
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

----o0o----



Web Information System

Version 1.0

NGUYỄN HỒNG PHƯƠNG
TRƯỞNG ĐIỆU LINH

Hà Nội, tháng 6-2009

Contents

Chapter 1.	10
1. Brief history of Internet and World Wide Web	10
1.1 History of Internet	10
1.1.1. Packet switching	10
1.1.2. Networks that led to the Internet	10
1.1.3 TCP/IP	12
1.1.4. ARPANET to Several Federal Wide Area Networks: MILNET, NSI, and NSFNet	13
1.1.5. Transition towards the Internet	15
1.1.6. TCP becomes worldwide	15
1.2. History of World Wide Web	16
1.2.1- 1980-1991: Development of the World Wide Web	16
1.2.2- 1992-1995: Growth of the WWW	17
1.2.3- 1996-1998: Commercialization of the WWW	18
1.2.4- 1999-2001: "Dot-com" boom and bust	18
1.2.5- 2002-Present: The Web becomes ubiquitous	18
1.3 Statistics on the number of internet user in world over the years	19
1.4 Definition of WWW	20
2.What is NOT a web system	20
2.1.Examples of Not web system	21
2.1.1.Email	21
2.1.2. FTP	23
3. Typical application	24
3.1. Blog	24
3.2. SNS	25
3.3 Search engine	25
3.4. Google map	26
4. Social Impact	27
4.1. Advantages	28
4.2. Disadvantages	28
Chapter 2.	29
1. Server-client model	29
1.1. Introduce client-server model of web system	29
1.2. Example of some Web server	31
1.3. Example of some Web client:	33
2. DNS and URL	37
2.1. DNS (Domain Name System)	37
2.2. URL	39
3.HTTP Request and Response	40
3.1.HTTP Protocol Overview	40
3.2. Request Message Structure	42
3.3. Response Message Structure	45
4.Character Encoding	46
4.1.Unicode	46
4.1.1.Over view	46
4.1.2.Character sets, coded character sets, and encodings	47

4.1.3. One character set, multiple encodings	48
4.1.4. Character escapes	49
4.1.5. Document character set	49
4.2. Different Encodings	49
4.2.1. UTF-8	49
4.2.2. EUC	51
5. Media Type	52
5.1. Type text: Human-readable text and source code	53
5.2. Type video: Video	53
5.3. Type image	54
5.4. Type audio: Audio	55
5.5. How multimedia data are included in HTTP message	55
Chapter 3.	57
1. Markup and rendering	57
1.1. Markup language	57
1.2. Visualise an HTML document using an web browser	58
2. Syntax of HTML	59
2.1. Structure of an HTML Document	59
2.2. Introduce basic tags	60
2.2.1. Page formatting tags:	60
2.2.2. Text style tags:	61
2.2.3. Tables	61
2.2.4. The Image Tag and the Src Attribute	62
2.2.5. Hyperlinks, Anchors, and Links	62
2.2. Introduce frame tags and input elements	63
2.2.1. Frame tags	63
2.2.2 HTML Input Elements	64
Chapter 4	67
1. Stylesheet	67
1.1 What are style sheets?	67
1.2 How do I use a style sheet?	67
1.3 What are the advantages of style sheets?	67
1.4 The advantages of separating structure and presentation	68
1.5 Example	68
2. Accessibility	72
2.1 Structure vs. Presentation	73
2.2 Text equivalents	73
2.2.1 Overview of technologies	74
2.2.2 Backward Compatibility	74
2.2.3 Alternative pages	75
2.3 Device-independent control for embedded interfaces	76
2.4 Navigation	76
2.5 Comprehension	77
2.5.1 Writing style	77
2.5.2 Multimedia equivalents	78
2.6 Content negotiation	78
2.7 Automatic page refresh	78
2.8 Screen flicker	79

2.9 Bundled documents.....	79
2.10 Validation.....	79
2.10.1 Automatic validators.....	79
2.10.2 Repair tools.....	80
2.10.3 User scenarios.....	80
2.10.4 Spell and grammar checks.....	81
2.11 Browser Support.....	81
2.12 Technologies Reviewed for Accessibility.....	81
2.13 Audio information.....	81
2.14 Visual information and motion.....	82
2.15 Collated text transcripts.....	82
3. Link CSS with HTML.....	83
3.1 In-line.....	83
3.2 Internal.....	83
3.3 External.....	84
4. Systax of CSS.....	85
4.1 What is CSS?.....	86
4.2 CSS Syntax.....	86
4.3 The simple CSS example.....	89
Chapter 5.....	91
1.Dynamic Pages.....	91
1.1 What are Dynamic Pages.....	91
1.2 The difference between static and dynamic page web.....	91
1.3 The Advantage of Dynamic Pages.....	92
1.4 The Disadvantages of Dynamic Pages.....	93
1.5 Example Dynamic pages.....	93
2 Client-side and Server-side.....	94
2.1 Server side.....	94
2.2 Client side.....	94
3. Three-Tier sytem.....	95
3.1 Presentation Tier.....	96
3.2 Application Tier (Business Logic/Logic Tier).....	96
3.3 Data Tier.....	96
Chapter 6.....	98
1. Scritng in a web browser.....	98
1.1 JavaScript.....	98
1.2 Applet.....	98
1.2.1 Advantages of Applet:.....	98
1.2.2 Disadvantages of Java Applet:.....	99
1.3 Flash.....	99
1.3.1 What do I have against Flash?.....	100
1.3.2 What I like about Flash development?.....	100
1.3.3 So when should you use Flash?.....	100
2.Javascript.....	101
2.1 Syntax javascript.....	101
2.1.1 JavaScript Variables.....	101
2.1.2 JavaScript Functions.....	102
Exlanation of code.....	104

2.1.4 Control structures in JavaScript	104
2.2 Using javascript to varity data in a form.....	108
2.2.1 Description	109
2.2.2 Introduction.....	109
2.2.3 Data Validation Precautions	110
2.2.4 Available validation functions	111
2.2.5 Using the Validation Functions	111
2.2.6 Specifying Custom Messages	112
2.2.7 Form Validation Command	112
2.2.8 Creating Custom Validation Functions.....	113
2.2.9 Smart Validation Behavior	114
2.2.10 Technical Specifications of the Validation Function.....	115
Chapter 7	117
1. CGI.....	117
1.1 What is CGI?.....	117
1.2 How CGI works	119
1.3 Example web using CGI languges used for CGI	119
2. Form, GET and POST.....	120
3. PHP.....	122
3.1 PHP Introduction	122
3.2 Php Syntax	123
3.3 PHP MySQL Connect to a Database	124
Chapter 8.....	127
1.SESSION MANAGEMENT	127
1.1.Stateless Nature of HTTP	127
1.2.Need of Tracking Client Identity and State	128
1.3.Session Concept	128
1.4.Some of Session Tracking Techniques.....	129
1.4.1.URL Rewriting.....	129
1.4.2.Hidden Form Field.....	129
1.4.3.Cookies (See more on Chap2)	130
2.COOKIE	132
2.1.Cookie Overview	132
2.2.Cookie function.....	132
2.3.Type of Cookie	133
2.3.1.Session Cookies	133
2.3.2.“Persistent” Cookies – Permenant Cookie- Stored Cookie	133
3.SERVLET.....	135
3.1.Static Web vs Dynamic Web	135
3.2.CGI-Solution for dynamic content generation.....	135
3.3.Shortcomings of CGI.....	136
3.4.Servlet- A Solution to CGI-Problem.....	137
3.5.Merits and Demerits of Servlet	138
3.6.Web container Concept.....	138
3.7.Servlet’s Life Cycle	138
3.8.HTTP Request Processing Life Cycle	139
3.9.Implement Servlet in Java.....	139
4.JAVA SERVER PAGES	141

4.1. What is a JSP Page.....	141
4.2. Benefits of JSP	141
4.3. Servlet and JSP	142
4.4. Architecture of JSP	142
4.5. Translation Unit	143
4.6. JSP Life Cycle.....	143
4.7. JSP Scripting elements.....	144
4.7.1. Scriptlets	144
4.7.2. Expression tag	144
4.7.3. Declaration tag	144
4.7.4. Comments	145
4.8. JSP Directives	145
4.8.1. Page Directives	145
4.8.2. Include Directives	146
4.8.3. TagLib Directive	147
4.9. JSP Action.....	147
Chapter 9.	149
1. Public Key Infrastructure	149
1.1. What is PKI ?	149
1.2. Components of PKI.....	149
1.3. Issuing a certificate	151
1.4. CA architectures.....	152
1.4.1. Single Architecture	152
1.4.2. Hierarchical Architecture.....	152
1.4.3. Mesh Architecture.....	153
1.5. PKI Architecture Overview	153
1.6. Digital Signature	154
1.6.1. Need of Digital Signature	155
1.6.2. What the digital signature made of ?	155
1.6.3. Working of digital signature	155
1.6.4. Validating data integrity	156
1.6.5. Drawback of digital signature	156
1.7. Digital Certificates	157
1.7.1. Standards and features of digital certificates	157
1.7.2. Verify the authenticity of the sender.....	158
2. Secure Socket Layer	161
2.1. What is SSL?.....	161
2.2. SSL Feature.....	161
2.3. SSL with client browser and server	161
2.4. Detail in SSL Handsake	163
2.4.1. Simple TLS/SSL handshake	163
2.4.2. Client-authenticated TLS/SSL handshake	165
2.4.3. Resumed TLS/SSL handshake by Session ID	166
2.5. TSL/SSL record protocol.....	167
2.6. Handshake protocol	169
2.7. Alert protocol	169
2.8. ChangeCipherSpec protocol	171
2.9. Application protocol	171

2.10.Example of Certificate	172
3.HTTPS	175
3.1.Overview of HTTPS	175
3.2.HTTPS Function	175
3.3.Browser integration.....	176
3.4.Application of HTTPS	176
3.4.1.Online payment and Online Shopping.....	176
3.4.2.Internet Banking.....	179
3.4.3.Manage Certificate.....	181
Chapter 10.	183
1.Clustering	183
1.1.Cluster	183
1.2.Cluster categorizations.....	183
1.2.1.High-availability (HA) clusters.....	183
1.2.2.Load-balancing clusters	184
1.2.3.Compute clusters.....	184
1.2.4.Grid computing	184
1.3.Technologies	185
1.4.Example of clustered web server system	185
2.Load balancing	188
2.1.What 's Load balancing	188
2.2.For Internet Service.....	188
2.3.Persistence.....	189
2.4.Load balancer features	190
2.5.Implement Load balancing	191
2.5.1.Local DNS Caching	191
2.5.2.Using Standard DNS for load balancing.....	192
2.5.3.HTTP Redirect	192
2.6.Some Alogrithm for Load balancing	194
2.6.1.Random	194
2.6.2.Round Robin	194
2.6.3.Weighted Round Robin (called Ratio on the BIG-IP).....	194
2.6.4.Dynamic Round Robin (Called Dynamic Ratio on the BIG-IP)	195
2.6.5.Fastest	195
2.6.6.Least Connections.....	195
2.6.7.Observed	196
2.6.8.Predictive	196
2.6.9.Locality Aware Request Distribution	196
2.6.10.Genetic Based GDE Approach in Load Balancing.....	197
3.Round robin DNS	199
3.1.What 's round robin DNS	199
3.2.Implement Round Robin DNS.....	199
3.2.1.DNS load balancing implementation (Multiple CNAMEs).....	199
3.2.2.DNS load balancing implementation (Multiple A Records).....	200
3.3.Performance	200
3.4.Drawbacks.....	201
Chapter 11.	202
1.Web Services	202

1.1. Concept of WebServices.....	202
1.2.Generic Architecture of WebServices	202
1.3.Life Cycle of a Web Service	203
1.4.XML Web Services.....	204
1.4.1.Concept of XML Web Services.....	204
1.4.2.XML Web Services Infrastructure.....	205
1.4.3.Client and XML Web Service Communication.....	207
1.5 Advantages and Disadvantages compared to the CORBA & RMI	208
1.5.1.Advantages.....	208
1.5.2.Disadvantages	208
1.6.SDL (Service Description Language).....	208
1.6.1The need of SDL.....	209
1.6.2.Definition of SDL	209
1.7.WSDL (Web Service Description Language).....	211
1.7.1.Concept of WSDL.....	211
1.7.2.Element of a WSDL File.....	212
1.8.SOAP (Simple Object Access Protocol).....	212
1.8.1.Overview SOAP.....	212
1.8.2.SOAP Specifications.....	213
1.8.3.Working of SOAP.....	214
1.8.4.SOAP Message Architecture	215
1.8.5.SOAP request message and SOAP response message.....	216
11.8.6.SOAP Data Types	217
1.9.UDDI.....	218
1.9.1.Overview of UDDI	218
1.9.2.Purpose.....	219
1.9.3.Registry Type.....	219
1.9.4.UDDI Architecture.....	220
1.9.5.Working of UDDI.....	221
1.9.6.Data Structures.....	222
1.10.XML (See more in next chapter).	222
2.XML	224
2.1.Evolution of XML.....	224
2.2.Features of XML.....	224
2.3.XML Markup	224
2.4.Benefits of XML	225
2.5.XML Document Structure	225
2.6.XML Document Life Cycle	226
2.7.Well-formed XML document	226
2.8.Classification of character data.....	227
3.XHTML	228
3.1. What is XHTML	228
3.2. Why the need for XHTML?.....	228
3.3. Valid XHTML documents	228
3.3.1.Root element	229
3.3.2. DOCTYPEs.....	229
3.3.3. XML declaration	230
3.3.4.Using XHTML with other namespaces	230

3.4.Differences with HTML	231
3.4.1.Documents must be well-formed	231
3.4.2. Element and attribute names must be in lower case	231
3.4.3. For non-empty elements, end tags are required	232
3.4.4. Attribute values must always be quoted	232
3.4.5. Attribute Minimization	232
3.4.6. Empty Elements	233
3.4.7. White Space handling in attribute values.....	233
3.4.8. Script and Style elements.....	233
Chapter 12	234
1 Web 2.0	234
1.1 definition	234
1.2 Characteristics.....	234
1.3 Examples.....	239
2.DOM	240
2.1 Introduction.....	240
3 AJAX	250
3.1 What is AJAX?	250
3.2 How does it work?	251
3.3 AJAX - applications popular	252
3.4 THE weakness of AJAX.....	253
4 RIA	254
4.1 Concept of RIA	254
4.2 Examples of RIA frameworks	254
Chapter 13	260
1 Web 3.0	260
1.1 The breakthrough of new media channels	260
1.2 definition	260
1.3 Characteristics.....	260
2 Metadata	262
2.1 Definitions.....	262
2.2 Purpose.....	264
2.3 Examples of metadata	264
2.4 Use	265
2.5 Types of metadata	266
3 RDF	267
3.1 Introduction to RDF	267
3.1.1 Definitions.....	267
3.1.2 RDF - Examples of Use	268
RDF Example.....	278
4. SPARQL Query Language for RDF	280
4.1 Introduction.....	280
4.2 SPARQL Syntax	280
4.2.2 Syntax for Literals.....	282
4.2.3 Syntax for Query Variables	284
4.2.4 Syntax for Blank Nodes	284
4.2.5 Syntax for Triple Patterns	285
4.2.6 Predicate-Object Lists.....	286

4.2.7 Object Lists	286
4.2.8 RDF Collections.....	287
4.2.9 rdf:type.....	287
5.1. Introduction.....	288
5.2 The three sublanguages of OWL	289
5.3 Language Synopsis	291
5.3.1 OWL Lite Synopsis.....	291
5.3.2 OWL DL and Full Synopsis	292
5.4. Language Description of OWL Lite	293
5.4.1 OWL Lite RDF Schema Features	293
5.4.2 OWL Lite Equality and Inequality	294
5.4.3 OWL Lite Property Characteristics	295
5.4.4 OWL Lite Property Restrictions	296
5.4.5 OWL Lite Restricted Cardinality.....	297
5.4.6 OWL Lite Class Intersection	299
5.4.7 OWL Lite Header Information	299
5.4.8 OWL Lite Annotation Properties.....	299
5.4.9 OWL Lite Versioning	299
5.5. Incremental Language Description of OWL DL and OWL Full.....	299

Chapter 1.

1. Brief history of Internet and World Wide Web

1.1 History of Internet

The Internet was the result of some visionary thinking by people in the early 1960s who saw great potential value in allowing computers to share information on research and development in scientific and military fields.

J.C.R. Licklider of MIT, first proposed a global network of computers in 1962, and moved over to the Defense Advanced Research Projects Agency (DARPA) in late 1962 to head the work to develop it.

Leonard Kleinrock of MIT and later UCLA developed the theory of packet switching, which was to form the basis of Internet connections.

Lawrence Roberts of MIT connected a Massachusetts computer with a California computer in 1965 over dial-up telephone lines. It showed the feasibility of wide area networking, but also showed that the telephone line's circuit switching was inadequate. Kleinrock's packet switching theory was confirmed. Roberts moved over to DARPA in 1966 and developed his plan for ARPANET. These visionaries and many more left unnamed here are the real founders of the Internet.

1.1.1. Packet switching

At the tip of the inter-networking problem lay the issue of connecting separate physical networks to form one logical network, with much wasted capacity inside the assorted separate networks.

During the 1960s, Donald Davies (NPL), Paul Baran (RAND Corporation), and Leonard Kleinrock (MIT) developed and implemented packet switching. Early networks used for the command and control of nuclear forces were message switched, not packet-switched, although current strategic military networks are, indeed, packet-switching and connectionless. Baran's research had approached packet switching from studies of decentralisation to avoid combat damage compromising the entire network

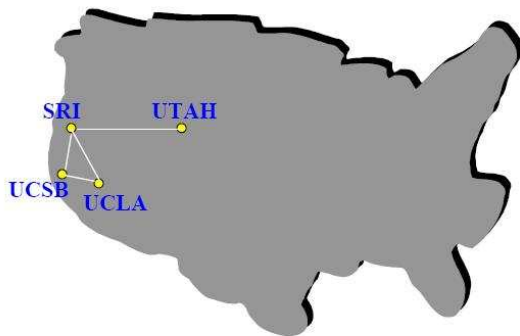
1.1.2. Networks that led to the Internet

ARPANET

Promoted to the head of the information processing office at DARPA, Robert Taylor intended to realize Licklider's ideas of an interconnected networking system. Bringing in Larry Roberts from MIT, he initiated a project to build such a network.

The first ARPANET link was established between the University of California, Los Angeles and the Stanford Research Institute on 22:30 hours on October 29, 1969.

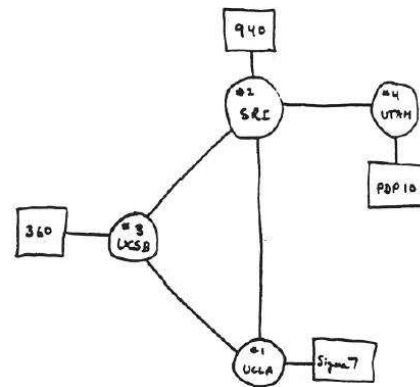
By December 5, 1969, a 4-node network was connected by adding the University of Utah and the University of California, Santa Barbara.



