>

</fieldset>

</form>
```

9. **不再使用 <b> 和<font> 标记，**将内容和展示分离
10. **不再使用 <frame>, , <big> 标记**
11. 本地数据库

7. CSS

7.1.

7.2. 浏览器标准模式怪异模式

- IE8 有 4 种模式：IE5.5 怪异模式、IE7 标准模式、IE8 几乎标准模式、IE8 标准模式

怪异模式：各个浏览器在对页面的渲染上存在差异，甚至同一浏览器在不同版本中，对页面的渲染也不同。在 W3C 标准出台以前，浏览器在对页面的渲染上没有统一规范，产生了差异(Quirks mode 或者称为 Compatibility Mode)；
标准模式：由于 W3C 标准的推出，浏览器渲染页面有了统一的标准(CSScompat 或称为 Strict mode 也有叫做 Standars mode)，这就是二者最简单的区别。

W3C 标准推出以后，浏览器都开始采纳新标准，但存在一个问题就是如何保证旧的网页还能继续浏览，在标准出来以前，很多页面都是根据旧的渲染方法编写的，如果用标准来渲染，将导致页面显示异常。**为保持浏览器渲染的兼容性，使以前的页面能够正常浏览，浏览器都保留了旧的渲染方法（如：微软的 IE）。这样浏览器渲染上就产生了 Quircks mode 和 Standars mode，两种渲染方法**

共存在一个浏览器上。

触发标准模式

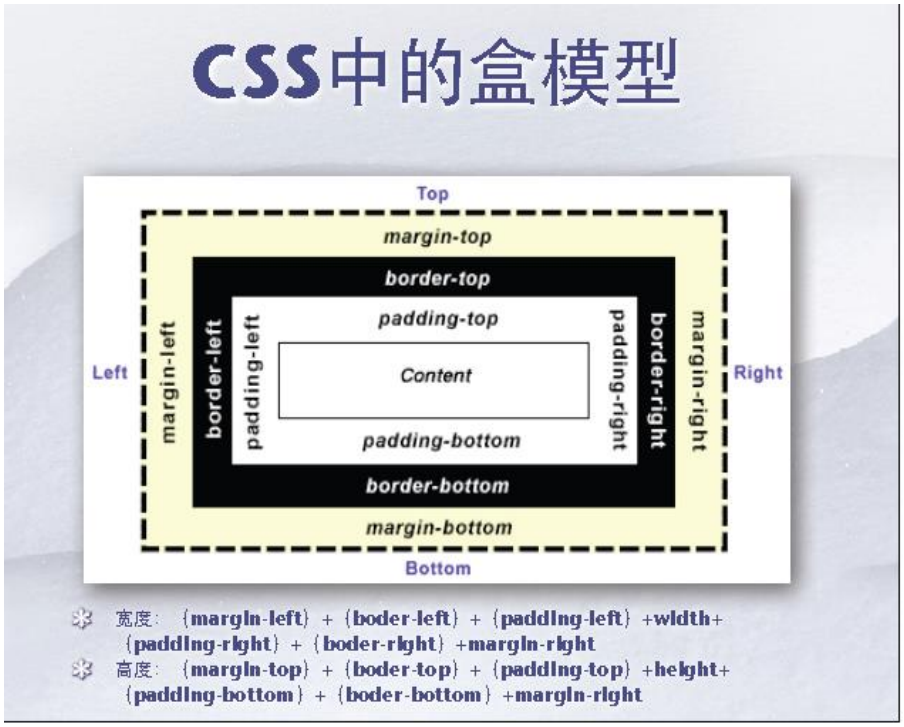
1、加 DOCTYPE 声明,比如：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<!DOCTYPE html>
2、设置 X-UA-Compatible 触发。

触发怪异模式

1、 无 doctype 声明、定义旧的 HTML 版本（HTML4 以下,例如 3.2）
2、加 XML 声明，可在 ie6 下触发
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ...>
2、
3、在 XML 声明和 XHTML 的 DOCTYPE 之间加入 HTML 注释，可在 ie7 下触发 <?xml version="1.0" encoding="utf-8"?>
<!-- keep IE7 in quirks mode -->
<!DOCTYPE ...>
5、<!--->放在<!DOCTYPE 前面

7.3. CSS 盒模型



内容(content)、填充(padding)、边框(border)、边界(margin)， CSS 盒子模式都具备这些属性。

7.4. 优先级顺序和继承关系

7.4.1. 优先级

所谓 CSS 优先级，即是指 CSS 样式在浏览器中被解析的先后顺序。

[1 位重要标志位] > [4 位特殊性标志] > 声明先后顺序

既然样式有优先级，那么就会有一个规则来约定这个优先级，而这个“规则”就是本次所需要讲的重点。

样式表中的特殊性描述了不同规则的相对权重，它的基本规则是：

1. 统计选择符中的 ID 属性个数。
2. 统计选择符中的 CLASS 属性个数。
3. 统计选择符中的 HTML 标记名个数。

最后，按正确的顺序写出三个数字，不要加空格或逗号，得到一个三位数(css2.1 是用 4 位数表示)。(注意，你需要把数字转换成一个以三个数字结尾的更大的数)。相应于选择符的最终数字列表可以很容易确定较高数字特性凌驾于较低数字的。

例如：

1. 内联样式[1.0.0.0]
2. 每个 ID 选择符(#someid)，加 0,1,0,0。
3. 每个 class 选择符(.someclass)、每个属性选择符(形如[attr=value]等)、每个伪类(形如:hover 等)加 0,0,1,0。
4. html 元素或伪元素(:firstchild)等，加 0,0,0,1。
5. 其它选择符包括全局选择符*，加 0,0,0,0。相当于没加，不过这也是一种 specificity，后面会解释。
6. 继承样式和结合符[无]

三、特性分类的选择符列表

以下是一个按特性分类的选择符的列表：

选择符	特性值
h1 {color:blue;}	1
p em {color:purple;}	2
.apple {color:red;}	10
p.bright {color:yellow;}	11
p.bright em.dark {color:brown;}	22
#id316 {color:yellow}	100

单从上面这个表来看，貌似不大好理解，下面再给出一张表：

选择符	特性值
h1 {color:blue;}	1
p em {color:purple;}	1+1=2
.apple {color:red;}	10
p.bright {color:yellow;}	1+10=11
p.bright em.dark {color:brown;}	1+10+1+10=22
#id316 {color:yellow}	100

以上是原作者归纳的，现在我补充几点：

1、内嵌样式（即元素内部style=""所定义的样式）也是有特殊性的，而且特性值很高，为1000，高于ID选择符。

2、通配符"*"特性值可以看成是为0，可以用于任何一种元素。

3、关于重要性（!important），后面有关于它的说明，在此我想说一下它的使用：

首先，IE6不识别，这一点估计没人不知道了，纯属啰嗦；

!important总是放在声明的最后，即分号前，如果一个属性的值包含多个关键词，比如font属性，必须将!important放在所有声明的最后，如：