A complete network with 4 nodes, 56kbps

SRI: Stanford Research Institute
 UCLA: University California Los Angeles
 UCSB: University of California, Santa Barbara
 UTAH: University of Utah

source: <http://www.cybergeography.org/atlas/historical.html>



THE ARPA NETWORK

DEC 1969

4 NODES

FIGURE 6.2 Drawing of 4 Node Network
 (Courtesy of Alex McKenzie)

A complete network with 4 nodes, 56kbps

Building on ideas developed in ALOHAnet, the ARPANET grew rapidly. By 1981, the number of hosts had grown to 213, with a new host being added approximately every twenty days.

ARPANET became the technical core of what would become the Internet, and a primary tool in developing the technologies used.

X.25 and public access

Following on from ARPA's research, packet switching network standards were developed by the International Telecommunication Union (ITU) in the form of X.25 and related standards. In 1974, X.25 formed the basis for the SERCnet network between British academic and research sites, which later became JANET.

The initial ITU Standard on X.25 was approved in March 1976. This standard was based on the concept of virtual circuits.

The first dial-in public networks used asynchronous TTY terminal protocols to reach a concentrator operated by the public network. Some public networks, such as CompuServe used X.25 to multiplex the terminal sessions into their packet-switched backbones, while others, such as Tymnet, used proprietary protocols.

In 1979, CompuServe became the first service to offer electronic mail capabilities and technical support to personal computer users. The company broke new ground again in 1980 as the first to offer real-time chat with its CB Simulator. There were also the America Online (AOL) and Prodigy dial in networks and many bulletin board system (BBS) networks such as FidoNet. FidoNet in

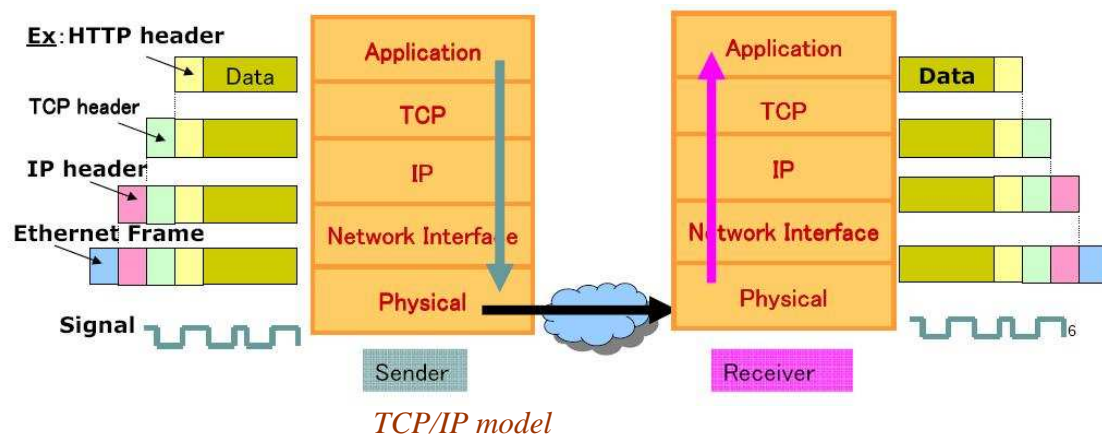
particular was popular amongst hobbyist computer users, many of them hackers and amateur radio operators.

UUCP (the Unix to Unix Copy Protocol)

UUCP was invented in 1978 at Bell Labs. Usenet was started in 1979 based on UUCP. Newsgroups, which are discussion groups focusing on a topic, followed, providing a means of exchanging information throughout the world .

UUCP networks spread quickly due to the lower costs involved, ability to use existing leased lines, X.25 links or even ARPANET connections, and the lack of strict use policies (commercial organizations who might provide bug fixes) compared to later networks like CSnet and Bitnet. All connects were local.

1.1.3 TCP/IP

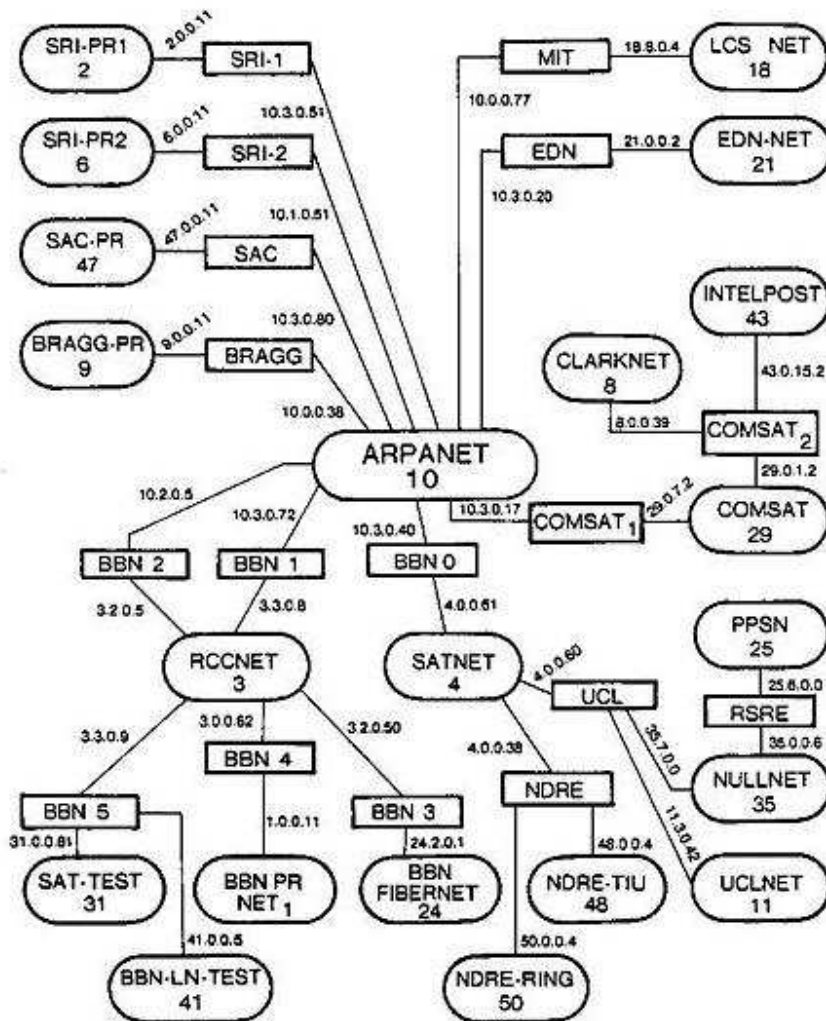


For the first connections between the computers, the Network Working Group developed the Network Control Protocol.

The Internet matured in the 70's as a result of the TCP/IP architecture first proposed by Bob Kahn at BBN and further developed by Kahn and Vint Cerf at Stanford and others throughout the 70's.

DARPA sponsored or encouraged the development of TCP/IP implementations for many operating systems and then scheduled a migration of all hosts on all of its packet networks to TCP/IP. On January 1, 1983, TCP/IP protocols became the only approved protocol on the ARPANET, replacing the earlier NCP protocol.

POSTEL 25 FEB 82



Map of the TCP/IP test network in January 1982

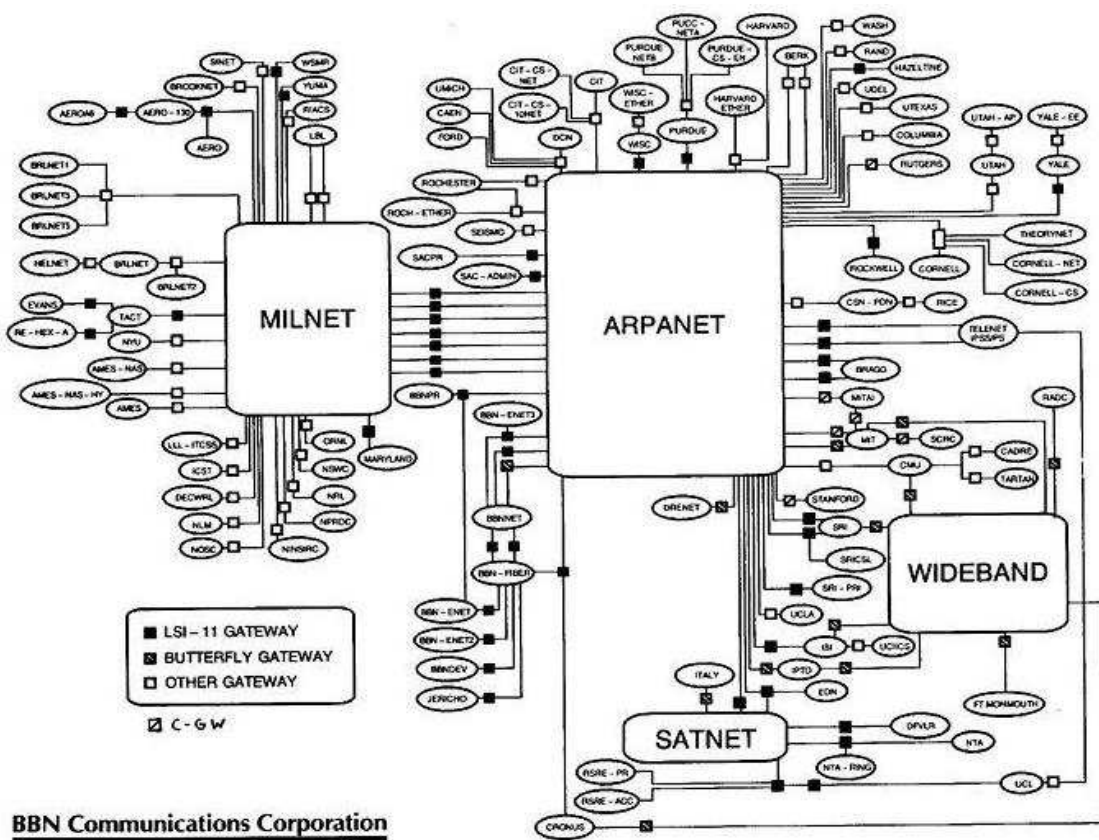
1.1.4. ARPANET to Several Federal Wide Area Networks: MILNET, NSI, and NSFNet

After the ARPANET had been up and running for several years, ARPA looked for another agency to hand off the network to; ARPA's primary mission was funding cutting edge research and development, not running a communications utility.

In 1983, the U.S. military portion of the ARPANET was broken off as a separate network, the MILNET. MILNET subsequently became the unclassified but military-only NIPRNET, in parallel with the SECRET-level SIPRNET and

JWICS for TOP SECRET and above. NIPRNET does have controlled security gateways to the public Internet.

The networks based around the ARPANET were government funded and therefore restricted to noncommercial uses such as research; unrelated commercial use was strictly forbidden. This initially restricted connections to military sites and universities. During the 1980s, the connections expanded to more educational institutions, and even to a growing number of companies such as Digital Equipment Corporation and Hewlett-Packard, which were participating in research projects or providing services to those who were.



BBN Technologies TCP/IP internet map early 1986

In the mid 1980s, all three of the National Aeronautics and Space Agency (NASA), the National Science Foundation (NSF), and the Department of Energy (DOE) branches developed the first Wide Area Networks based on TCP/IP. NASA developed the NASA Science Network, NSF developed CSNET and DOE evolved the Energy Sciences Network or ESNet.

In 1984 NSF developed CSNET exclusively based on TCP/IP. CSNET connected with ARPANET using TCP/IP, and ran TCP/IP over X.25, but it also supported departments without sophisticated network connections, using

automated dial-up mail exchange. This grew into the NSFNet backbone, established in 1986, and intended to connect and provide access to a number of supercomputing centers established by the NSF.

1.1.5. Transition towards the Internet

The term "internet" was adopted in the first RFC published on the TCP protocol (RFC 675: Internet Transmission Control Program, December 1974) as an abbreviation of the term *internetworking* and the two terms were used interchangeably. In general, an *internet* was any network using TCP/IP. It was around the time when ARPANET was interlinked with NSFNet in the late 1980s, that the term was used as the name of the network, Internet, being a large and global TCP/IP network.

The term "internet protocol" had also been used to refer to other networking systems such as Xerox Network Services.

Many sites unable to link directly to the Internet started to create simple gateways to allow transfer of e-mail, at that time the most important application. Sites which only had intermittent connections used UUCP or FidoNet and relied on the gateways between these networks and the Internet. Some gateway services went beyond simple e-mail peering, such as allowing access to FTP sites via UUCP or e-mail.

Finally, the Internet's remaining centralized routing aspects were removed. The EGP routing protocol was replaced by a new protocol, the Border Gateway Protocol (BGP), in order to allow the removal of the NSFNet Internet backbone network. In 1994, Classless Inter-Domain Routing was introduced to support better conservation of address space which allowed use of route aggregation to decrease the size of routing tables.

1.1.6. TCP becomes worldwide

Between 1984 and 1988 CERN began installation and operation of TCP/IP to interconnect its major internal computer systems, workstations, PCs and an accelerator control system. CERN continued to operate a limited self-developed system CERNET internally and several incompatible (typically proprietary) network protocols externally. There was considerable resistance in Europe towards more widespread use of TCP/IP and the CERN TCP/IP intranets remained isolated from the Internet until 1989.

In 1988 Daniel Karrenberg, from CWI in Amsterdam, visited Ben Segal, CERN's TCP/IP Coordinator, looking for advice about the transition of the European side of the UUCP Usenet network (much of which ran over X.25 links) over to TCP/IP. In 1987, Ben Segal had met with Len Bosack from the then still

small company Cisco about purchasing some TCP/IP routers for CERN, and was able to give Karrenberg advice and forward him on to Cisco for the appropriate hardware. This expanded the European portion of the Internet across the existing UUCP networks, and in 1989 CERN opened its first external TCP/IP connections. This coincided with the creation of Réseaux IP Européens (RIPE), initially a group of IP network administrators who met regularly to carry out co-ordination work together. Later, in 1992, RIPE was formally registered as a cooperative in Amsterdam.

At the same time as the rise of internetworking in Europe, ad hoc networking to ARPA and in-between Australian universities formed, based on various technologies such as X.25 and UUCPNet. These were limited in their connection to the global networks, due to the cost of making individual international UUCP dial-up or X.25 connections. In 1989, Australian universities joined the push towards using IP protocols to unify their networking infrastructures. AARNet was formed in 1989 by the Australian Vice-Chancellors' Committee and provided a dedicated IP based network for Australia.

The Internet began to penetrate Asia in the late 1980s. Japan, which had built the UUCP-based network JUNET in 1984, connected to NSFNet in 1989. It hosted the annual meeting of the Internet Society, INET'92, in Kobe. Singapore developed TECHNET in 1990, and Thailand gained a global Internet connection between Chulalongkorn University and UUNET in 1992

1.2. History of World Wide Web

The World Wide Web ("WWW" or simply the "Web") is a global information medium which users can read and write via computers connected to the Internet. The term is often mistakenly used as a synonym for the Internet itself, but the Web is a service that operates over the Internet, as e-mail does. The history of the Internet dates back significantly further than that of the World Wide Web.

1.2.1- 1980-1991: Development of the World Wide Web

In 1984 Berners-Lee considered its problems of information presentation: physicists from around the world needed to share data, with no common machines and no common presentation software. He wrote a proposal in March 1989 for "a large hypertext database with typed links", but it generated little interest.

By Christmas 1990, Berners-Lee had built all the tools necessary for a working Web: the HyperText Transfer Protocol (HTTP) 0.9, the HyperText Markup Language (HTML), the first Web browser (named WorldWideWeb, which was

also a Web editor), the first HTTP server software (later known as CERN httpd), the first web server and the first Web pages that described the project itself.

On August 6, 1991, Berners-Lee posted a short summary of the World Wide Web project on the alt.hypertext newsgroup. This date also marked the debut of the Web as a publicly available service on the Internet.

1.2.2- 1992-1995: Growth of the WWW

There was still no graphical browser available for computers besides the NeXT. This gap was filled in April 1992 with the release of Erwise, an application developed at Helsinki University of Technology, and in May by ViolaWWW, created by Pei-Yuan Wei, which included advanced features such as embedded graphics, scripting, and animation. Both programs ran on the X Window System for Unix.

Early Browser

The turning point for the World Wide Web was the introduction of the Mosaic web browser in 1993, a graphical browser developed by a team at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign (UIUC), led by Marc Andreessen.

The origins of Mosaic had begun in 1992. In November 1992, the NCSA at the University of Illinois (UIUC) established a website. In December 1992, Andreessen and Eric Bina, students attending UIUC and working at the NCSA, began work on Mosaic. They released an X Window browser in February 1993. It gained popularity due to its strong support of integrated multimedia, and the authors' rapid response to user bug reports and recommendations for new features.

The first Microsoft Windows browser was Cello, written by Thomas R. Bruce for the Legal Information Institute at Cornell Law School to provide legal information, since more lawyers had access to Windows than to Unix. Cello was released in June 1993.

After graduation from UIUC, Andreessen and James H. Clark, former CEO of Silicon Graphics, met and formed Mosaic Communications Corporation to develop the Mosaic browser commercially. The company changed its name to Netscape in April 1994, and the browser was developed further as Netscape Navigator.

Web organization

In May 1994 the first International WWW Conference, organized by Robert Cailliau, was held at CERN; the conference has been held every year since. In

April 1993 CERN had agreed that anyone could use the Web protocol and code royalty-free; this was in part a reaction to the perturbation caused by the University of Minnesota announcing that it would begin charging license fees for its implementation of the Gopher protocol.

In September 1994, Berners-Lee founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology with support from the Defense Advanced Research Projects Agency (DARPA) and the European Commission. It comprised various companies that were willing to create standards and recommendations to improve the quality of the Web. Berners-Lee made the Web available freely, with no patent and no royalties due. The World Wide Web Consortium decided that their standards must be based on royalty-free technology, so they can be easily adopted by anyone.

By the end of 1994, while the total number of websites was still minute compared to present standards, quite a number of notable websites were already active, many of whom are the precursors or inspiring examples of today's most popular services.

1.2.3- 1996-1998: Commercialization of the WWW

By 1996 it became obvious to most publicly traded companies that a public Web presence was no longer optional. Though at first people saw mainly the possibilities of free publishing and instant worldwide information, increasing familiarity with two-way communication over the "Web" led to the possibility of direct Web-based commerce (e-commerce) and instantaneous group communications worldwide. More dotcoms, displaying products on hypertext webpages, were added into the Web.

1.2.4- 1999-2001: "Dot-com" boom and bust

The low interest rates in 1998–99 helped increase the start-up capital amounts. Although a number of these new entrepreneurs had realistic plans and administrative ability, most of them lacked these characteristics but were able to sell their ideas to investors because of the novelty of the dot-com concept.

In 2001 the bubble burst, and many dot-com startups went out of business after burning through their venture capital and failing to become profitable.

1.2.5- 2002-Present: The Web becomes ubiquitous

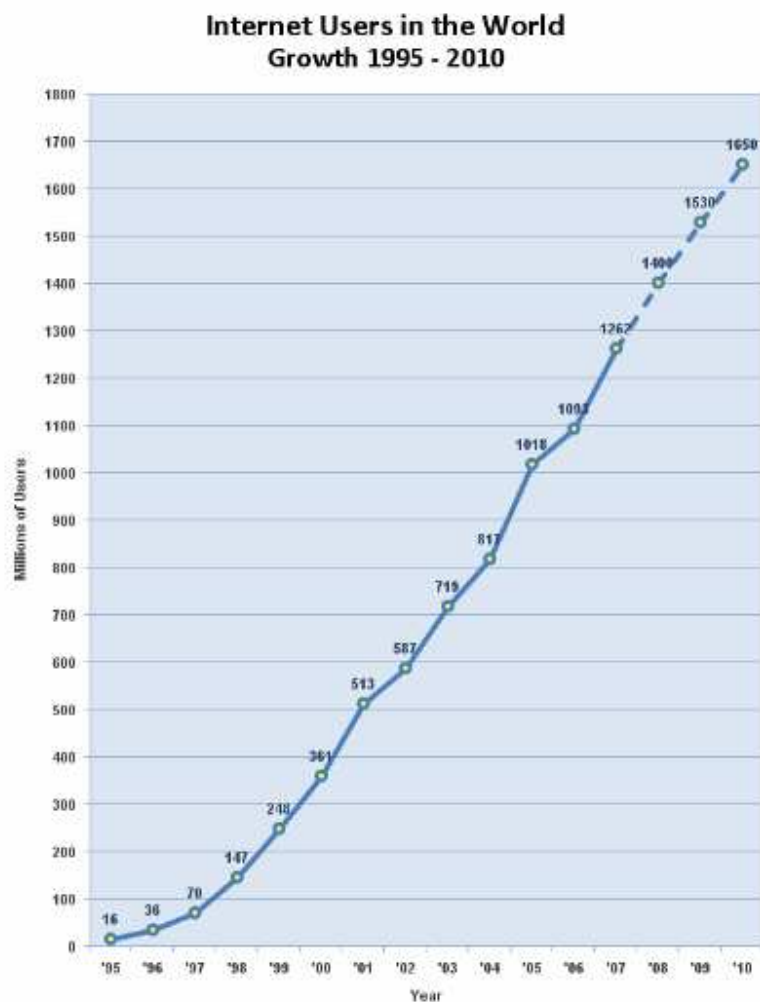
A handful of companies found success developing business models that helped make the World Wide Web a more compelling experience. These include airline booking sites, Google's search engine and its profitable approach to simplified,

keyword-based advertising, as well as Ebay's do-it-yourself auction site and Amazon.com's big selection of books.

This new era also begot social networking websites, such as MySpace, Xanga, Friendster, and Facebook, which, though unpopular at first, very rapidly gained acceptance in becoming a major part of youth culture.

1.3 Statistics on the number of internet user in world over the years

The amount of internet users in the world will keep increase every years. Statistic datas have shown significant growth level. In June 2008 is predicted 1,463 million people use internet and this number will reach 1,650 millions in 2010.



Source: www.internetworldstats.com - January, 2008
Copyright © 2008, Miniwatts Marketing Group

Statistics on the number of internet user in world over the years

Apparently the increasing users number is influenced by significant growth at Middle East and Africa during 2000-2008. But Asia, Europe and North America are still being the top three dominate more than 10% users of the world.

WORLD INTERNET USAGE AND POPULATION STATISTICS						
World Regions	Population (2008 Est.)	Internet Users Dec/31, 2000	Internet Usage, Latest Data	% Population (Penetration)	Usage % of World	Usage Growth 2000-2008
Africa	955,200,348	4,514,400	51,065,630	5.3 %	3.5 %	1,031.2 %
Asia	3,776,181,949	114,304,000	578,538,257	15.3 %	39.5 %	406.1 %
Europe	800,401,065	105,096,093	384,633,765	48.1 %	26.3 %	266.0 %
Middle East	197,090,443	3,284,800	41,939,200	21.3 %	2.9 %	1,176.8 %
North America	337,187,248	108,098,800	248,241,369	73.6 %	17.0 %	129.6 %
Latin America/Caribbean	576,091,673	18,008,919	139,009,209	24.1 %	9.5 %	609.3 %
Oceania / Australia	33,981,562	7,620,480	20,204,331	59.5 %	1.4 %	166.1 %
WORLD TOTAL	6,076,120,288	300,985,492	1,463,632,361	21.9 %	100.0 %	305.5 %

1.4 Definition of WWW

The World Wide Web (commonly abbreviated as "the Web") is a system of interlinked hypertext documents accessed via the Internet. With a Web browser, one can view Web pages that may contain text, images, videos, and other multimedia and navigate between them using hyperlinks.

Using concepts from earlier hypertext systems, the World Wide Web was started in 1989 by the English physicist Sir Tim Berners-Lee, now the Director of the World Wide Web Consortium, and later by Robert Cailliau, a Belgian computer scientist, while both were working at CERN in Geneva, Switzerland. In 1990, they proposed building a "web of nodes" storing "hypertext pages" viewed by "browsers" on a network, and released that web in December. Connected by the existing Internet, other websites were created, around the world, adding international standards for domain names & the HTML language. Since then, Berners-Lee has played an active role in guiding the development of Web standards (such as the markup languages in which Web pages are composed), and in recent years has advocated his vision of a Semantic Web.

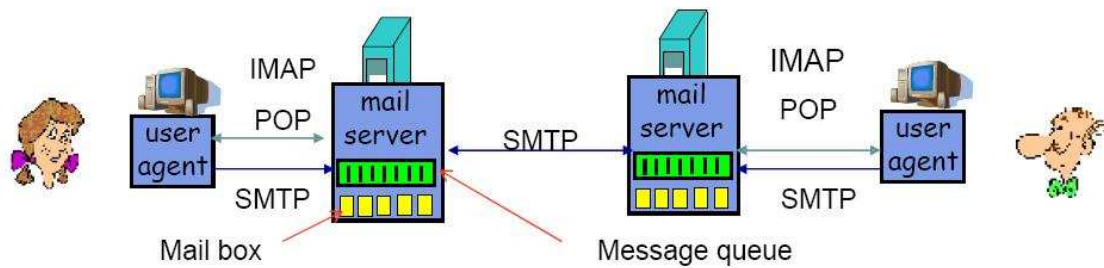
The World Wide Web enabled the spread of information over the Internet through an easy-to-use and flexible format. It thus played an important role in popularizing use of the Internet. Although the two terms are sometimes conflated in popular use, World Wide Web is not synonymous with Internet. The Web is an application built on top of the Internet.

2. What is NOT a web system

2.1.Examples of Not web system

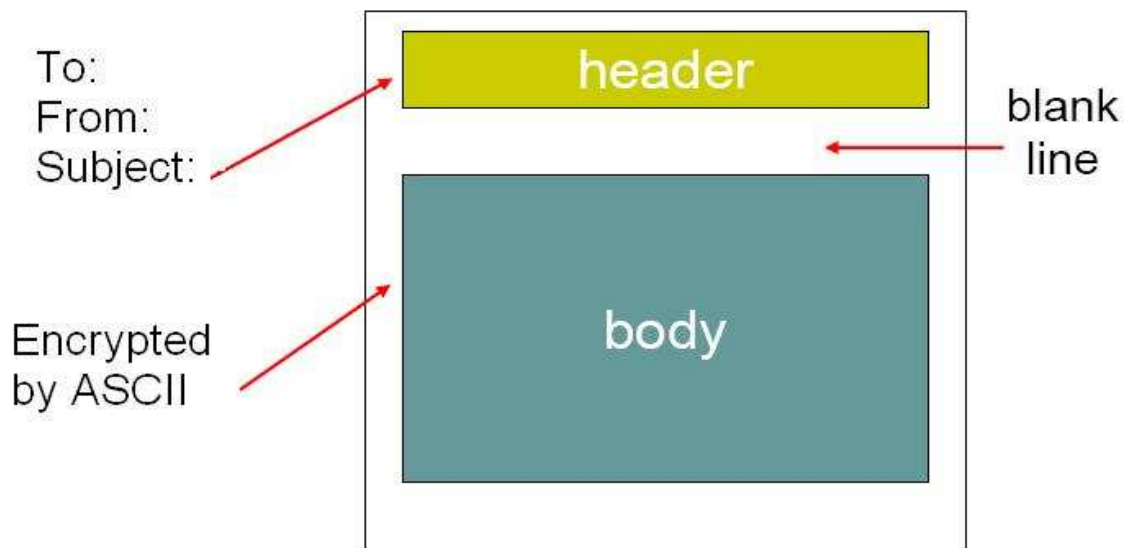
2.1.1.Email

Electronic mail, often abbreviated as email or e-mail, is a method of exchanging digital messages, designed primarily for human use.



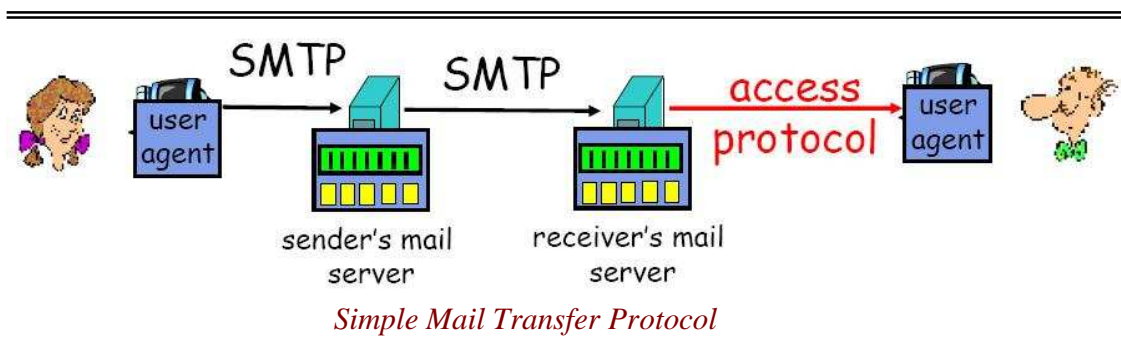
Email

An electronic mail message consists of two components, the message header, and the message body, which is the email's content. The message header contains control information, including, minimally, an originator's email address and one or more recipient addresses. Usually additional information is added, such as a subject header field.



E-mail formatting

Email Services and SMTP/POP Protocols



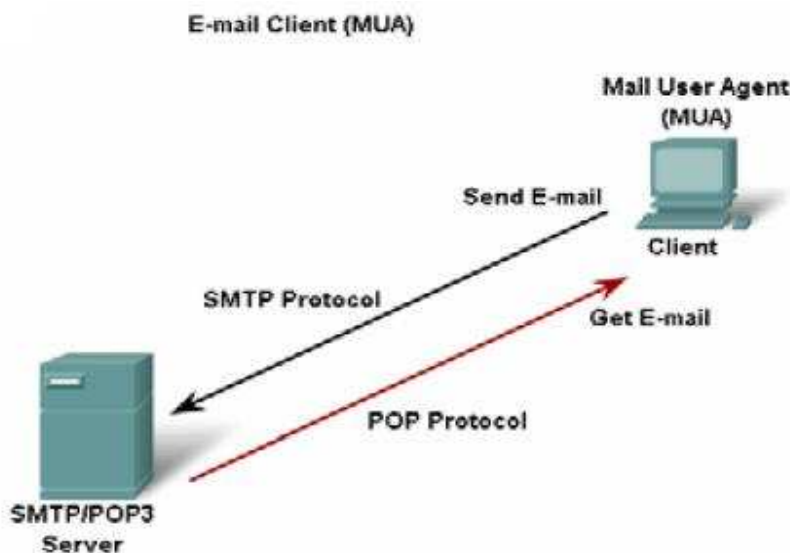
Email, the most popular network service and run on a computer or other end device, e-mail requires several applications and services

POP/SMTP define client/server processes.

Mail User Agent (MUA): allows messages to be sent and places received messages into the client's mailbox, both of which are distinct processes. MUA include: POP and SMTP.

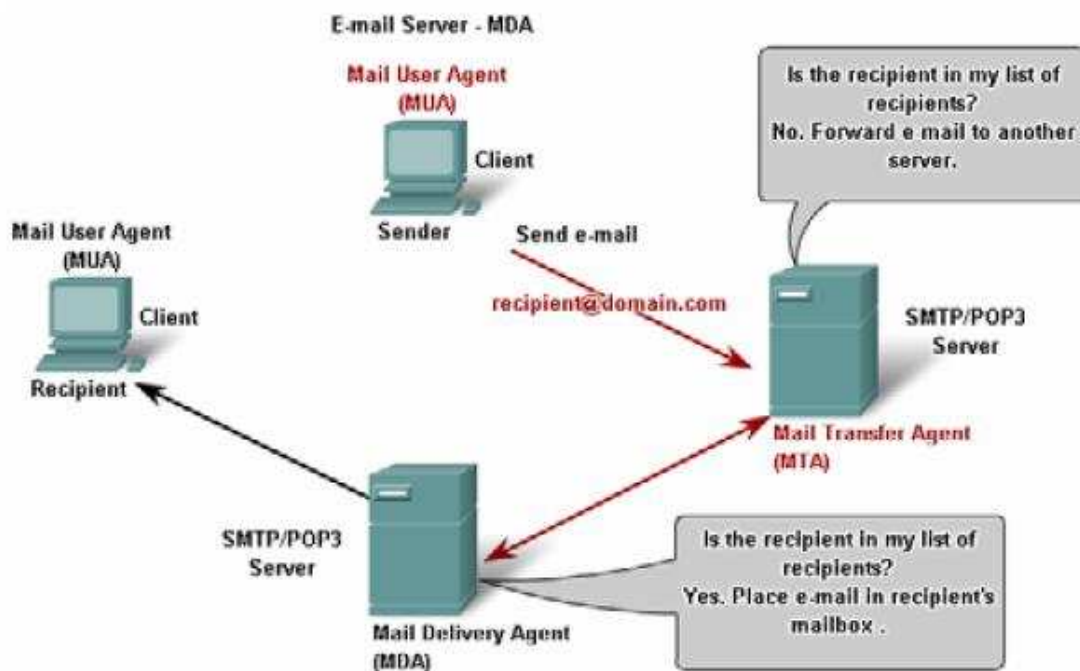
POP: used to receive e-mail messages from an e-mail server

SMTP: used to send e-mail from either a client or a server uses message formats and command strings.



E-mail Client (MUA)

E-mail Server Processes – MTA (Mail Transfer Agent) and MDA (Mail Delivery Agent)



E-mail Server - MDA

The MDA accepts a piece of e-mail from a MTA and performs the actual delivery.

The MDA receives all the inbound mail from the MTA and places it into the appropriate users' mailboxes.

The MDA can also resolve final delivery issues, such as virus scanning, spam filtering, and return-receipt handling.

POP and POP3 are inbound mail delivery protocols and are typical client/server protocols. They deliver e-mail from the e-mail server to the client (MUA).

The MDA listens for when a client connects to a server. Once a connection is established, the server can deliver the e-mail to the client.

Some of the commands specified in the SMTP protocol are: HELO, EHLO, MAIL FROM, RCPT TO, DATA.

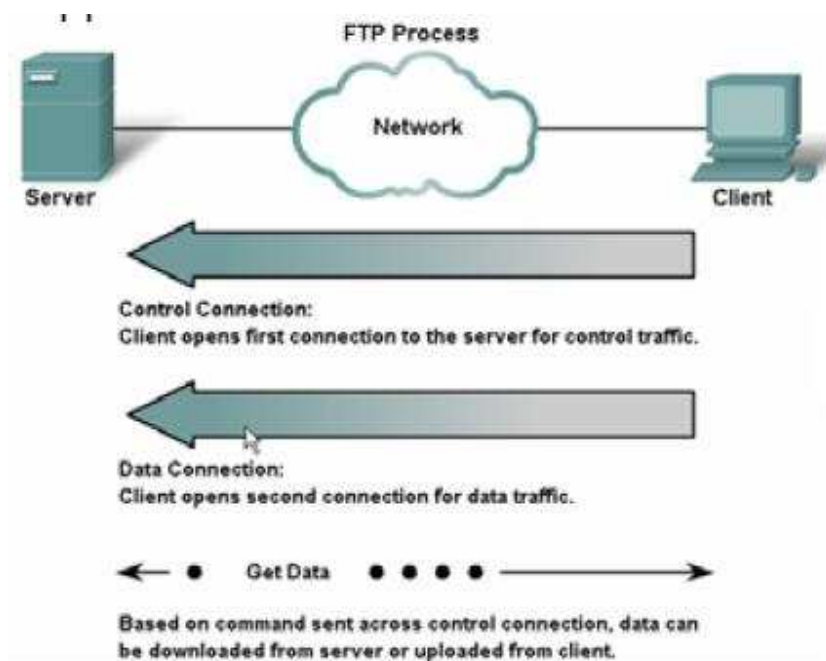
2.1.2. FTP

The File Transfer Protocol (FTP) is another commonly used Application layer protocol.

FTP was developed to allow for file transfers between a client and a server. A FTP client is an application that runs on a computer that is used to push and pull files from a server running the FTP daemon (FTPD).

The client establishes the first connection to the server on TCP port 21. The client establishes the second connection to the server over TCP port 20.

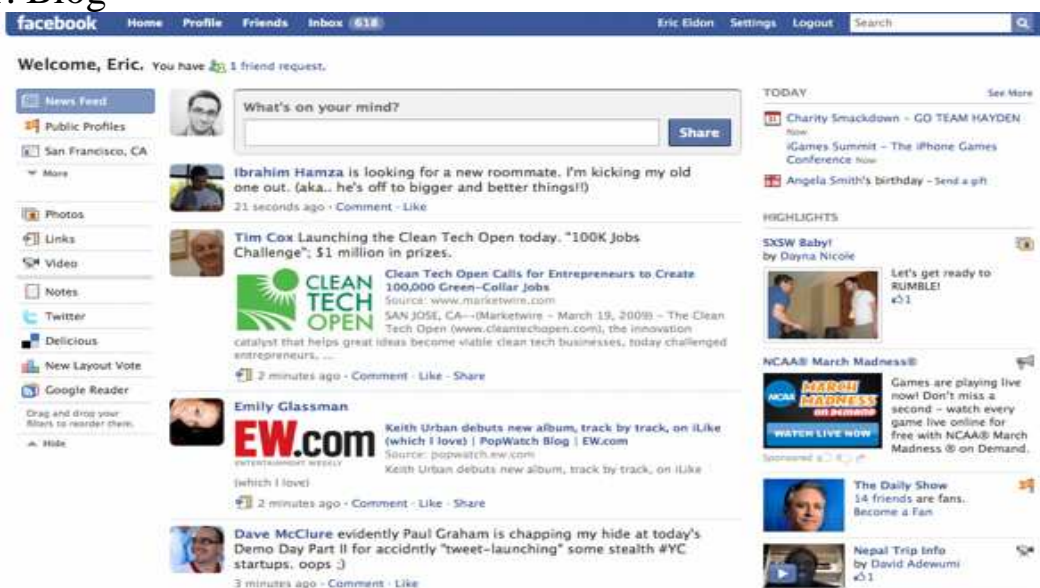
The file transfer can happen in either direction



FTP Process

3. Typical application

3.1. Blog



A blog (a contraction of the term weblog) is a type of website, usually maintained by an individual with regular entries of commentary, descriptions of events, or other material such as graphics or video. Entries are commonly displayed in reverse-chronological order. "Blog" can also be used as a verb, meaning to maintain or add content to a blog.

There are many different types of blogs, differing not only in the type of content, but also in the way that content is delivered or written.

- The personal blog, an ongoing diary or commentary by an individual, is the traditional, most common blog. Personal bloggers usually take pride in their blog posts, even if their blog is never read by anyone but them. Blogs often become more than a way to just communicate; they become a way to reflect on life or works of art.