```
p.text {color: green !important;font: normal !important 14px/1.5em Tahoma,Verdana,"宋体",Arial;}/* 不能这样写!important */
p.text {color: red;font: normal 10px/1.5em Tahoma,Verdana,"宋体",Arial;}
/* 文本样式最终为字体颜色：绿色；字体大小：10px */
```

4、在CSS中，样式的继承非常常见，但是继承的样式是没有特殊性的，甚至连0特殊性都没有，所以无论出现的顺序如何，继承所得的样式优先级低于上面所说的任何一种。还有，在CSS中，所有边框模型属性（外边距、内边距、背景和边框）都不能继承。

7.4.2. 继承

继承：子元素继承父元素的特性，
继承是 CSS 的一个主要特征，它是依赖于祖先-后代的关系的。继承是一种机制，它允许样式不仅可以应用于某个特定的元素，还可以应用于它的后代。例如一个 BODY 定义了的颜色值也会应用到段落的文本中。

1. 文内的样式优先级为 1,0,0,0，所以始终高于外部定义。这里文内样式指形如 blah 的样式，而外部定义指经由<link>或<style& gt;卷标定义的规则。
2. 有!important 声明的规则高于一切。
3. 如果!important 声明冲突，则比较优先权。
4. 如果优先权一样，则按照在源码中出现的顺序决定，后来者居上。
5. 由继承而得到的样式没有 specificity 的计算，它低于一切其它规则(比如全局选择符*定义的规则)。
6. 关于经由@import 加载的外部样式，由于@import 必须出现在所有其它规则定义之前(如不是，则浏览器应该忽略之)，所以按照后来居上原则，一般优先权冲突时是占下风的。（如果样式表中有导入的样式表，一般认为出现在导入样式表中的声明在前，主样式表中的所有声明在后。

8. Javascript

8.1. 优缺点

- **优点**：使用 JavaScript 就可以在客户端进行数据验证。
  JavaScript 可以方便地操纵各种浏览器的对象
  可以使用 JavaScript 来控制浏览器的外观，状态甚至运行方式
  可以根据用户的需要“定制”浏览器，从而使网页更加友好。
  JavaScript 可以使多种任务仅在用户端就可以完成而不需要网络和服务器的参与，从而支持分布式的运算和处理。
- **缺点**
  1. JavaScript 简单性
  2. 解释执行
  3. 基于对象，一个变量可以赋不同数据类型的值，弱类型,会产生很多难以查到的错误基本语法

8.2. First-class functions

1. **Javascript 特点：**

   Dynamic （动态）

   dynamic typing （动态类型）

object based （基于对象）

run-time evaluation（运行时执行）

Functional （函数式）

first-class functions （functions 是第一个类）

nested functions （嵌套函数）

closures （闭包）

Prototype-based （基于原型）

prototypes

functions as object constructors

functions as methods

Miscellaneous

run- time environment

variadic functions

array and object literals

regular expressions

2. 在 javascript 解释器启动时,会首先创建一个全局的对象(global object),也就是"window"所引用的对象.然后我们定义的所有全局属性和方法等都会成为这个对象的属性.不同的函数和变量的作用域是不同的,因而构成了一个作用域链(scope chain).很显然，在 javascript 解释器启动时,这个作用域链只有一个对象:window(Window Object，即 global object).
3. Firstclass functions 以 functions 来构造对象
4. Function 对象是 JavaScript 里面的固有对象，所有的函数实际上都是一个 Function 对象

```
function Person(name, age) {
    this.getName = function() { return name; };
    this.setName = function(newName) { name = newName; };
    this.getAge = function() { return age; };
    this.setAge = function(newAge) { age = newAge; };
}

var p1 = new Person("sdcyst",3);
alert(p1.getName());  //sdcyst
alert(p1.name);       //undefined     因为 Person('类')没有 name 属性
```