- Corporate blogs

A blog can be private, as in most cases, or it can be for business purposes. Blogs, either used internally to enhance the communication and culture in a corporation or externally for marketing, branding or public relations purposes are called corporate blogs.

3.2. SNS

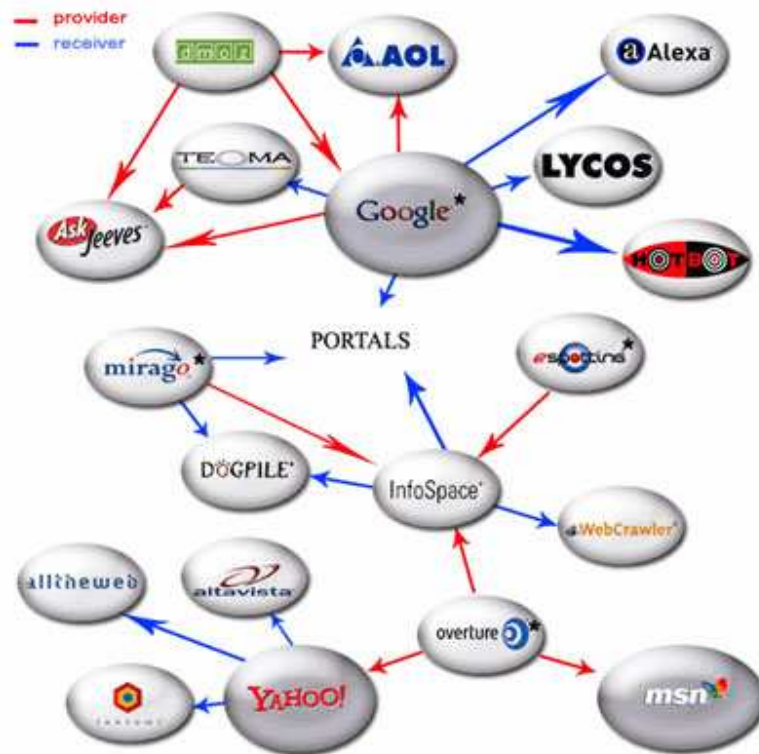
A social network service focuses on building online communities of people who share interests and/or activities, or who are interested in exploring the interests and activities of others. Most social network services are web based and provide a variety of ways for users to interact, such as e-mail and instant messaging services.

Social networking has encouraged new ways to communicate and share information. Social networking websites are being used regularly by millions of people.

3.3 Search engine

A Web search engine is a tool designed to search for information on the World Wide Web. The search results are usually presented in a list and are commonly called hits.

The information may consist of web pages, images, information and other types of files. Some search engines also mine data available in databases or open directories. Unlike Web directories, which are maintained by human editors, search engines operate algorithmically or are a mixture of algorithmic and human input.



Search engines diagram.

3.4. Google map

Google Maps (for a time named Google Local) is a web mapping service application and technology provided by Google, free (for non-commercial use), that powers many map-based services, including the Google Maps website, Google Ride Finder, Google Transit, and maps embedded on third-party websites via the Google Maps API. It offers street maps, a route planner for traveling by foot, bicycle, car, or public transport and an urban business locator for numerous countries around the world. It also can help find the location of businesses.

Google Maps uses the Mercator projection, so it cannot show areas around the poles. A related product is Google Earth, a stand-alone program for Microsoft Windows, Mac OS X, Linux, SymbianOS, and iPhone OS which offers more globe-viewing features, including showing polar areas.



Google Map Maker Released

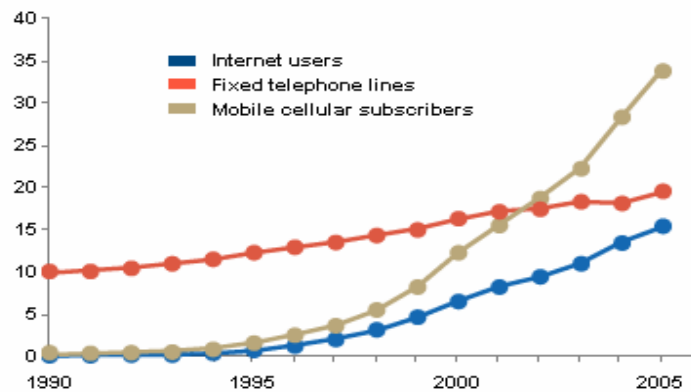


4. Social Impact

Connectivity is increasing, with the number of Internet users and telephone subscribers expanding worldwide. The most rapid growth has taken place in the mobile sector, which has been critical to improving communications in regions with few fixed telephone lines. The number of mobile subscribers worldwide rose from 11 million in 1990 to 2.2 billion in 2005, compared to growth in fixed lines of 520 million to 1.2 billion over the same period.

Access to information and communication technologies grows fastest in the mobile sector

Number of telephone subscriptions and Internet connections per 100 population, 1990-2005 (Percentage)



4.1. Advantages

The Internet provides opportunities galore, and can be used for a variety of things. Some of the things that you can do via the Internet are:

- **E-mail:** E-mail is an online correspondence system. With e-mail you can send and receive instant electronic messages, which works like writing letters. Your messages are delivered instantly to people anywhere in the world, unlike traditional mail that takes a lot of time.
- **Access Information:** The Internet is a virtual treasure trove of information. Any kind of information on any topic under the sun is available on the Internet. The 'search engines' on the Internet can help you to find data on any subject that you need.
- **Shopping:** Along with getting information on the Internet, you can also shop online. There are many online stores and sites that can be used to look for products as well as buy them using your credit card. You do not need to leave your house and can do all your shopping from the convenience of your home.
- **Online Chat:** There are many 'chat rooms' on the web that can be accessed to meet new people, make new friends, as well as to stay in touch with old friends.
- **Downloading Software:** This is one of the most happening and fun things to do via the Internet. You can download innumerable, games, music, videos, movies, and a host of other entertainment software from the Internet, most of which are free.

4.2. Disadvantages

There are certain cons and dangers relating to the use of Internet that can be summarized as:

- **Personal Information:** If you use the Internet, your personal information such as your name, address, etc. can be accessed by other people. If you use a credit card to shop online, then your credit card information can also be 'stolen' which could be akin to giving someone a blank check.
- **Pornography:** This is a very serious issue concerning the Internet, especially when it comes to young children. There are thousands of pornographic sites on the Internet that can be easily found and can be a detriment to letting children use the Internet.
- **Spamming:** This refers to sending unsolicited e-mails in bulk, which serve no purpose and unnecessarily clog up the entire system.
- **Virus threat:** Virus is nothing but a program which disrupts the normal functioning of your computer systems. Computers attached to internet are more prone to virus attacks and they can end up into crashing your whole hard disk, causing you considerable headache.

Chapter 2.

1. Server-client model

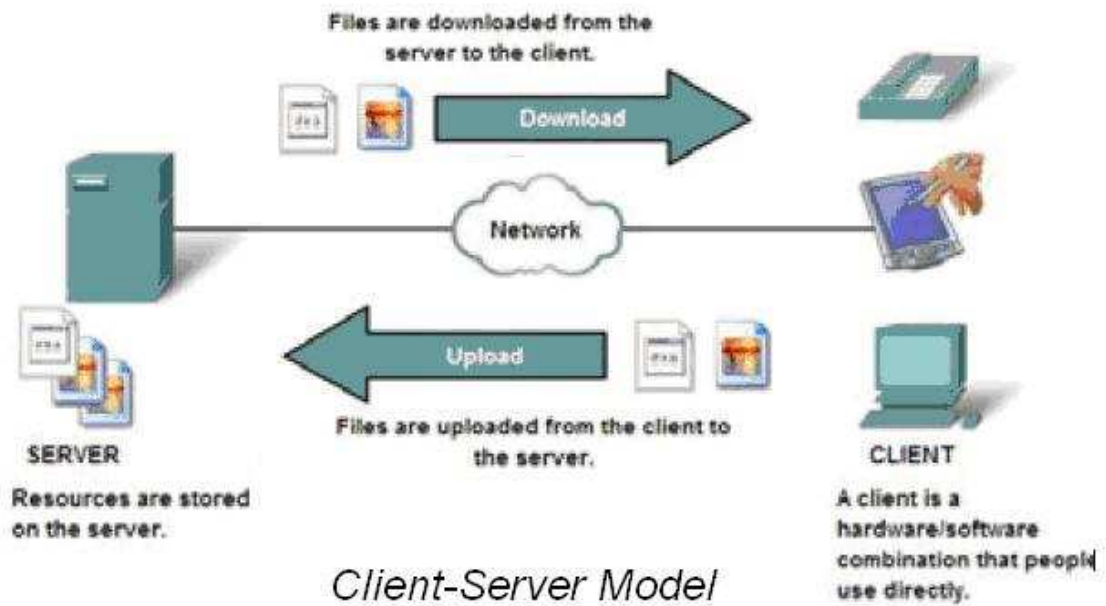
1.1. Introduce client-server model of web system

The client-server software architecture model distinguishes client systems from server systems, which communicate over a computer network.

In the client/server model, the device requesting the information is called a client and the device responding to the request is called a server. Client and server processes are considered to be in the Application layer.

The client begins the exchange by requesting data from the server, which responds by sending one or more streams of data to the client. Application layer protocols describe the design of the requests and responds between clients and servers. In addition to the actual data transfer, this exchange can require control information, such as user authentication and the identification of a data file to be transferred.

Data transfer from a client to a server is referred to as an *upload*, and data from a server to a client is a *download*. Data flow can be equal in both directions or can even be greater in the direction going from the client to the server.

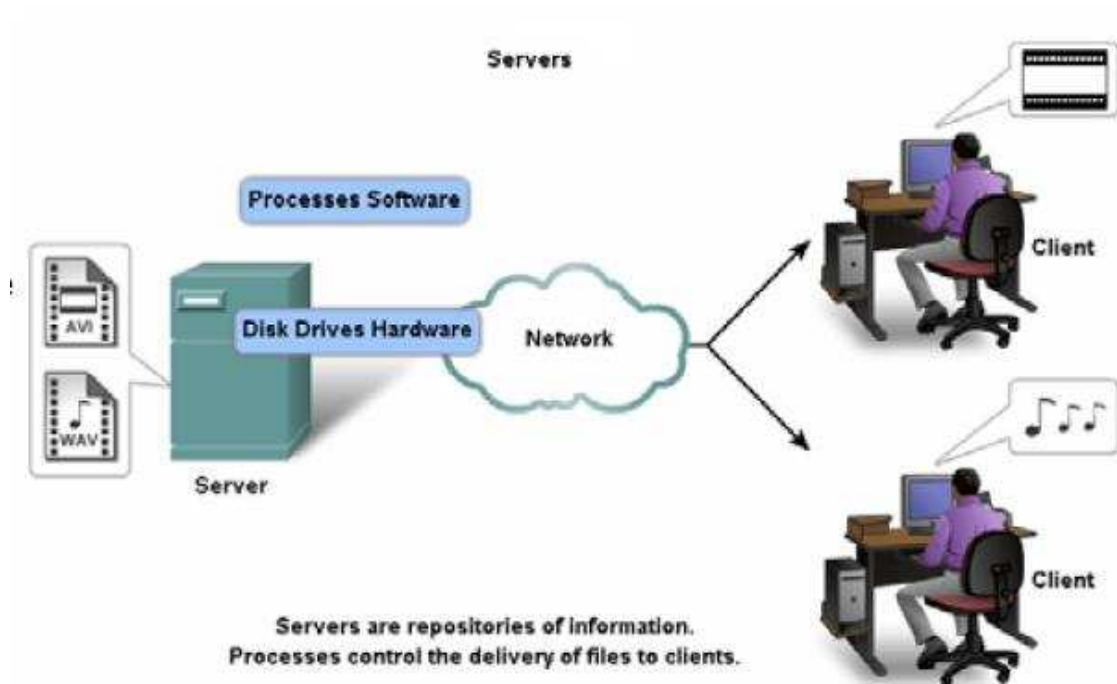


In a general networking context, any device that responds to requests from client application is functioning as a server.

A server is usually a computer that contains information to be shared with many client systems.

Different types of server applications can have different requirements for client access.

Some servers may require authentication of user account information to verify whether the user has permission to access the requested data or to use a particular operation. Such servers rely on a central list of user accounts and the authorizations, or permissions (both for data access and operations), granted to each user.



1.2. Example of some Web server

IIS (Internet Information Service)

Internet Information Services (IIS) - is a set of Internet-based services for servers created by Microsoft for use with Microsoft Windows.

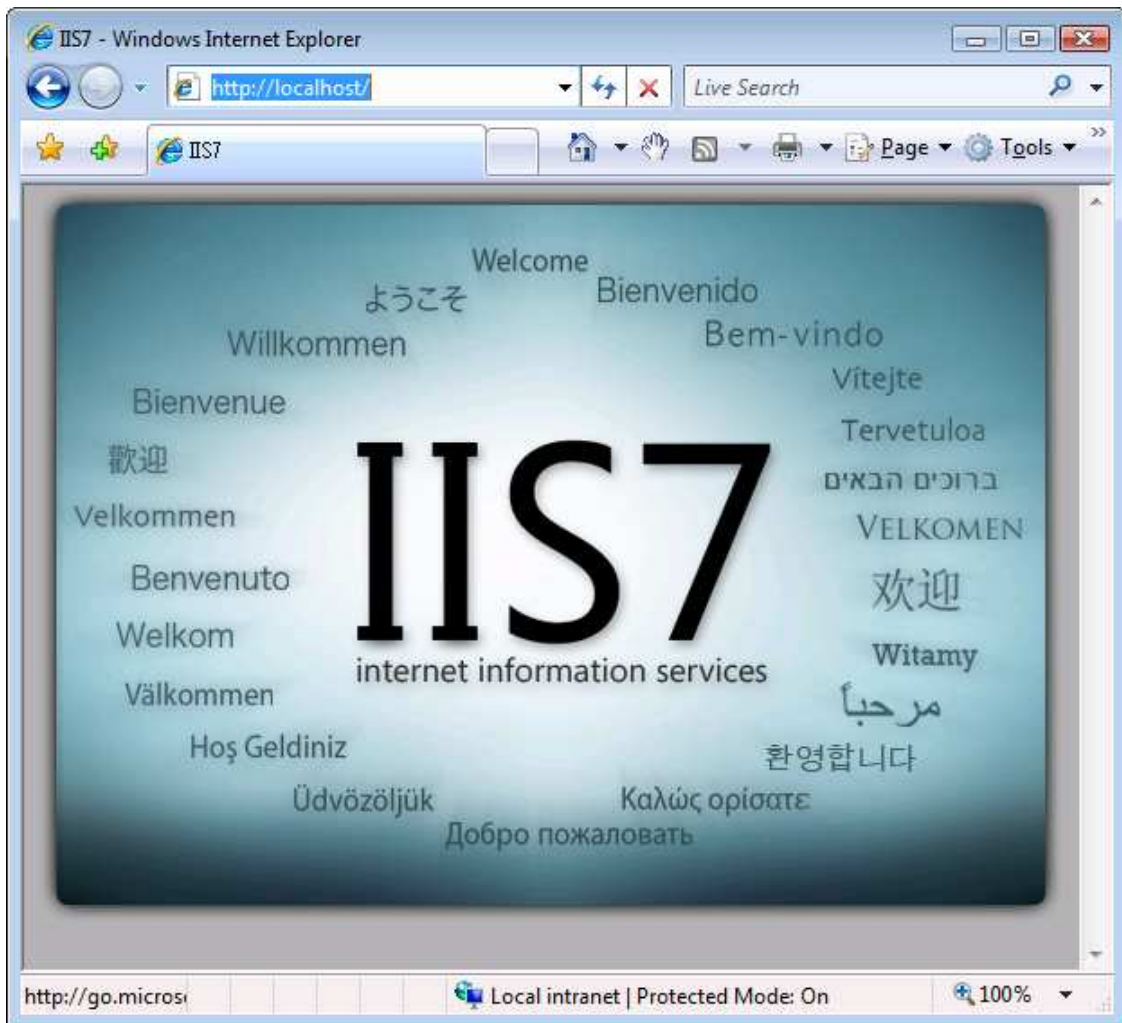
It is the world's second most popular web server in terms of overall websites behind the industry leader Apache HTTP Server.

As of April 2009[update] it served 29.27% of all websites according to Netcraft. The services provided currently include FTP, FTPS, SMTP, NNTP, and HTTP/HTTPS.

Versions

- IIS 1.0, Windows NT 3.51 available as a free add-on
 - IIS 2.0, Windows NT 4.0
 - IIS 3.0, Windows NT 4.0 Service Pack 3
 - IIS 4.0, Windows NT 4.0 Option Pack
 - IIS 5.0, Windows 2000
 - IIS 5.1, Windows XP Professional, Windows XP Media Center Edition
 - IIS 6.0, Windows Server 2003 and Windows XP Professional x64 Edition
 - IIS 7.0, Windows Server 2008 and Windows Vista (Business, Enterprise, Ultimate Editions)
 - IIS 7.5, Windows Server 2008 R2 (Beta) and Windows 7 (Beta)
-

Once IIS is installed on your machine you can view your home page in a web browser by typing `http://localhost` into the address bar of your web browser. Since you have not yet created a web site you should see the default IIS page.



Apache HTTP Server

The Apache HTTP Server, commonly referred to as Apache, is a web server notable for playing a key role in the initial growth of the World Wide Web and in 2009 became the first web server to surpass the 100 million web site milestone. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently known as Sun Java System Web Server), and has since evolved to rival other Unix-based web servers in terms of functionality and performance. The majority of all web servers using Apache are Linux web servers.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating systems, including Unix, GNU, FreeBSD, Linux, Solaris, NoVell NetWare, Mac OS X, Microsoft Windows, OS/2, TPF, and

eComStation. Released under the Apache License, Apache is characterized as free software and open source software.

Since April 1996 Apache has been the most popular HTTP server on the World Wide Web. As of March 2009[update] Apache served over 46% of all websites and over 66% of the million busiest.

Features

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes.

Popular compression methods on Apache include the external extension module, `mod_gzip`, implemented to help with reduction of the size (weight) of web pages served over HTTP. Apache logs can be analyzed through a web browser using free scripts such as AWStats/W3Perl or Visitors.

Virtual hosting allows one Apache installation to serve many different actual websites.

Use

Apache is primarily used to serve both static content and dynamic Web pages on the World Wide Web.

Apache is the web server component of the popular LAMP web server application stack, alongside Linux, MySQL, and the PHP/Perl/Python (and now also Ruby) programming languages.

Apache is redistributed as part of various proprietary software packages including the Oracle Database or the IBM WebSphere application server.

Apache is used for many other tasks where content needs to be made available in a secure and reliable way.

Programmers developing web applications often use a locally installed version of Apache in order to preview and test code as it is being developed.

License

The software license under which software from the Apache Foundation is distributed is a distinctive part of the Apache HTTP Server's history and presence in the open source software community. The Apache License allows for the distribution of both open and closed source derivations of the source code.