```
p1.name = "mypara"    //显示的给 p1 添加 name 属性
alert(p1.getName());  //sdcyst      但是并不会改变 getName 方法的返回值
alert(p1.name);       //mypara      显示出 p1 对象的 name 属性
p1.setName("sss");    //改变私有的"name"属性
alert(p1.getName());  //sss
alert(p1.name);       //仍旧为 mypara
```

8.3. 事件驱动编程

JavaScript 是基于对象(object-based)的语言。这与 Java 不同,Java 是面向对象的语言。而基于对象的基本特征，就是采用事件驱动(event-driven)。它是在用形界面的环境下，使得一切输入变化简单化。通常鼠标或热键的动作我们称之为事件(Event)，而由鼠标或热键引发的一连串程序的动作，称之为事件驱动(Event Driver)。而对事件进行处理程序或函数，我们称之为事件处理程序(Event Handler)。

在 JavaScript 中对象事件的处理通常由函数(Function)担任。其基本格式与函数全部一样，可以将前面所介绍的所有函数作为事件处理程序。

格式如下：

Function 事件处理名(参数表){

事件处理语句集;

……

}

(1) 单击事件 onClick：button，radio，reset button，submit
(2) onChange 改变事件
(3) 选中事件 onSelect
(4) 获得焦点事件 onFocus
(5) 失去焦点 onBlur
(6) 载入文件 onLoad
(7) 卸载文件 onUnload：当 Web 页面退出时引发 onUnload 事件，并可更新 Cookie 的状态。

8.4. InnerHTML

```
<html>
<head>
<script language="javascript">
```

```
function Test(){
        var str="";
        str+="Hello,";
        str+="This is a Test!<br />";
        str+="I Love you;<br />";
        str+="I Love you,too!";
        p.innerHTML=str+"<br /><br />"+Math.random();
        setTimeout('Test();',1000);
}
</script>
</head>
<body onload=Test()>
<span id="p"></span>
</body>
</html>
```

innerTEXT 与 innerHTML 的区别：

innerHTML 自带了语法检查功能,他会自动把不完整的 HTML 代码补充完整.运行如下的一个测试代码就可以发现了.

```
document.getElementById("AlbumList").innerHTML="<table><tr>";
alert(document.getElementById("AlbumList").innerHTML);
```

他会自动把我的代码里面添加了<tbody>和</tr></table>等标记.神奇!!!

所以在网页中,我们经常这样用;

```
<div id="content"></div>

<script language="javascript">

document.getElementById("content").innerHTML="    需显示的内容  "
```

</script>

这样就会在 id 是 content 的标记那里显示"需显示的内容";

这个可以用在显示一篇文章的摘要的部分,防止将带有 HTML 标记的文章中的代码截掉标记的结束部分,

e,g:

str="沙漠中的绿洲<img src=image.jpg>"

left(str,60)则是"沙漠中的绿洲<img src=image"

<script language="javascript">

document.getElementById("content").innerHTML="left(str,60) "

</script>

innerHTML，是在块中加 html 代码；innerText 在块中加文字。 (注意大小写)

innerText 属性在 IE 浏览器中可以得到当前元素过滤掉 HTML Tags 之后的文本内容，在某些时候还是比较有用。但类似的非标准属性/方法在其他浏览器中并不一定都得到支持。

例子:<html>
<head>
<title>DHtml 举例 13</title>
<style><!--
body {font-family:"宋体";color="blue";font-size:"9pt"}
.blue {color:blue;font-size:9pt}
-->
</style>
<script language="JavaScript">
function OutputText()
{
if(frm.txt.text!="")
{ Output.innerHTML="在此处输出文本：<u>"+frm.txt.value+"</u>";} //Output 为一对象。
else
{ Output.innerText="在此处输出文本：";}
}//function

```
</script>
</head>

<body>
<p><br></p>

<form name="frm">
<p><font color="gray">请在文本框中输入文字:</font>
<input type="text" name="txt" size="50"><br>
<input type="button" value="输出文本" name="B1" class="blue" onclick="OutputText()"></p>
</form>

<p id="Output">在此处输出文本：</p>
</body>
</html>
```

8.5. 面向对象

1. **用 JavaScript 实现类**

JavaScritpt 没有专门的机制实现类，这里是借助它的函数允许嵌套的机制来实现类的。一个函数可以包含变量，又可以包含其它函数，这样，变量可以作为属性，内部的函数就可以作为成员方法了。因此外层函数本身就可以作为一个类了。如下：

```
function myClass()
{
  //此处相当于构造函数
}
```

这里 myClass 就是一个类。其实可以把它看成类的构造函数。至于非构造函数的部分，以后会详细描述。

2. **如何获得一个类的实例**

实现了类就应该可以获得类的实例，JavaScript 提供了一个方法可以获得对象实例。即 new 操作符。 其实 JavaScript 中，类和函数是同一个概念，当用 new 操作一个函数时就返回一个对象。如下：

```
var obj1 = new myClass();
```

3. **对象的成员的引用**

在 JavaScript 中引用一个类的属性或方法的方法有以下三种。

1> 点号操作符

这是一种最普遍的引用方式，就不累赘。即如下形式：

```
对象名.属性名;
对象名.方法名;
```

2> 方括号引用

JavaScript 中允许用方括号引用对象的成员。如下：

```
对象名["属性名"];
对象名["方法名"];
```

这里方括号内是代表属性或方法名的字符串，不一定是字符串常量。也可以使用变量。这样就可以使用变量传递属性或方法名。为编程带来了方便。在某些情况下，代码中不能确定要调用那个属性或方法时，就可以采用这种方式。否则，如果使用点号操作符，还需要使用条件判断来调用属性或方法。

另外，使用方括号引用的属性和方法名还可以以数字开头，或者出现空格，而使用点号引用的属性和方法名则遵循标示符的规则。但一般不提倡使用非标示符的命名方法。

6. **prototype**

每个函数对象都具有一个子对象 prototype，因为函数也可以表示类，所以 prototype 表示一个类的成员的集合。当 new 一个对象时，prototype 对象的成员都会被实例化成对象的成员。先看一个例子：