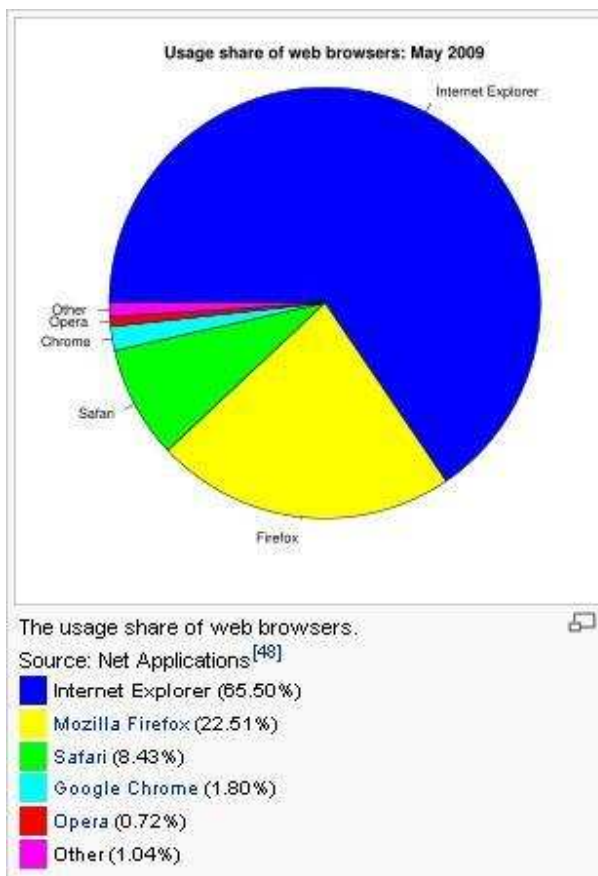
The Free Software Foundation does not consider the Apache License to be compatible with version 2 of the GNU General Public License (GPL) in that software licensed under the Apache License cannot be integrated with software that is distributed under the GPL.

1.3. Example of some Web client:

IE

Windows Internet Explorer (formerly Microsoft Internet Explorer; abbreviated to MSIE or, more commonly, IE) is a series of graphical web browsers developed by Microsoft and included as part of the Microsoft Windows line of operating systems starting in 1995.

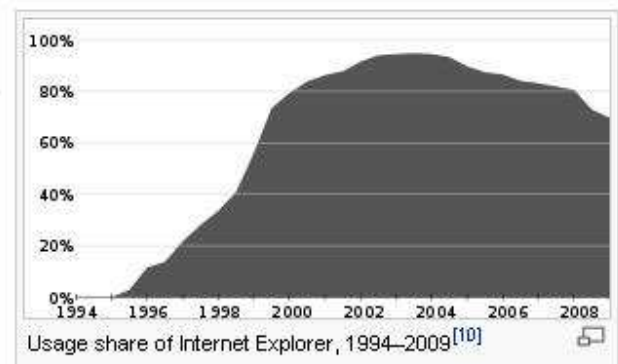
It has been the most widely used, and still most widely used web browser since 1999, attaining a peak of about 95% usage share during 2002 and 2003 with IE5 and IE6. It has been alleged that Internet Explorer's share would have been lower if it was not bundled with Windows. That percentage share has since declined in the face of renewed competition from other web browsers – Mozilla Firefox most of all. Microsoft spent over \$100 million a year on IE in the late 1990s, with over 1,000 people working on it by 1999.



IE market share overview

— May 2009^[9]

Internet Explorer 5	0.04%
Internet Explorer 5.5	0.03%
Internet Explorer 6	16.94%
Internet Explorer 7	40.83%
Internet Explorer 8	6.85%
All variants	66.10%



The usage share of web browsers

Firefox

Mozilla Firefox is a free and open source web browser descended from the Mozilla Application Suite and managed by Mozilla Corporation.

Firefox had 22.51% of the recorded usage share of web browsers as of May 2009, making it the second most popular browser in terms of current use worldwide, after Internet Explorer.

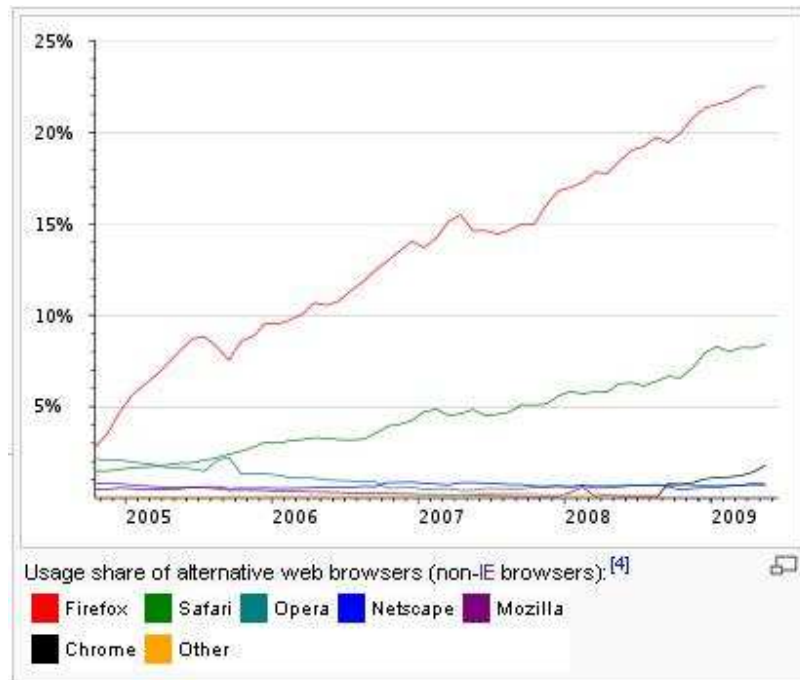
Firefox 1.0	0.05%
Firefox 1.5	0.15%
Firefox 2.0	1.63%
Firefox 3.0	20.43%
Firefox 3.5	0.19%
All versions ^[4]	22.51%

To display web pages, Firefox uses the Gecko layout engine, which implements some current web standards in addition to a few features which are intended to anticipate likely additions to the standards.

Firefox features include tabbed browsing, a spell checker, incremental find, live bookmarking, a download manager, and an integrated search system that uses the user's desired search engine (Google by default in most localizations). Functions can be added through add-ons, created by third-party developers, of which there is a wide selection, a feature that has attracted many of Firefox's users.

Firefox runs on various versions of Linux, Mac OS X, Microsoft Windows, and many other Unix-like operating systems. Its current stable release is version 3.0.11, released on June 11, 2009.

Firefox's source code is free software, released under a tri-license GNU GPL/GNU LGPL/MPL. Official versions are distributed under the terms of a proprietary EULA.



Usage share of alternative web browsers (non IE browsers)

Safari

Safari is a web browser developed by Apple Inc.

First released as a public beta on January 7, 2003 on the company's Mac OS X operating system, it became Apple's default browser beginning with Mac OS X v10.3 "Panther". Apple has also made Safari the native browser for the iPhone OS. A version of Safari for the Microsoft Windows operating system was first released on June 11, 2007, and supports both Windows XP and Windows Vista. The current stable release of the browser is 4.0 for both Macintosh and Windows. Safari has a 8.43% market share as of May 2009.

Features

Safari offers most features common to modern web browsers such as:

- Tabbed browsing
 - Bookmark Management
 - A resizable web-search box in the toolbar which uses Google on the Mac and either Google or Yahoo! on Windows
 - Pop-up ad blocking
 - History and bookmark search
 - Text search
 - Spell-checking
-
-

-
- Expandable text boxes
 - Automatic filling in of web forms
 - Built-in password management via Keychain
 - Subscribing to and reading web feeds
 - Quartz-style font-smoothing
 - The Web Inspector, a DOM Inspector-like utility that lets users and developers browse the Document Object Model of a web page
 - Support for CSS 3 web fonts
 - Support for CSS animation
 - Bookmark integration with Address Book
 - ICC colour profile support
 - Inline PDF viewing
 - Integration with iPhoto photo management
 - Mail integration
 - Ability to save parts of web pages as web clips for viewing on the Apple Dashboard.

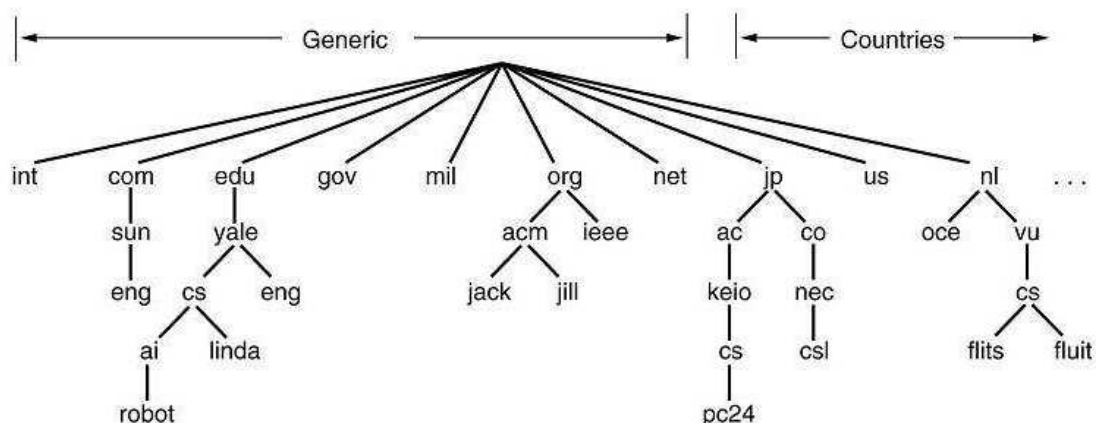
2. DNS and URL

2.1. DNS (Domain Name System)

The Domain Name System (DNS) is a hierarchical naming system for computers, services, or any resource participating in the Internet

A domain naming system was developed in order to associate the contents of the site with the address of that site. The Domain Name System (DNS) is a system used on the Internet for translating names of domains and their publicly advertised network nodes into IP addresses.

The Domain Name System makes it possible to assign domain names to groups of Internet users in a meaningful way, independent of each user's physical location.



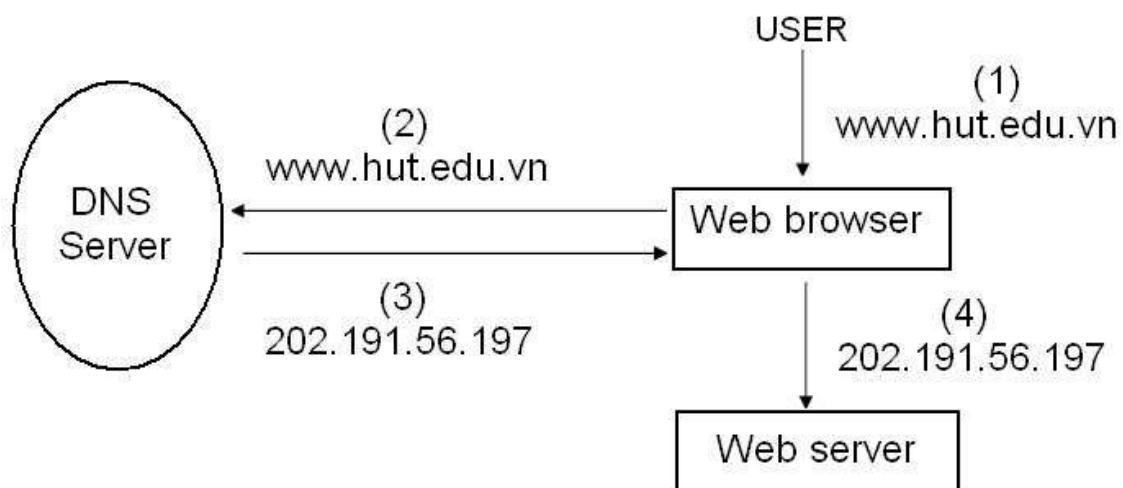
DNS Server Hierarchy

The Domain Name System (DNS) was created for domain name to address resolution for these networks. DNS uses a distributed set of servers to resolve the names associated with these numbered addresses.

The DNS protocol defines an automated service that matches resource names with the required numeric network address.

DNS protocol communications use a single format called a message.

DNS is used for all types of client queries and server responses, error messages, and the transfer of resource record information between servers.



The role of DNS

The DNS server stores different types of resource records used to resolve names. These records contain the name, address, and type of record.

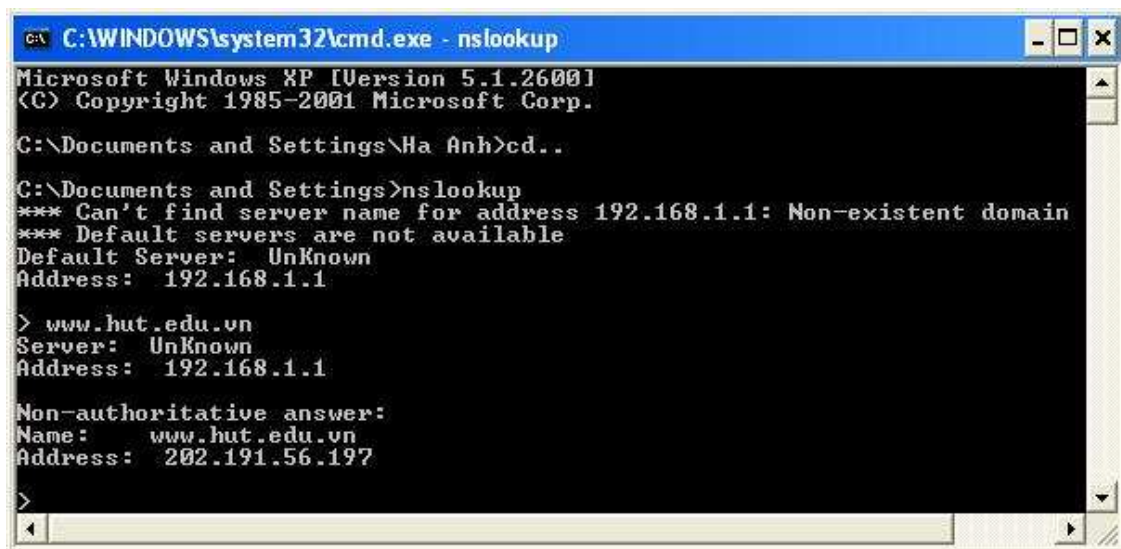
DNS uses the same message format for:

- all types of client queries and server responses
- error messages
- the transfer of resource record information between servers

Header	
Question	The question for the name server
Answer	Resource Records answering the question
Authority	Resource Records pointing toward an authority
Additional	Resource Records holding additional information

When configuring a network device, we generally provide one or more DNS Server addresses that the DNS client can use for name resolution.

Computer operating systems also have a utility called nslookup that allows the user to manually query the name servers to resolve a given host name. This utility can also be used to troubleshoot name resolution issues and to verify the current status of the name servers.



```
C:\WINDOWS\system32\cmd.exe - nslookup
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Ha Anh>cd..

C:\Documents and Settings>nslookup
*** Can't find server name for address 192.168.1.1: Non-existent domain
*** Default servers are not available
Default Server: UnKnown
Address: 192.168.1.1

> www.hut.edu.vn
Server: UnKnown
Address: 192.168.1.1

Non-authoritative answer:
Name: www.hut.edu.vn
Address: 202.191.56.197

>
```

2.2. URL

In computing, a Uniform Resource Locator (URL) is a type of Uniform Resource Identifier (URI) that specifies where an identified resource is available and the mechanism for retrieving it. In popular usage and in many technical

documents and verbal discussions it is often incorrectly used as a synonym for URI. In popular language, a URL is also referred to as a Web address.

Every resource on the Web has a unique address. For example, 207.46.130.149 is the address of the Microsoft web site. Now, remembering those numbers can be quite difficult and confusing. Hence, the Uniform Resource Locators (URLs) are used. A URL is a string that supplies the Internet address of a Web site or resource on the World Wide Web.

The typical format is, www.nameofsite.typeofsite.countrycode

For example, 216.239.33.101 can be represented by the URL, www.google.com

The URL also identifies the protocol by which the site or resource is accessed. The most common URL type is “http”, which gives the Internet address of a Web page. Some other URL types are “gopher”, which gives the Internet address of a Gopher directory, and “ftp”, which gives the network location of an FTP resource.

The URL can also refer to a location within a resource. For example, you can create a link to a topic within the same document. In that case, a fragment identifier is used at the end of the URL.

The format is, protocol: name of site/main document#fragment identifier

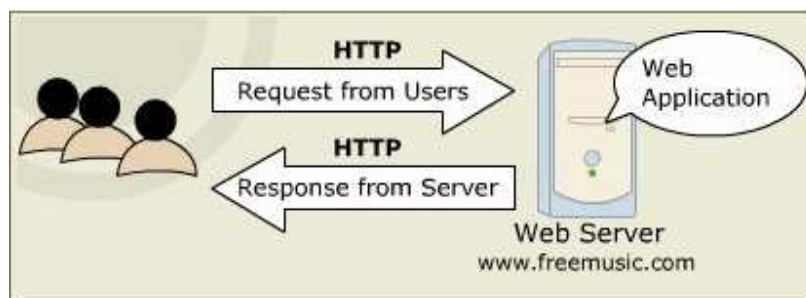
There are two types of URLs:

➤ Absolute URL- is the full Internet address of a page or file, including the protocol, network location, and optional path and file name. For example, <http://www.microsoft.com> is an absolute URL.

➤ Relative URL- is a URL with one or more of its parts missing. Browsers take the missing information from the page containing the URL. For example, if the protocol is missing, the browser uses the protocol of the current page.

3.HTTP Request and Response

3.1.HTTP Protocol Overview



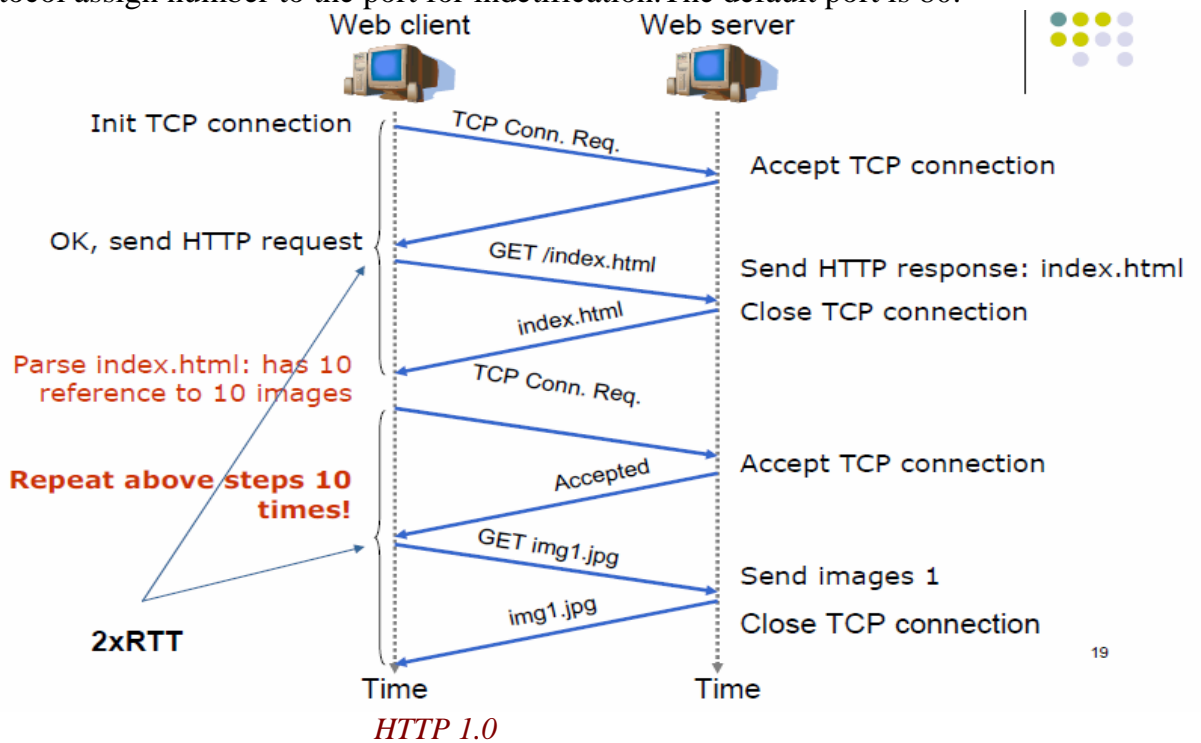
The request and response in Web Application are sent using HTTP.

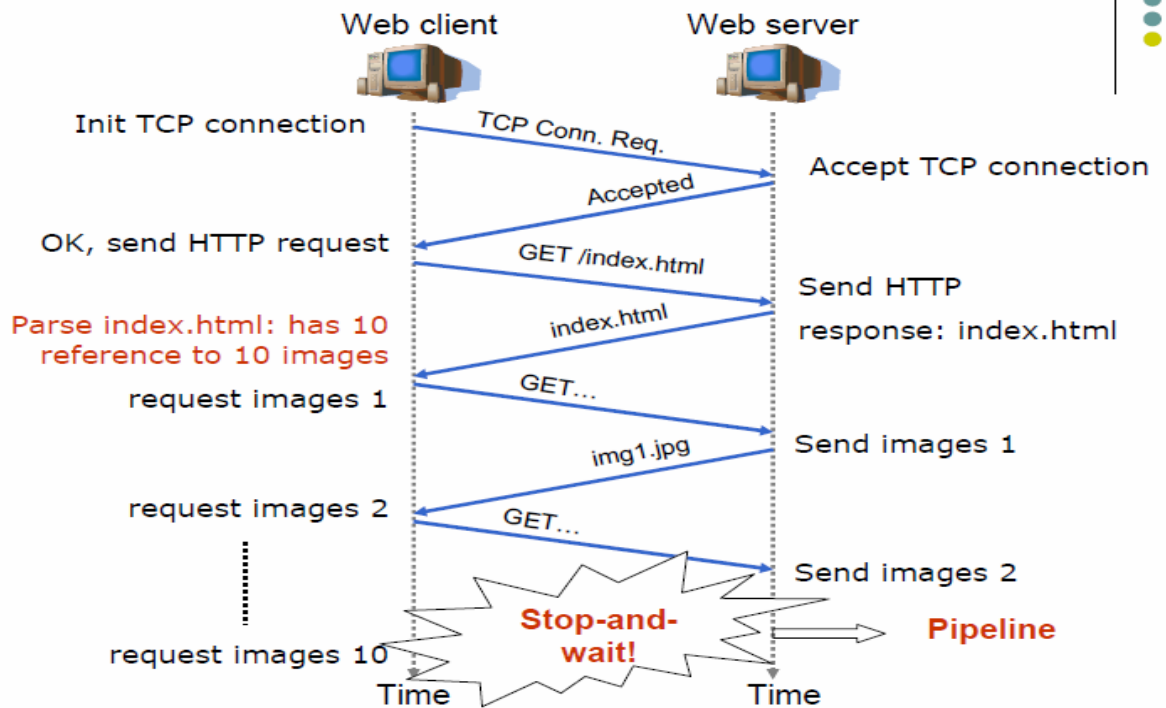
An HTTP Client, such as Web Browsers (Firefox, IE, Opera), open the connection and send the request message to an HTTP server asking resource. The server, in turn, return the response message with the request resource. Once the resource is delivered, Server closes the connection.

HTTP doesn't store any connection information and hence it's referred to a stateless protocol.

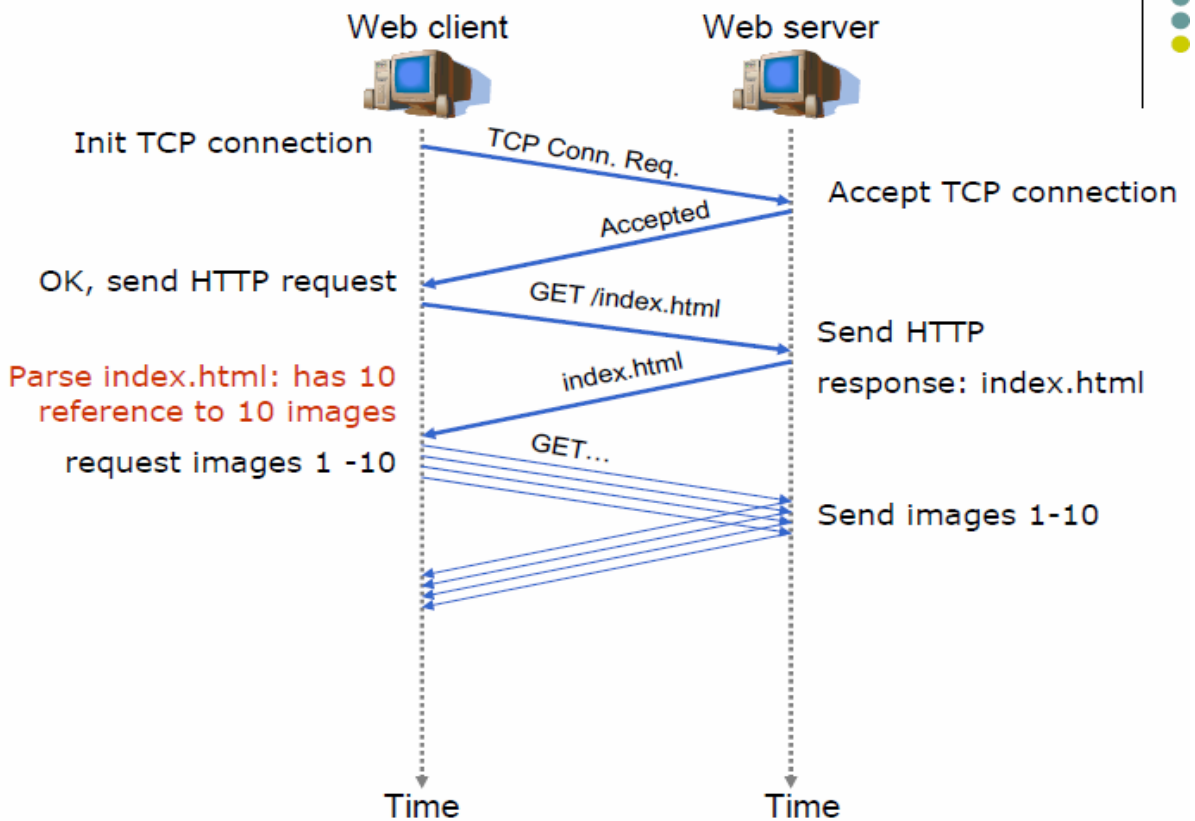
In HTTP Connection last for only one transaction. A transaction consists of several request-response pairs.

Port is a channel used by protocols for sending and receiving data. Networking protocol assign number to the port for identification. The default port is 80.



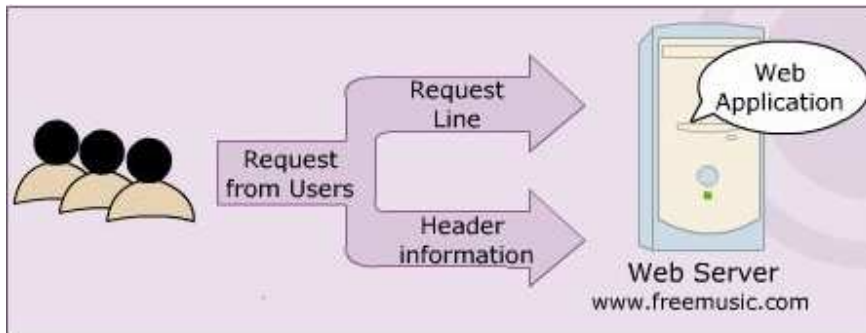


HTTP 1.1



HTTP 1.1 with pipeline.

3.2. Request Message Structure



The request message sent by the client is a stream of text. It consists of the following elements.

Request line

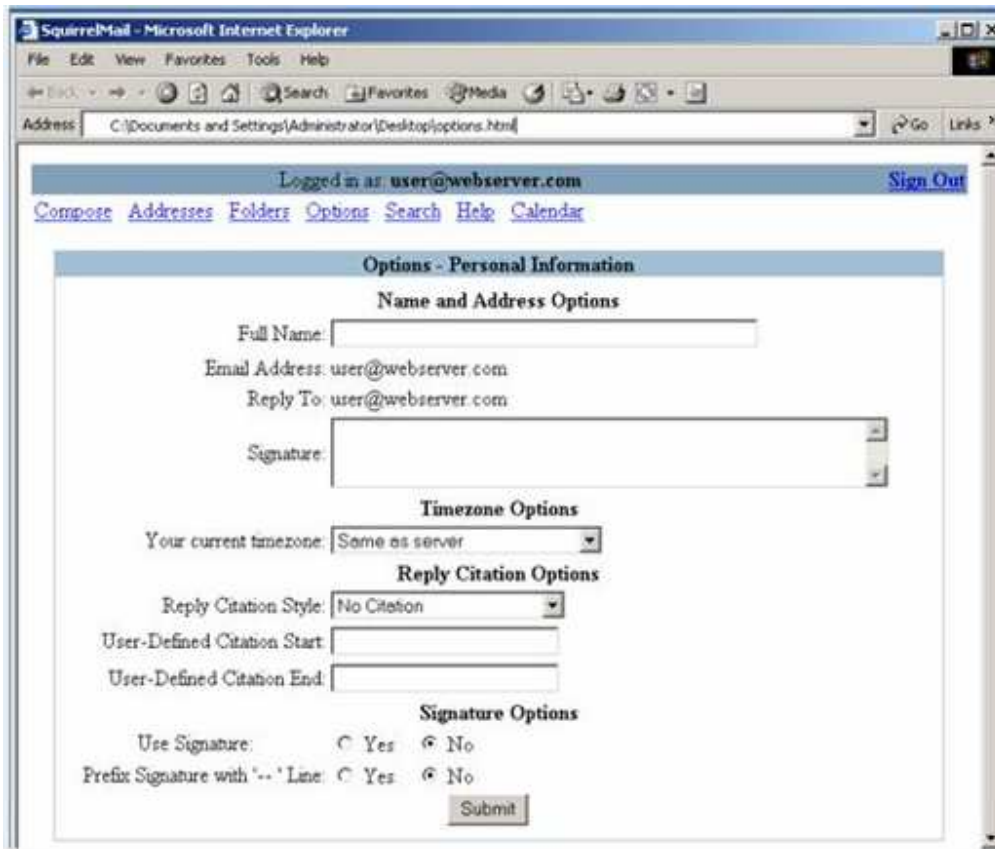
This component sent method, resource information, such as the type and name, and the HTTP version of the request message.

Header Information

This component return user- agent along with the Accept Header. The user-agent element header indicate the browser used by client. The accept header element provides information on what media types the client can accept. After the header, client send blank line indicating the end of request message.

Request Parameter

Web application allow user to enter information using forms and send to the server for processing.



For example, in the banking site, the user enter the username and password. These are sent to web server through request parameters and after validating the data received with the database, the account information is send as the response. The server treat the data entered in each field as request parameter. The server can extract the values of each request parameter in the request to processing.

Request Method

Get



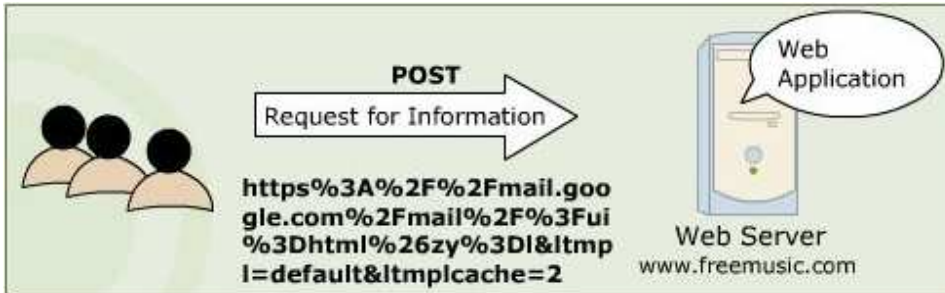
The get method used for getting information such as a document, a char, a database query in the form of plain text. Search Engine such as www.google.com use the get method to retrieve results for a search strings entered by user. Following is a sample query string constructed by google.

<http://www.google.co.uk/search?hl=en&q=java&meta=>

Where java is search keyword.

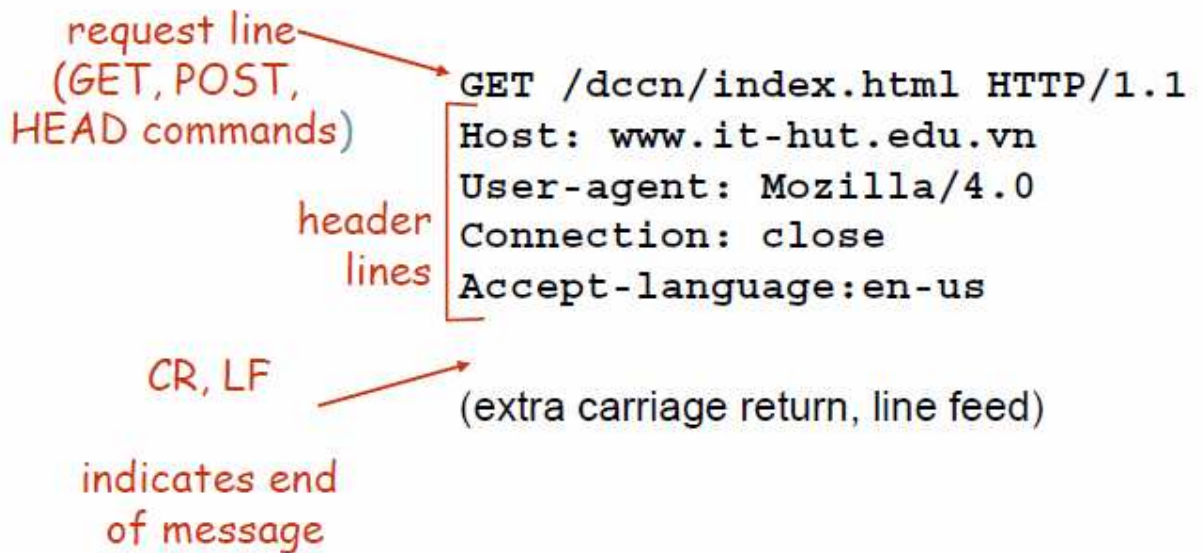
The length of query string is restricted from 240 to 255 character depending on the server. Hence this method can't be used to send data from bulky forms.

POST

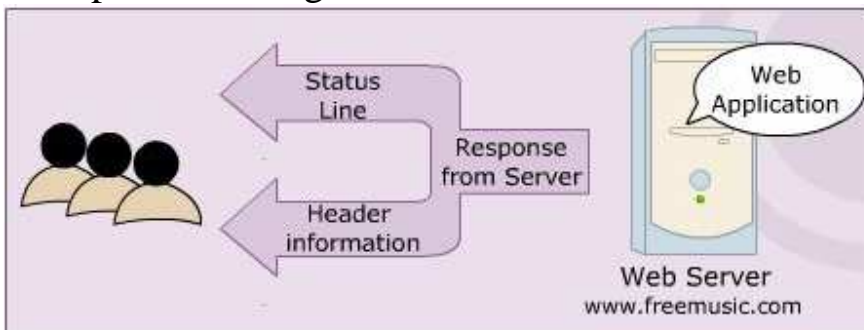


Post method is used when sending information, such as credit card number or information to be saved in the database. Data send using post is in encrypted format and not visible to the client and there is not limit on amount of data being sent. However, pages request using page method can't be bookmarked or emailed. This because Post method is used to transmit sensitive data password.

Example



3.3. Response Message Structure



The Server processes the request sent by a client and generates a response message consists of the following elements.

Status line

This element indicates the status of request message.

Header Information

The header information contains information such as server, last modify date, content length and content type. The server header specifies the software being used on the server. Last-modified header indicates the last modified date of the requested file. The content length specifies the size of file in bytes. Content type header specifies the type of document.

Example

status line
(protocol
status code
status phrase) → HTTP/1.1 200 OK

header lines → Connection close
Date: Tue, 16 Mar 2008 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 15 Mar 2008
Content-Length: 8990
Content-Type: text/html

data, e.g.,
requested
HTML file → data data data data data ...

The response code

<p>200 OK</p> <ul style="list-style-type: none">● request succeeded, requested object later in this message <p>301 Moved Permanently</p> <ul style="list-style-type: none">● requested object moved, new location specified later in this message (Location:) <p>400 Bad Request</p> <ul style="list-style-type: none">● request message not understood by server <p>404 Not Found</p> <ul style="list-style-type: none">● requested document not found on this server <p>505 HTTP Version Not Supported</p>

4.Character Encoding

4.1.Unicode

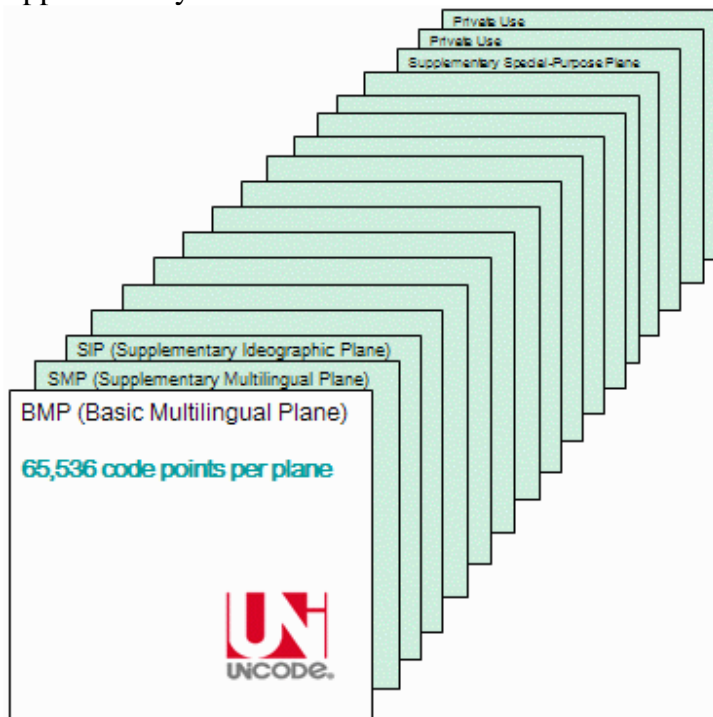
4.1.1.Over view

Unicode is a universal character set, ie. a standard that defines, in one place, all the characters needed for writing the majority of living languages in use on computers. It aims to be, and to a large extent already is, a superset of all other character sets that have been encoded.