```
function myClass()
{}

myClass.prototype.ID = 1;
myClass.prototype.Name = "johnson";
myClass.prototype.showMessage = function()
{
    alert("ID: " + this.ID + "Name: " + this.Name);
}

var obj1 = new myClass();
obj1.showMessage();
```

porttype 实现继承

8.6. 匿名函数

```
(function(){

//这里忽略 jQuery 所有实现

})();
```

匿名函数就是没有实际名字的函数.

例如我们在设定一个 DOM 元素事件处理函数的时候，我们通常都不会为他们定名字，而是赋予它的对应事件引用一个匿名函数。

对匿名函数的调用其实还有一种做法，也就是我们看到的 jQuery 片段——使用()将匿名函数括起来，然后后面再加一对小括号（包含参数列表）。

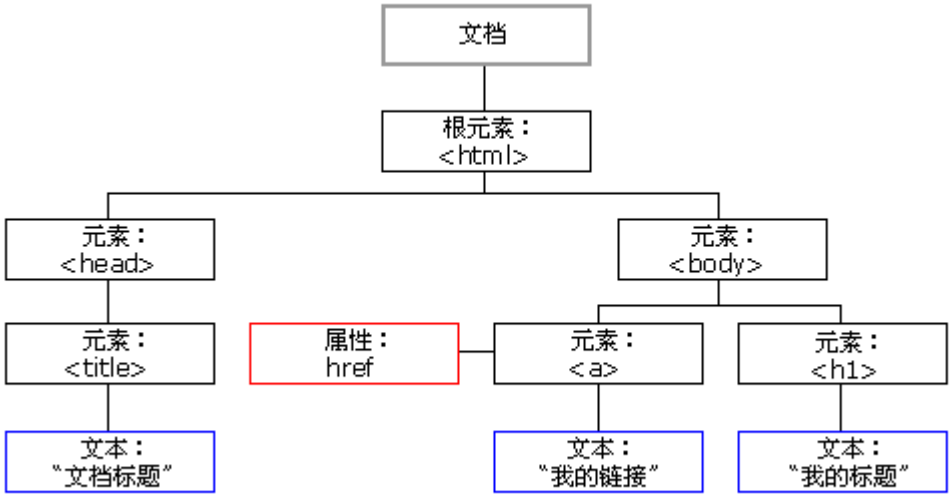
9. DOM

9.1. XHTML/xml 与 DOM 树

DOM 是 W3C（万维网联盟） 的推荐标准。DOM 定义了访问诸如 XML 和 XHTML 文档的标准。

DOM 的全称是 Document Object Model，也即文档对象模型。在应用程序中，基于 DOM 的 XML 分析器将一个 XML 文档转换成一个对象模型的集合（通常称 DOM 树），应用程序正是通过对这个对象模型的操作，来实现对 XML 文档数据的操作。通过 DOM 接口，应用程序可以在任何时候访问 XML 文档中的任何一部分数据，因此，这种利用 DOM 接口的机制也被称作随机访问机制。

DOM 接口提供了一种通过分层对象模型来访问 XML 文档信息的方式，这些分层对象模型依据 XML 的文档结构形成了一棵节点树。无论 XML 文档中所描述的是什么类型的信息，即便是制表数据、项目列表或一个文档，利用 DOM 所生成的模型都是节点树的形式。也就是说，DOM 强制使用树模型来访问 XML 文档中的信息。由于 XML 本质上就是一种分层结构，所以这种描述方法是相当有效的。



9.2. DOMO DOM2 事件流（捕获、目标和冒泡）

1.事件流

1.1 事件冒泡（IE 事件流）

□事件冒泡（event bubbling），即事件开始时由最具体的元素（文档中嵌套层次最深的那个节点）接受，然后逐级向上传播到较为不具体的节点（文档）。

□所有浏览器均支持事件冒泡。Firefox、chrome、safari 将事件一直冒泡到 window 对象。

1.2 事件捕获（Netscape 事件流）

□不太具体的节点更早收到事件，而具体的节点最后收到节点。

□Safari、chrome、Opera、firefox 支持，但从 window 对象开始捕获（DOM2 级规范是从 document 开始）。

1.3DOM 事件流

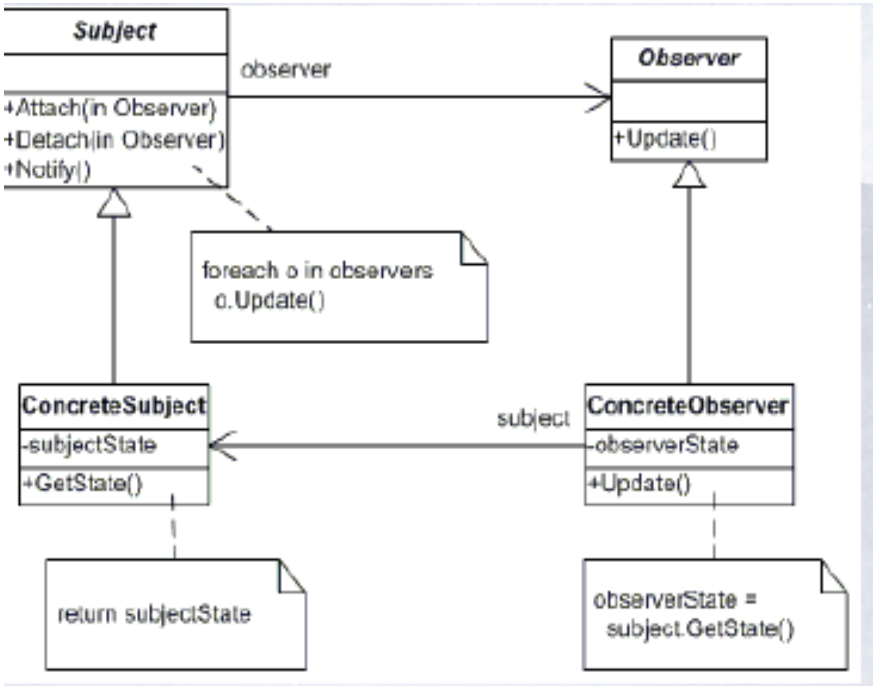
□"DOM2 级事件"规定的事件流包括三个阶段：**事件捕获阶段、处于目标阶段和事件冒泡阶段。**

□DOM2：捕获阶段不含实际目标，不涉及事件目标，只为截获事件提供机会。

□Safari、chrome、Firefox 和 Opera9.5 以上都会在捕获阶段触发事件对象上的事件。结果有两个机会在目标上操作事件。

□IE 不支持 DOM 事件流。其他支持。

9.3. 观察者模式



观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听一个主题对象。这个主题对象在状态上发生变化时，会通知所有观察者，使它们能够自动更新自己。

9.4. 事件处理程序

2.事件处理程序（或事件侦听器）

定义：响应某个事件的函数。

2.1 HTML 事件处理程序

□某个元素支持的每种事件，都可以使用一个相应事件处理程序同名的 HTML 特性来指定。

□其中的函数代码字符需经过 HTML 转义。

□存在问题：

◇时差问题：当触发事件时，事件处理程序有可能尚不具执行条件。使用 try-catch 块封装，以便错误不会浮出水面。