The following table lists some of the growing number of scripts that are covered by Unicode:

Arabic	Greek	Khmer	Runic
Armenian	Gujurati	Lao	Sinhala
Bengali	Gurmukhi	Latin	Syriac
Canadian Syllabics	Han	Malayalam	Tamil
Cherokee	Hangul	Mongolian	Telugu
Cyrillic	Hebrew	Myanmar	Thaana
Devanagari	Hiragana	Ogham	Thai
Ethiopic	Kannada	Oriya	Tibetan
Georgian	Katakana	Panjabi	etc...

The first 65,536 code point positions in the Unicode character set are said to constitute the Basic Multilingual Plane (BMP). The BMP includes most of the more common characters in use. Around a million further code point positions are available in the Unicode character set. Characters in this latter range are referred to as supplementary characters.



4.1.2.Character sets, coded character sets, and encodings

It is important to clearly distinguish between the concepts character set and character encoding.

A character set or repertoire comprises the set of characters one might use for a particular purpose – be it those required to support Western European languages in computers, or those a Chinese child will learn at school in the third grade (nothing to do with computers).

A coded character set is a set of characters for which a unique number has been assigned to each character. Units of a coded character set are known as code points. For example, the code point for the letter à in the Unicode coded character set is 225 in decimal, or E1 in hexadecimal notation. (Note that hexadecimal notation is commonly used for identifying such characters, and will be used here.)

The character encoding reflects the way these abstract characters are mapped to bytes for manipulation in a computer.

4.1.3. One character set, multiple encodings

Many character encoding standards, such as ISO 8859 series, use a single byte for a given character and the encoding is straightforwardly related to the scalar position of the characters in the coded character set. For example, the letter A in the ISO 8859-1 coded character set is in the 65th character position (starting from zero), and is encoded for representation in the computer using a byte with the value of 65. For ISO 8859-1 this never changes.

For Unicode, however, things are not so straightforward. Although the code point for the letter à in the Unicode coded character set is always 225 (in decimal), it may be represented in the computer by two bytes. In other words there isn't a trivial, one-to-one mapping between the coded character set value and the encoded value for this character.

In addition, in Unicode there are a number of ways of encoding the same character. For example, the letter à can be represented by two bytes in one encoding and four bytes in another. The encoding forms that can be used with Unicode are called UTF-8, UTF-16, and UTF-32.

UTF-8 uses 1 byte to represent characters in the ASCII set, two bytes for characters in several more alphabetic blocks, and three bytes for the rest of the BMP. Supplementary characters use 4 bytes.

UTF-16 uses 2 bytes for any character in the BMP, and 4 bytes for supplementary characters.

UTF-32 uses 4 bytes for all characters.

In the following chart, the first line of numbers represents the position of the characters in the Unicode coded character set. The other lines show the byte values used to represent that character in a particular character encoding.

	A	à	□	𐀀
Code point	U+0041	U+05D0	U+597D	U+233B4
UTF-8	41	D7 90	E5 A5 BD	F0 A3 8E B4
UTF-16	00 41	05 D0	59 7D	D8 4C DF

				B4
UTF-32	00 00 00 41	00 00 05 D0	00 00 59 7D	00 02 33 B4

4.1.4.Character escapes

A character escape is an alternative way of representing a character, without actually using the code point of the character.

For example, there is no way of representing the Hebrew character א in your document if you are using an ISO 8859-1 encoding (which covers Western European languages). One way to indicate that you want to include that character is to use the XHTML escape א. Because the document character set is Unicode, the user agent should recognize that this represents a Hebrew aleph character.

4.1.5.Document character set

For XML and HTML (from version 4.0 onwards) the document character set is defined to be the Universal Character Set (UCS) as defined by both ISO/IEC 10646 and Unicode standards. (For simplicity and in line with common practice, we will refer to the UCS here simply as Unicode.)

This means that the logical model describing how XML and HTML are processed is described in terms of the set of characters defined by Unicode.

Note that this does not mean that all HTML and XML documents have to be encoded as Unicode! It does mean, however, that documents can only contain characters defined by Unicode. Any encoding can be used for your document as long as it is properly declared and a subset of the Unicode repertoire.

4.2.Different Encodings

4.2.1.UTF-8

The bits of a Unicode character are distributed into the lower bit positions inside the UTF-8 bytes, with the lowest bit going into the last bit of the last byte. In this table, x represent the lowest 8 bits of the Unicode value, y represent the next higher 8 bits, and z represent the bits higher than that:

Unicode	Byte1	Byte2	Byte3	Byte4	example
U+0000-U+007F	0xxxxxxx				'\$' U+0024 → 00100100 → 0x24
U+0080-U+07FF	110yyyyx	10xxxxxx			'ç' U+00A2 → 11000010, 10100010 → 0xC2, 0xA2
U+0800-U+FFFF	1110yyyy	10yyyyyx	10xxxxxx		'€' U+20AC → 11100010, 10000010, 10101100 → 0xE2, 0x82, 0xAC
U+10000-U+10FFFF	11110zzz	10zzyyyy	10yyyyyx	10xxxxxx	U+10ABCD → 11110100, 10001010, 10101111, 10001101 → 0xF4, 0x8A, 0xAF, 0x8D

So the first 128 characters (US-ASCII) need one byte. The next 1920 characters need two bytes to encode. This includes Latin letters with diacritics and characters from Greek, Cyrillic, Coptic, Armenian, Hebrew, Arabic, Syriac and Tāna alphabets. Three bytes are needed for the rest of the Basic Multilingual Plane (which contains virtually all characters in common use). Four bytes are needed for characters in the other planes of Unicode, which are rarely used in practice.

By continuing the pattern given above it is possible to deal with much larger numbers. The original specification allowed for sequences of up to six bytes covering numbers up to 31 bits (the original limit of the Universal Character Set). However, UTF-8 was restricted by RFC 3629 to use only the area covered by the formal Unicode definition, U+0000 to U+10FFFF, in November 2003.

With these restrictions, bytes in a UTF-8 sequence have the following meanings. The ones marked in red can never appear in a legal UTF-8 sequence. The ones in green are represented in a single byte. The ones in white must only appear as the first byte in a multi-byte sequence, and the ones in orange can only appear as the second or later byte in a multi-byte sequence:

binary	hex	decimal	notes
00000000-01111111	00-7F	0-127	US-ASCII (single byte)
10000000-10111111	80-BF	128-191	Second, third, or fourth byte of a multi-byte sequence
11000000-11000001	C0-C1	192-193	Overlong encoding: start of a 2-byte sequence, but code point <= 127
11000010-11011111	C2-DF	194-223	Start of 2-byte sequence
11100000-11101111	E0-EF	224-239	Start of 3-byte sequence
11110000-11110100	F0-F4	240-244	Start of 4-byte sequence
11110101-11110111	F5-F7	245-247	Restricted by RFC 3629: start of 4-byte sequence for codepoint above 10FFFF
11111000-11111011	F8-FB	248-251	Restricted by RFC 3629: start of 5-byte sequence
11111100-11111101	FC-FD	252-253	Restricted by RFC 3629: start of 6-byte sequence
11111110-11111111	FE-FF	254-255	Invalid: not defined by original UTF-8 specification

4.2.2.EUC

Extended Unix Code (EUC) is a multibyte character encoding system used primarily for Japanese, Korean, and simplified Chinese.

The structure of EUC is based on the ISO-2022 standard, which specifies a way to represent character sets containing a maximum of 94 characters, or 8836 (942) characters, or 830584 (943) characters, as sequences of 7-bit codes. Only ISO-2022 compliant character sets can have EUC forms. Up to four coded character sets (referred to as G0, G1, G2, and G3 or as code sets 0, 1, 2, and 3) can be represented with the EUC scheme. G0 is almost always an ISO-646 compliant coded character set (e.g. US-ASCII/KS X 1003/ISO 646:KR in EUC-KR and US-ASCII/the lower half of JIS X 0201 in EUC-JP) that is invoked on GL (i.e. with the most significant bit cleared).

To get the EUC form of an ISO-2022 character, the most significant bit of each 7-bit byte of the original ISO 2022 codes is set (by adding 128 to each of these original 7-bit codes); this allows software to easily distinguish whether a particular byte in a character string belongs to the ISO-646 code or the ISO-2022 (EUC) code.

The most commonly-used EUC codes are variable-width encodings with a character belonging to G0 (ISO-646 compliant coded character set) taking one byte and a character belonging to G1 (taken by a 94x94 coded character set) represented in two bytes. The EUC-CN form of GB2312 and EUC-KR are examples of such two-byte EUC codes. EUC-JP includes characters represented by up to three bytes whereas a single character in EUC-TW can take up to four bytes.

EUC-CN is the usual way to use the GB2312 standard for simplified Chinese characters. Unlike the case of Japanese, the ISO-2022 form of GB2312 is not normally used, though a variant form called HZ was sometimes used on USENET.

EUC-CN can also be used to encode the Unicode-based GB18030 character encoding, which includes traditional characters, although GB18030 is more frequently used without EUC encoding, since GB18030 is already a Unicode encoding. However, GB18030 encoded in EUC-CN is a variable-width encoding, because GB18030 contains more than 8836 (94x94) characters.

EUC-JP is a variable-width encoding used to represent the elements of three Japanese character set standards, namely JIS X 0208, JIS X 0212, and JIS X 0201.

A character from the lower half of JIS-X-0201 (ASCII, code set 0) is represented by one byte, in the range 0x21 – 0x7E.

A character from the upper half of JIS-X-0201 (half-width kana, code set 2) is represented by two bytes, the first being 0x8E, the second in the range 0xA1 – 0xDF.

A character from JIS-X-0208 (code set 1) is represented by two bytes, both in the range 0xA1 – 0xFE.

A character from JIS-X-0212 (code set 3) is represented by three bytes, the first being 0x8F, the following two in the range 0xA1 – 0xFE.

This encoding scheme allows the easy mixing of 7-bit ASCII and 8-bit Japanese without the need for the escape characters employed by ISO-2022-JP, which is based on the same character set standards.

In Japan, the EUC-JP encoding is heavily used by Unix or Unix-like operating systems (except for HP-UX), while Shift_JIS or its extensions (Windows code page 932 and MacJapanese) are used on other platforms. Therefore, whether Japanese web sites use EUC-JP or Shift_JIS often depends on what OS the author uses.

EUC-JISX0213 is similar to but different from EUC-JP in that two planes of JIS X 0213 take place of JIS-X-0208 and JIS-X-0212. There is a similar relationship between Shift_JIS and Shift-JISX0213.

EUC-KR is a variable-width encoding to represent Korean text using two coded character sets, KS X 1001 (formerly KS C 5601) and KS X 1003 (formerly KS C 5636)/ISO 646:KR/US-ASCII. KS X 2901 (formerly KS C 5861) stipulates the encoding and RFC 1557 dubbed it as EUC-KR. A character drawn from KS X 1001 (G1, code set 1) is encoded as two bytes in GR (0xA1-0xFE) and a character from KS X 1003/US-ASCII (G0, code set 0) takes one byte in GL (0x21-0x7E).

It is the most widely used legacy character encoding in Korea on all three major platforms (Unix-like OS, Windows and Mac), but its use has been very slowly decreasing as UTF-8 gains popularity, especially on Linux and Mac OS X. It is usually referred to as Wansung (한글) in South Korea. The default Korean codepage for Windows is a proprietary, but upward compatible extension of EUC-KR referred to as Unified Hangeul Code (한글 유니코드, Tonghab Wansunghyung). Mac Korean used in classic Mac OS is also compatible with EUC-KR.

EUC-TW is a variable-width encoding that supports US-ASCII and 16 planes of CNS 11643, each of which is 94x94. It is a rarely used encoding for traditional Chinese characters as used on Taiwan. Big5 is much more common. A character in US-ASCII (G0, code set 0) is encoded as a single byte in GL(0x21-0x7E) and a character in CNS 11643 plane 1 (code set 1) is encoded as two bytes in GR (0xA1-0xFE). A character in plane 1 through 16 of CNS 11643 (code set 2) is encoded as four bytes with the first byte always being 0x8E(Single Shift 2) and the second byte indicating the plane (the plane number is obtained by subtracting 0xA0 from the second byte). The third and fourth bytes are in GR (0xA1-0xFE). Note that the plane 1 of CNS 11643 is encoded twice as code set 1 and a part of code set 2.

5. Media Type

An Internet media type, originally called a MIME type after MIME and sometimes a Content-type after the name of a header in several protocols whose value is such a type, is a two-part identifier for file formats on the Internet. The identifiers were originally defined in RFC 2046 for use in e-mail sent through SMTP, but their use has expanded to other protocols such as HTTP and SIP.

A media type is composed of at least two parts: a type, a subtype, and one or more optional parameters. For example, subtypes of text type have an optional

charset parameter that can be included to indicate the character encoding, and subtypes of multipart type often define a boundary between parts.

Types or subtypes that begin with x- are nonstandard – they cannot be registered with IANA. Subtypes that begin with vnd. are vendor-specific; subtypes in the personal or vanity tree begin with prs.

5.1.Type text: Human-readable text and source code

✓Text/css: Cascading Style Sheets. Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation (that is, the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document, including SVG and XUL.

✓Text/csv: Comma-separated values. A Comma separated values (CSV) file is used for the digital storage of data structured in a table of lists form, where each associated item (member) in a group is in association with others also separated by the commas of its set. Each line in the CSV file corresponds to a row in the table. Within a line, fields are separated by commas, each field belonging to one table column. CSV files are often used for moving tabular data between two different computer programs, for example between a database program and a spreadsheet program.

✓Text/html: HTML. HTML, an acronym for HyperText Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document—by denoting certain text as links, headings, paragraphs, lists, etc.—and to supplement that text with interactive forms, embedded images, and other objects. HTML is written in the form of "tags" that are surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code (such as JavaScript) that can affect the behavior of Web browsers and other HTML processors.

✓Text/javascript (Obsolete): JavaScript. JavaScript is a scripting language used to enable programmatic access to objects within other applications. It is primarily used in the form of client-side JavaScript for the development of dynamic websites. JavaScript is a dialect of the ECMAScript standard and is characterized as a dynamic, weakly typed, prototype-based language with first-class functions. JavaScript was influenced by many languages and was designed to look like Java, but to be easier for non-programmers to work with.

✓Text/plain: Textual data, Defined in RFC 2046 and RFC 3676.

✓Text/xml: Extensible Markup Language; Defined in RFC 3023.

5.2.Type video: Video

✓Video/mpeg: MPEG-1 video with multiplexed audio. MPEG-1 is a standard for lossy compression of video and audio. It is designed to compress VHS-quality raw digital video and CD audio down to 1.5 Mbit/s (26:1 and 6:1 compression

ratios respectively) without excessive quality loss, making Video CDs, digital cable/satellite TV and digital audio broadcasting (DAB) possible.

✓Video/mp4: MP4 video. MPEG-4 Part 14, formally ISO/IEC 14496-14:2003, is a multimedia container format standard specified as a part of MPEG-4. It is most commonly used to store digital audio and digital video streams, especially those defined by MPEG, but can also be used to store other data such as subtitles and still images. Like most modern container formats, MPEG-4 Part 14 allows streaming over the Internet. A separate hint track is used to include streaming information in the file. The official filename extension for MPEG-4 Part 14 files is .mp4, thus the container format is often referred to simply as MP4.

✓Video/quicktime: QuickTime video. QuickTime is a multimedia framework developed by Apple Inc., capable of handling various formats of digital video, media clips, sound, text, animation, music, and interactive panoramic images. It is available for Mac OS (Mac OS 9, 8, 7, etc.), Mac OS X and Microsoft Windows operating systems.

✓Video/x-ms-wmv: Windows Media Video. Windows Media Video (WMV) is a compressed video file format for several proprietary codecs developed by Microsoft. The original codec, known as WMV, was originally designed for Internet streaming applications, as a competitor to RealVideo. The other codecs, such as WMV Screen and WMV Image, cater for specialized content.

5.3. Type image

✓Image/gif: GIF image. The Graphics Interchange Format (GIF) is a bitmap image format that was introduced by CompuServe in 1987 and has since come into widespread usage on the World Wide Web due to its wide support and portability.

✓Image/jpeg: JPEG JFIF image. JPEG is a commonly used method of compression for photographic images. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. JPEG typically achieves 10:1 compression with little perceptible loss in image quality.

✓Image/png: Portable Network Graphics. Portable Network Graphics (PNG) is a bitmapped image format that employs lossless data compression. PNG was created to improve upon and replace GIF (Graphics Interchange Format) as an image-file format not requiring a patent license.

✓Image/svg+xml: SVG vector image. Scalable Vector Graphics (SVG) is a family of specifications of XML-based file format for describing two-dimensional vector graphics, both static and dynamic (interactive or animated).

✓Image/tiff: Tag Image File Format. Tagged Image File Format (abbreviated TIFF) is a file format for storing images, including photographs and line art. It is as of 2009 under the control of Adobe Systems. Originally created by the company Aldus for use with what was then called "desktop publishing", the TIFF format is widely supported by image-manipulation applications, by publishing and page layout applications, by scanning, faxing, word processing, optical character recognition and other applications.

✓Image/vnd.microsoft.icon: ICO image. The ICO file format is an image file format used for icons in Microsoft Windows. The CUR file format for cursors is almost identical, as the only differences are the identification byte and a specification of a hotspot in the header.

5.4. Type audio: Audio

✓Audio/mpeg: MP3 or other MPEG audio MPEG-1. Audio Layer 3, more commonly referred to as MP3, is a patented digital audio encoding format using a form of lossy data compression. It is a common audio format for consumer audio storage, as well as a de facto standard of digital audio compression for the transfer and playback of music on digital audio players. MP3 is an audio-specific format that was designed by the Moving Picture Experts Group. The group was formed by several teams of engineers at Fraunhofer IIS in Erlangen, Germany, AT&T-Bell Labs in Murray Hill, NJ, USA, Thomson-Brandt, and CCETT as well as others. It was approved as an ISO/IEC standard in 1991.

✓Audio/x-ms-wma: Windows Media Audio. Windows Media Audio (WMA) is an audio data compression technology developed by Microsoft. The name can be used to refer to its audio file format or its audio codecs. It is a proprietary technology that forms part of the Windows Media framework. WMA consists of four distinct codecs. The original WMA codec, known simply as WMA, was conceived as a competitor to the popular MP3 and RealAudio codecs.

✓Audio/vnd.rn-realaudio: RealAudio. RealAudio is a proprietary audio format developed by RealNetworks. It uses a variety of audio codecs, ranging from low-bitrate formats that can be used over dialup modems, to high-fidelity formats for music. It can also be used as a streaming audio format, that is played at the same time as it is downloaded. In the past, many internet radio stations used RealAudio to stream their programming over the internet in real time. In recent years, however, the format has become less common and has given way to more popular audio formats. It is used heavily by the BBC websites.

✓Audio/x-wav: WAV audio. WAV (or WAVE), short for Waveform audio format, also known as Audio for Windows[2], is a Microsoft and IBM audio file format standard for storing an audio bitstream on PCs. It is an application of the RIFF bitstream format method for storing data in “chunks”, and thus also close to the IFF and the AIFF format used on Amiga and Macintosh computers, respectively. It is the main format used on Windows systems for raw and typically uncompressed audio. The usual bitstream encoding is the Pulse Code Modulation (PCM) format.