```
<input type="button" value="click me"
onclick="try{show();}catch(ex){}"/>
```

◇HTML 与 JavaScript 代码紧密耦合。

2.2 DOM0 级事件处理程序

□DOM0 级事件处理程序：将一个函数赋值给一个事件处理程序属性。**优点：简单、跨浏览器。**

□这种方式的事件处理程序在代码运行以前不会绑定事件。

□使用 DOM0 级方法指定的事件处理程序被认为是元素的方法。事件处理程序在元素作用域中运行；程序中 this 引用当前元素。

□删除 DOM0 级事：btn.onclick = null； //删除事件处理程序

2.3 DOM2 级事件处理程序

□addEventListener()和 removeEventListener()。**接受 3 个参数：要处理的事件名、作为事件处理程序的函数和一个布尔值。**

◇如果最后这个布尔值是 true，表示捕获阶段调用事件处理程序；如果是 false 表示冒泡阶段调用事件处理程序。

```
Var btn = document.getElementById("myBtn");
Btn.addEventListener("click",function(){alert(this.id);},false);
```

□使用 DOM2 级方法添加事件处理程序的主要好处是可以添加多个事件处理程序。按添加顺序触发。

□通过 addEventListener()添加的事件处理程序只能使用 removeEventListener()来移除；移除时传入的参数与添加处理程序时使用的参数相同。即匿名函数无法移除。

□将事件处理程序添加到冒泡阶段，得到最大限度兼容。

□Firefox、Safari、Chrome 和 Opera 支持 DOM2 级事件处理程序。IE 不支持，有独有的事件处理程序。

10. XML

1.1. 优缺点

XML 的特点			
	1	具有良好的格式	标记一定要拥有结尾标记，如：<name>coolsun</name>
	2	具有验证机制	XML 的标记是程序员自己定义的，标记的定义和使用是否符合语法，需要验证。XML 有两种验证方法：一种是 DTD（Document Type Definition），即文档类型定义，DTD 是一个专门的文件，用来定义和校验 XML 文档中的标记；另一种是 XML Schema，用 XML 语法描述，它比 DTD 更优越，多个 Schema 可以复合使用 XML 名称空间，可以详细定义元素的内容及属性值的数据类型。
	3	灵活的 WEB 应	在 XML 中，数据和显示格式是分离设计的，HTML 提

优点		用	供显示的内容，而 XML 描述数据本身。
	4	丰富的显示样式	XML 数据定义打印、显示排版信息主要有 3 种方法：用 CSS 定义打印和显示排版信息；用 XSLT 转换到 HTML 进行显示和打印；用 XSLT 转换成 XSL 的 FO（Formatter Object）进行显示和打印
	5	XML 是电子数据交换（EDI）的格式	XML 是为互联网的数据交换而设计，它不仅仅是 SGML 定义的用于描述的文档，而且在电子商务等各个领域使用数据交换成为可能
	6	便捷的数据处理	XML 可以很方便地与数据库中的表进行相互转换。XML 使计算机能够很简易地读取和存储资料，并确保数据结构精确。
	7	面向对象的特性	XML 是信息的对象化语言。DTD 和 Schema 是界面和类（Interface 和 Class），XML 是对象实例（Object），XSL 是方法和实现（Method 和 Implement）。XML-Data 解决了 XML 类的继承问题。
	8	开放的标准	XML 基于的标准是为 Web 进行过优化的。由于 XML 彻底把标识的概念同显示分开了，处理者能够在结构化的数据中嵌套程序化的描述以表明如何显示数据。XML 是信息的高层封装与运输的标准。
	9	选择性更新	通过 XML，数据可以在选择的局部小范围内更新。每当一部分数据变化后，不需要重发整个结构化的数据。
	10	XML 是一个技术大家族	XML 是一套完整的方案，有一系列相关技术，包括文件数据验证、显示输出、文件转换、文档对象和连接等。
缺点	1	树状存储	虽然搜索效率极高，但是插入和修改比较困难
	2	大数据量低效率	XML 的文本表现手法、标记的符号化会导致 XML 数据比二进制表现数据量增加，尤其当数据量很大的时候，效率就成为很大的问题。
	3	管理功能不完善	XML 文档做为数据提供着使用，没有数据库系统那样完善的管理功能
	4	通信难	由于 XML 是元置标语言，任何人、公司和组织都可以利用它定义新的标准，这些标准间的通信就成了巨大的问题。

1.2. 应用

- XHTML 是基于 xml 开发的标准
- 数据传输

1.3. DTD 与 XML Schemas

1.4. 在网页中使用 xml 数据

11. AJAX

1.1. 同步，异步通讯

1.2. Ajax 请求

AJAX 指异步 JavaScript 及 XML（Asynchronous JavaScript And XML）

Ajax 的核心是 JavaScript 对象 XmlHttpRequest。该对象在 Internet Explorer 5 中首次引入，它是一种支持异步请求的技术。简而言之，XmlHttpRequest 使您可以使用 JavaScript 向服务器提出请求并处理响应，而不阻塞用户。

```
  new Ajax.Request("url",
    {
      method: "get",
      onSuccess: functionName
    }
  );
  ...

function functionName(ajax) {
  do something with ajax.responseText;
}
```
JS

```
new Ajax.Request("url",
  {
    method: "post",    // optional
    parameters: { name: value, name: value, ..., name: value },
    onSuccess: functionName,
    onFailure: functionName,
    onException: functionName
  }
);
```
JS

1.3. 优缺点和不使用的比较

- **优点**

AJAX 不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的 Web 应用程序的技术。

通过 AJAX，您的 JavaScript 可使用 JavaScript 的 XMLHttpRequest 对象来直接与服务器进行通信。通过这个对象，您的 JavaScript 可在不重载页面的情况与 Web 服务器交换数据。

AJAX 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可使网页从服务器请求少量的信息，而不是整个页面。

AJAX 可使因特网应用程序更小、更快，更友好。

AJAX 是一种独立于 Web 服务器软件的浏览器技术。

AJAX 基于下列 Web 标准：

JavaScript XML HTML CSS 在 AJAX 中使用的 Web 标准已被良好定义，并被所有的主流浏览器支持。AJAX 应用程序独立于浏览器和平台。

Web 应用程序较桌面应用程序有诸多优势；它们能够涉及广大的用户，它们更易安装及维护，也更易开发。

不过，因特网应用程序并不像传统的桌面应用程序那样完善且友好。

通过 AJAX，因特网应用程序可以变得更完善，更友好。

最大优点：更迅捷的响应速度，就是能在不更新整个页面的前提下维护数据。这使得 Web 应用程序更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息。

- **缺点**
  - 对应用 Ajax 最主要的批评就是，它可能破坏浏览器后退按钮的正常行为。
  - 使用动态页面更新使得用户难于将某个特定的状态保存到收藏夹中
  - 网络延迟——即用户发出请求到服务器发出响应之间的间隔——需要慎重考虑。
  - 一些手持设备（如手机、PDA 等）现在还不能很好的支持 Ajax;
  - 用 JavaScript 作的 Ajax 引擎，JavaScript 的兼容性和 DeBug 都是让人头痛的事；
  - Ajax 的无刷新重载，由于页面的变化没有刷新重载那么明显，所以容易给用户带来困扰——用户不太清楚现在的数据是新的还是已经更新过的；现有的解决有：在相关位置提示、数据更新的区域设计得比较明显、数据更新后给用户提示等；
  - 对串流媒体的支持没有 FLASH、Java Applet 好；

1.4. XSS

XSS 又叫 CSS (Cross Site Script) ，跨站脚本攻击。它指的是恶意攻击者往 Web 页面里插入恶意脚本代码，当用户浏览该页之时，嵌入其中 Web 里面的脚本代码会被执行，从而达到恶意攻击用户的特殊目的。XSS 属于被动式的攻击，因为其被动且不好利用，所以许多人常忽略其危害性。
XSS 攻击分成两类，一类是来自内部的攻击，主要指的是利用程序自身的漏洞，构造跨站语句，如:dvbbs 的 showerror.asp 存在的跨站漏洞。

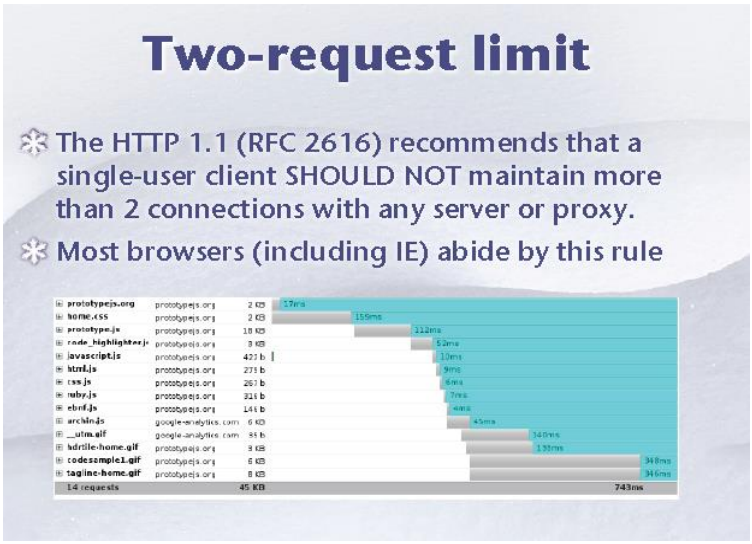
另一类则是来自外部的攻击，主要指的自己构造 XSS 跨站漏洞网页或者寻找非目标机以外的有跨站漏洞的网页。如当我们要渗透一个站点，我们自己构造一个有跨站漏洞的网页，然后构造跨站语句，通过结合其它技术，如社会工程学等，欺骗目标服务器的管理员打开。

攻击 Yahoo Mail 的 Yamanner 蠕虫是一个著名的 XSS 攻击实例。

1.5. AJAX LIMITS

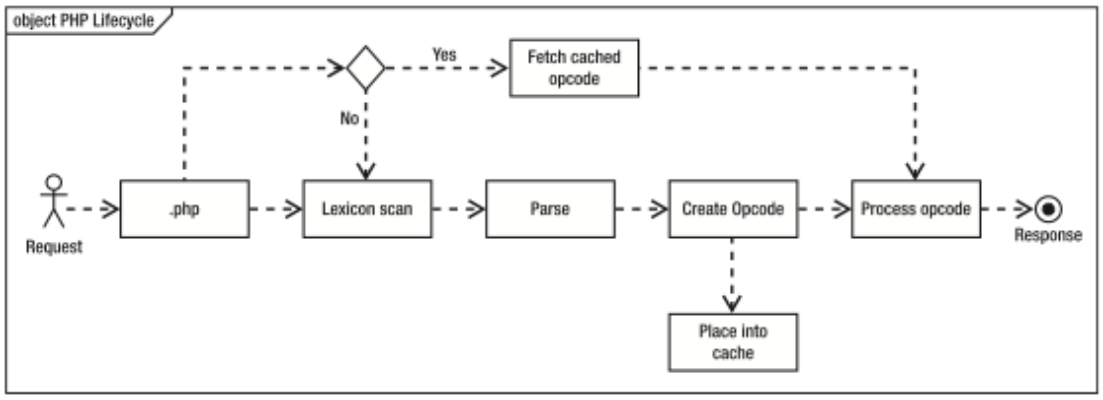
- SOP(same origin policy)

- Two-request limit



12. PHP

1.1. Lifecycle



1.2. cgi\fastcgi\php_mod 模式

1.3. 静态网页 动态网页 DHTML

1.4. 基本语法

1.5. HTML form

1.6. 浏览器-server 交互

1.7. 验证：不同方式

1.8. 正则表达式

如何表示至少“$1000.00”
/^$[1-9]\d{3}\.\d{2}/

2. 在对正则表达式的功能和作用有了初步的了解之后，我们就来具体看一下正则表达式的语法格式。

   正则表达式的形式一般如下：

   /love/

   其中位于“/”定界符之间的部分就是将要在目标对象中进行匹配的模式。用户只要把希望查找匹配对象的模式内容放入“/”定界符之间即可。为了能够使用户更加灵活的定制模式内容，正则表达式提供了专门的“元字符”。所谓元字符就是指那些在正则表达式中具有特殊意义的专用字符，可以用来规定其前导字符（即位于元字符前面的字符）在目标对象中的出现模式。

   较为常用的元字符包括： “+”， “*”，以及 “?”。其中，“+”元字符规定其前导字符必须在目标对象中连续出现一次或多次，“*”元字符规定其前导字符必须在目标对象中出现零次或连续多次，而“?”元字符规定其前导对象必须在目标对象中连续出现零次或一次。

   下面，就让我们来看一下正则表达式元字符的具体应用。

   /fo+/

   因为上述正则表达式中包含“+”元字符，表示可以与目标对象中的 “fool”, “fo”, 或者 “football” 等在字母 f 后面连续出现一个或多个字母 o 的字符串相匹配。

   /eg*/

   因为上述正则表达式中包含“*”元字符，表示可以与目标对象中的 “easy”, “ego”, 或者 “egg”等

在字母 e 后面连续出现零个或多个字母 g 的字符串相匹配。

/Wil?/

因为上述正则表达式中包含“？”元字符，表示可以与目标对象中的 “Win”, 或者 “Wilson”,等在字母 i 后面连续出现零个或一个字母 l 的字符串相匹配。

除了元字符之外，用户还可以精确指定模式在匹配对象中出现的频率。例如，

/jim{2,6}/

上述正则表达式规定字符 m 可以在匹配对象中连续出现 2-6 次，因此，上述正则表达式可以同 jimmy 或 jimmmmmy 等字符串相匹配。

在对如何使用正则表达式有了初步了解之后，我们来看一下其它几个重要的元字符的使用方式。

\s：用于匹配单个空格符，包括 tab 键和换行符；

\S：用于匹配除单个空格符之外的所有字符；

\d：用于匹配从 0 到 9 的数字；

\w：用于匹配字母，数字或下划线字符；

\W：用于匹配所有与\w 不匹配的字符；

. ：用于匹配除换行符之外的所有字符。

（说明：我们可以把\s 和\S 以及\w 和\W 看作互为逆运算）

下面，我们就通过实例看一下如何在正则表达式中使用上述元字符。

/\s+/

上述正则表达式可以用于匹配目标对象中的一个或多个空格字符。

/\d000/

如果我们手中有一份复杂的财务报表，那么我们可以通过上述正则表达式轻而易举的查找到所有总额达千元的款项。

3. 除了我们以上所介绍的元字符之外，正则表达式中还具有另外一种较为独特的专用字符，即定位符。定位符用于规定匹配模式在目标对象中的出现位置。

   较为常用的定位符包括： “^”, “$”, “\b” 以及 “\B”。其中，“^”定位符规定匹配模式必须出现在目标字符串的开头，“$”定位符规定匹配模式必须出现在目标对象的结尾，\b 定位符规定匹配模式必须出现在目标字符串的开头或结尾的两个边界之一，而“\B”定位符则规定匹配对象必须位于目标字符串的开头和结尾两个边界之内，即匹配对象既不能作为目标字符串的开头，也不能作为目标字符串的结尾。同样，我们也可以把“^”和“$”以及“\b”和“\B”看作是互为逆运算的两组定位符。举例来说：

   /^hell/

   因为上述正则表达式中包含“^”定位符，所以可以与目标对象中以 “hell”, “hello”或 “hellhound” 开头的字符串相匹配。

   /ar$/

因为上述正则表达式中包含“$”定位符，所以可以与目标对象中以 “car”, “bar”或 “ar” 结尾的字符串相匹配。

4. /\bbom/

因为上述正则表达式模式以“\b”定位符开头，所以可以与目标对象中以 “bomb”, 或 “bom”开头的字符串相匹配。

/man\b/

因为上述正则表达式模式以“\b”定位符结尾，所以可以与目标对象中以 “human”, “woman”或 “man”结尾的字符串相匹配。

为了能够方便用户更加灵活的设定匹配模式，正则表达式允许使用者在匹配模式中指定某一个范围而不局限于具体的字符。例如：

/[A-Z]/

上述正则表达式将会与从 A 到 Z 范围内任何一个大写字母相匹配。

/[a-z]/

上述正则表达式将会与从 a 到 z 范围内任何一个小写字母相匹配。

/[0-9]/

上述正则表达式将会与从 0 到 9 范围内任何一个数字相匹配。

/([a-z][A-Z][0-9])+/

上述正则表达式将会与任何由字母和数字组成的字符串，如 “aB0” 等相匹配。这里需要提醒用户注意的一点就是可以在正则表达式中使用 “()” 把字符串组合在一起。“()”符号包含的内容必须同时出现在目标对象中。因此，上述正则表达式将无法与诸如 “abc”等的字符串匹配，因为 “abc”中的最后一个字符为字母而非数字。

如果我们希望在正则表达式中实现类似编程逻辑中的“或”运算，在多个不同的模式中任选一个进行匹配的话，可以使用管道符 “|”。例如：

/to|too|2/

上述正则表达式将会与目标对象中的 “to”, “too”, 或 “2” 相匹配。

正则表达式中还有一个较为常用的运算符，即否定符 “[^]”。与我们前文所介绍的定位符 “^” 不同，否定符 “[^]”规定目标对象中不能存在模式中所规定的字符串。例如：

/[^A-C]/

上述字符串将会与目标对象中除 A，B，和 C 之外的任何字符相匹配。一般来说，当“^”出现在 “[]” 内时就被视做否定运算符；而当“^”位于“[]”之外，或没有“[]”时，则应当被视做定位符。

最后，当用户需要在正则表达式的模式中加入元字符，并查找其匹配对象时，可以使用转义符“\”。例如：

/Th*/

上述正则表达式将会与目标对象中的“Th*”而非“The”等相匹配。

4.1. 面向对象

13. 框架

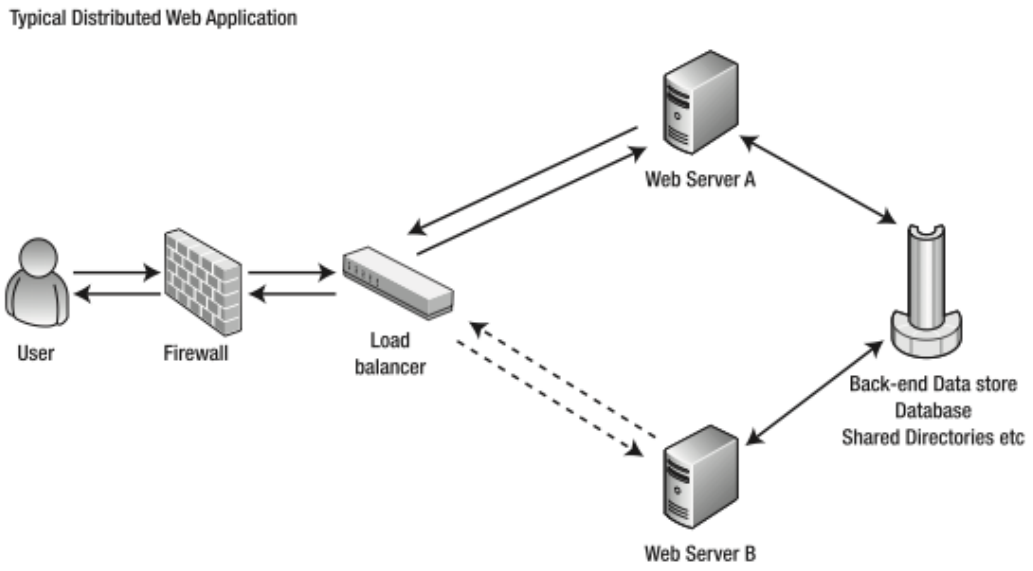
- 概念：框架就是通过提供一个开发 Web 程序的基本架构，PHP 开发框架把 PHPWeb 程序开发摆到了流水线上。换句话说，PHP 开发框架有助于促进快速软件开发（RAD），这节约了你的时间，有助于创建更为稳定的程序，并减少开发者的重复编写代码的劳动。这些框架还通过确保正确的数据库操作以及只在表现层编程的方式帮助初学者创建稳定的程序。PHP 开发框架使得你可以花更多的时间去创造真正的 Web 程序，而不是编写重复性的代码
- 模板引擎： 模板引擎（这里特指用于 Web 开发的模板引擎）是为了使用户界面与业务数据（内容）分离而产生的，它可以生成特定格式的文档，用于网站的模板引擎就会生成一个标准的 HTML 文档。
- MVC
- 框架的优缺点

14. 大规模 Web 服务开发技术

- php 优化技术
  - 声明静态方法。动态的要慢 50%。
  - 多使用局部变量
  - 提前计算循环长度

- 循环中不要使用方法
- 使用 foreach 替代 while、for 遍历数组
- 小文件数据使用 fread，大文件使用 file_get_contents
- 少用 PCRE

- 负载均衡



- 负载均衡（Load Balance）建立在现有网络结构之上，它提供了一种廉价、有效、透明的方法，来扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络
- 的灵活性和可用性。负载均衡有两方面的含义：首先，大量的并发访问或数据流量分担到多台节点设备上分别处理，减少用户等待响应的时间；其次，单个重负载的运算分担到多台节点设备上做并行处理，每个节点设备处理结束后，将结果汇总，返回给用户，系统处理能力得到大幅度提高。
- 分为纯软件、单独负载均衡服务器软件解决方案、物理负载均衡组件、负载均衡服务四种
- http://baike.baidu.com/view/1234431.htm