5.5. How multimedia data are included in HTTP message

Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of e-mail to support:

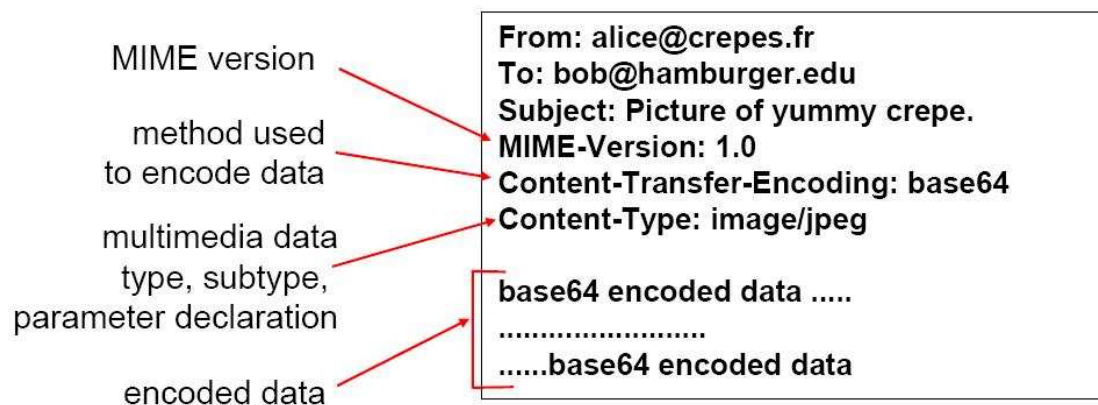
- ✎Text in character sets other than ASCII
 - ✎Non-text attachments
 - ✎Message bodies with multiple parts
 - ✎Header information in non-ASCII character sets
-
-

MIME's use, however, has grown beyond describing the content of e-mail to describing content type in general, including for the web.

Virtually all human-written Internet e-mail and a fairly large proportion of automated e-mail is transmitted via SMTP in MIME format. Internet e-mail is so closely associated with the SMTP and MIME standards that it is sometimes called SMTP/MIME e-mail.

The content types defined by MIME standards are also of importance outside of e-mail, such as in communication protocols like HTTP for the World Wide Web. HTTP requires that data be transmitted in the context of e-mail-like messages, even though the data may not actually be e-mail.

MIME is specified in six linked RFC memoranda: RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 and RFC 2049, which together define the specifications.



Chapter 3.

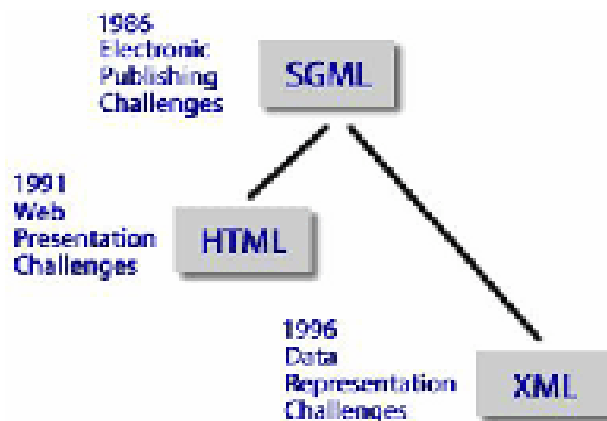
1. Markup and rendering

1.1. Markup language

A markup language is a set of annotations to text that describe how it is to be structured, laid out, or formatted. Markup languages might be manuscript form (often marks among or alongside text describing required formatting or binding), or they might be markup codes used in computer typesetting and word-processing systems.

The former are also commonly used to describe the required layout of papers, articles, standards, or books. The latter tend more to be used to instantiate a particular document and nowadays are not generally used directly by authors. The code used to specify the formatting are called tags.

Various markup languages used are SGML , HTML , XML and the latest being WML.



By markup language we mean a set of markup conventions used together for encoding texts. A markup language must specify what markup is allowed, what markup is required, how markup is to be distinguished from text, and what the markup means.

Historically, markup was used to refer to:

The process of marking manuscript copy for typesetting with directions for use of type fonts and sizes, spacing, indentation, etc. (from the Chicago Manual of Style, the bible of most publishers.)

Electronic Markup originally referred to the internal, sometimes invisible codes in documents which described the formatting.

In WYSIWYG systems, the system inserts the codes. In early WYSIWYG systems such as Wordstar, the markup is visible on the screen.

Markup can be classified as one of two types:

✍ Procedural Markup which is concerned with the appearance of text - its font, spacing etc.

✍ Descriptive or Declarative Markup which is concerned with the structure or function of the tagged item.

Markup Languages permit you to use your information for applications beyond traditional publishing. For example:

- ◆ World Wide Web home pages
- ◆ Information databases
- ◆ Diagnostic/expert systems
- ◆ Electronic mail
- ◆ Hypermedia and hypertext documents
- ◆ Database publishing
- ◆ CD-ROM publishing
- ◆ Interactive Electronic Technical Manuals (IETMs)
- ◆ Electronic review

1.2. Visualise an HTML document using an web browser

The HTML document is displayed in a browser. What is a browser? A browser is an application that is installed on the client machine. The browser reads the HTML source code and displays the page as instructed.

A browser is used to view Web pages and navigate through it. The earliest known browser was Mosaic developed by the National Center for Supercomputing Applications (NCSA). Today there are many browser available for browsing the Internet. Netscape's Navigator and Microsoft's Internet Explorer are two popular browsers in use. For the user, a browser is easy to use because it provides a point-and-click graphical interface.

To create the source document, an HTML editor is required. There are several editors in use today: Microsoft FrontPage is a comprehensive tool that can be used to create, design, and edit Web pages. We can also add text, images, tables and other HTML elements to the page. In addition, a form can be created through FrontPage. Once we create the interface, the FrontPage Editor creates the required HTML code. We can also use Notepad to create the HTML document. In order to view the document in a browser you have to save the document with a .htm/.html extension.

HTML commands are called Tags. Tags are used to control the content and appearance of the HTML document. The “opening tag” is a “ < >” pair of brackets. This indicates the beginning of the HTML command. The “closing tag” is represented as “</ >” to indicate the end of the HTML command.

2. Syntax of HTML

2.1. Structure of an HTML Document

An HTML document has three basic sections:

➤ **The HTML Section:** Every HTML document must begin with an opening HTML tag and end with a closing HTML tag.

```
<HTML> ... </HTML>
```

The HTML tag tells the browser that the content between these two tags is an HTML document.

➤ **The header Section:** The header section begins with a <HEAD> tag and is closed with a </HEAD> tag. This section contains the title that is displayed in the navigation bar of the Web page. The title itself is enclosed within the TITLE tag, which begins with a <TITLE> and is closed with a </TITLE>.

The title is of considerable importance. Bookmarks are used to mark a web site. The browser uses the “title” to store the bookmark. Also, when the users are searching for information, the title of the Web page provides the vital keyword that the user is “searching”.

➤ **The body Section:** This comes after the HEAD section. The BODY section contains the text, images and link that you want to display in your Web page. The BODY section begins with a <BODY> tag and ends with a </BODY>.

Example

```
Example
<HTML>
  <HEAD>
    <TITLE> Welcome to the world of
HTML</TITLE>
  </HEAD>
  <BODY>
    <P>This is going to be real fun </P>
  </BODY>
</HTML>
```

2.2. Introduce basic tags

Generally, the first line of an HTML page will be a HEADING tag. If you think about a HEADING as part of an outline of a document, the first HEADING (<H1></H1>) tag is roman numeral one, a second level heading (<H2></H2>) would be roman numeral two, and so on. In most cases, the first heading on a web page will be the same as, or similar to, the document title to let people know right off what the page is about.

<H1> is the largest size heading, which you would normally use at the start of a document. <H6> is the smallest, with <H2> to <H5> of varying sizes in between them.

2.2.1. Page formatting tags:

These tags affect how the text is spaced on a page:

➤ <P> </P> - Paragraph. Inserts an empty line (it's like double-spacing in word processing).

The spaces between this line and the line above and below are examples of paragraph tags.

➤ <HR> - Horizontal line. Useful for breaking up sections of your page. Creates a shadowed line across the page. The shadowed line you see below this text is an example of the <HR> tag. The <HR> tag is one of those exceptions I mentioned, and doesn't need a closing tag.

➤
 - Line break. Doesn't insert a space between lines, just forces a break between lines of text. TIP: if you want to create blank lines on your page, use multiple
 tags, not the <P> tag as the browser only sees the first <P> tag

and ignores the others but sees and creates a line break for all
 tags. The space (or lack of it, actually) between this line and the line above it is an example of a line break. The
 tag doesn't need a closing tag.

2.2.2. Text style tags:

Text style tags affect the appearance of text on a page. You already know about text style tags as you use them all the time in word processors like MS Word. When you select text and click on the B button in MS Word your text turns bold, right? HTML works the same way, but we need to use HTML tags to tell the browser what text style to use.

The two most useful text style tags are:

 ... - Bold

Example of bold text tag.

<I> ... </I> - Italics

<I>Example of italicized text tag at work.</I>

Text Formatting Tags

Tag	Description
	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
	Defines deleted text
<s>	Deprecated. Use instead
<strike>	Deprecated. Use instead
<u>	Deprecated. Use styles instead

2.2.3. Tables

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). The letters td

stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

Example

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

2.2.4. The Image Tag and the Src Attribute

In HTML, images are defined with the `` tag.

The `` tag is empty, which means that it contains attributes only and it has no closing tag.

To display an image on a page, you need to use the `src` attribute. `src` stands for "source". The value of the `src` attribute is the URL of the image you want to display on your page.

The syntax of defining an image:

```

```

The URL points to the location where the image is stored. An image named "boat.gif" located in the directory "images" on "www.w3schools.com" has the URL: <http://www.w3schools.com/images/boat.gif>.

The browser puts the image where the image tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

2.2.5. Hyperlinks, Anchors, and Links

In web terms, a hyperlink is a reference (an address) to a resource on the web.

Hyperlinks can point to any resource on the web: an HTML page, an image, a sound file, a movie, etc.

An anchor is a term used to define a hyperlink destination inside a document.

The HTML anchor element `<a>`, is used to define both hyperlinks and anchors.

We will use the term HTML link when the `<a>` element points to a resource, and the term HTML anchor when the `<a>` elements defines an address inside a document..

An HTML Link

Link syntax:

```
<a href="url">Link text</a>
```

The start tag contains attributes about the link.

The element content (Link text) defines the part to be displayed.

Note: The element content doesn't have to be text. You can link from an image or any other HTML element.

The href Attribute

The href attribute defines the link "address".

This `<a>` element defines a link to HUT:

```
<a href="http://www.hut.edu.vn/">Visit HUT!</a>
```

The code above will display like this in a browser:

[Visit HUT!](http://www.hut.edu.vn/)

2.2. Introduce frame tags and input elements

2.2.1. Frame tags

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- ▲ The web developer must keep track of more HTML documents
- ▲ It is difficult to print the entire page

The Frameset Tag

The <frameset> tag defines how to divide the window into frames

Each frameset defines a set of rows or columns

The values of the rows/columns indicate the amount of screen area each row/column will occupy

The Frame Tag

The <frame> tag defines what HTML document to put into each frame

In the example below we have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The HTML document "frame_a.htm" is put into the first column, and the HTML document "frame_b.htm" is put into the second column:

Example	
<frameset	cols="25%,75%">
<frame	src="frame_a.htm">
<frame	src="frame_b.htm">
</frameset>	

2.2.2 HTML Input Elements

Once we create a form, we can place controls on the form to accept input from the user. These controls are generally used with a <FORM> element. However, we can also use them outside a form to create user interfaces.

The INPUT element

The <INPUT> element defines the type and appearance of the control to be displayed on the form. The attributes of the element are:

Attributes	Description
TYPE	This specifies the type of element. We have a choice of TEXT, PASSWORD, CHECKBOX, RADIO, SUBMIT, RESET, FILE, HIDDEN and BUTTON. The default is TEXT.
NAME	This specifies the name of control.
VALUE	This is an optional attribute that specifies the initial value of the control. However, if the TYPE is RADIO then we have to specify a value.
SIZE	This specifies the initial width of the control.
MAXLENGTH	This is used to specify the maximum number of characters that can be entered in the TEXT or PASSWORD element. The default is unlimited.
CHECKED	This is a Boolean attribute that specifies if the button is on. This attribute is used when the input type is RADIO or CHECKBOX
SRC	SRC = "URL". This is used when we are using an IMAGE as the input type.
Button	
Name	
NAME	This sets or retrieves the name of the control.
SIZE	This sets or retrieves the size of the control.
TYPE	This retrieves the type of intrinsic control represented by the <INPUT type=button>.
VALUE	This sets or retrieves the value of the <INPUT type=button>.

Example

```
<INPUT TYPE=button VALUE="click" NAME="b1">
```

Text

This creates a single-line text entry control. The SIZE attribute defines the number of characters that can be displayed in the Text element. The MAXLENGTH attribute specifies the maximum number of characters that can be entered in the Text element.

Example

```
<INPUT TYPE=text VALUE="" NAME="textbox" SIZE=20>
```

The element Value here displays the initial text string and retrieves the text that is specified when the form is submitted.

Checkbox

This creates a checkbox. The user can select more than one checkbox. When a checkbox element is selected, a name/ value pair is submitted with the FORM. The default value of checkbox is on. The checkbox element is an inline element and does not require a closing tag.

Name	Description
CHECKED	This sets or retrieves the state of the checkbox.
NAME	This sets or retrieves the name of the checkbox.
SIZE	This sets or retrieves the size of the checkbox.
STATUS	This sets or retrieves whether the checkbox is selected.
TYPE	This sets or retrieves the type of intrinsic control represented by the <INPUT type=checkbox>.
VALUE	This sets or retrieves the value of the <INPUT type=checkbox>.

Radio

This creates a radio button control. A radio button control is used mutually for exclusive sets of values. Each radio button control in the group should be given the same name. The user can select only one option at any given time. Only the selected radio button in the group generates a name/value pair in the submitted data. Radio buttons require an explicit value property.

Name	Description
CHECKED	This sets or retrieves the state of the radio button.
NAME	This sets or retrieves the name of the control
SIZE	This sets or retrieves the size of the control
STATUS	This sets or retrieves whether the radio button is selected.
TYPE	This retrieves the type of intrinsic control represented by the <INPUT type=radio>.
VALUE	This sets or retrieves the value of the <INPUT type=radio>.

Example

```
<INPUT TYPE=radio NAME="sex" VALUE="male">Male
```



Chapter 4

1. Stylesheet

1.1 What are style sheets?

Style sheets describe how documents are presented on screens, in print, or perhaps how they are pronounced. W3C has actively promoted the use of style sheets on the Web since the Consortium was founded in 1994. The Style Activity has produced several W3C Recommendations (CSS1, CSS2, XPath, XSLT). CSS especially is widely implemented in browsers.

By attaching style sheets to structured documents on the Web (e.g. HTML), authors and readers can influence the presentation of documents without sacrificing device-independence or adding new HTML tags.

The easiest way to start experimenting with style sheets is to find a browser that supports CSS. Discussions about style sheets are carried out on the www-style@w3.org mailing list and on comp.-infosystems.-www.-authoring.-stylesheets.

The W3C Style Activity is also developing XSL, which consists of a combination of XSLT and “Formatting Objects” (XSL-FO).

1.2 How do I use a style sheet?

A style sheet is saved as a separate document. If you want a web page to follow the rules outlined in a style sheet, it must contain a link to the style sheet. When a browser requests a web page, the web page will link to the style sheet, which will in turn instruct the browser to display the web page using the style attributes defined in the style sheet.

1.3 What are the advantages of style sheets?

Style sheets ensure visual continuity throughout a site. By referring to the same style sheet, all pages in a site can display the same stylistic qualities.

Style sheets simplify your site's maintenance. By concentrating your style definition in one external file, any change you implement in your style sheet will

instantly apply to all the web pages linked to it.

By pulling the style definitions out of your pages, you will make them smaller and faster to download. This will allow you to make more efficient use of your web server space and your data transfer allowance.

1.4 The advantages of separating structure and presentation

For principals

Simple management. Thanks to the separation of presentation and content, a change in layout is easy to implement for the whole site. Even a redesign is relatively easy to implement.

The smaller individual files and the caching mentioned before **save bandwidth** by the web host. Besides improved performance, this can also result in cost cuts.

For web developers and content managers

Separating the presentation of the site results in well-organised content and simple structure. This renders construction and maintenance of the site easier.

Because the presentation is centralised in one or a few files, the appearance of many pages in the site can be modified and supplemented quickly and easily. Without the separation of structure and design, each page would have to be modified individually, which would take considerably more time. A centralised presentation may also result in a consistent appearance of the site.

Often several developers are responsible for the construction of a website. Thanks to the separation principle, developers can largely work on the appearance and the content/structure of the site independent of each other. This speeds up the production process and reduces the chance of flaws.

For visitors

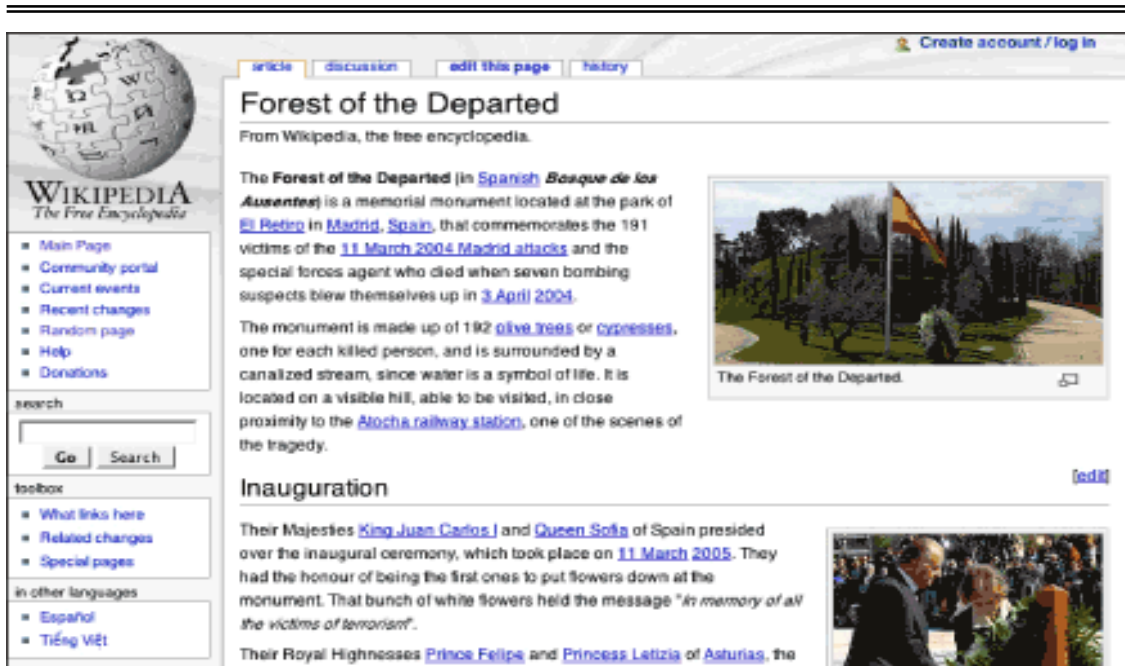
Consistency in the appearance of the site makes a site more recognisable and therefore easier to use.

Individual files are smaller which results in **shorter download times and saving the visitor bandwidth** for the visitor.

Visitors can easily **influence the presentation of a website so that it meets their preferences better**, by means of User Style Sheets.

1.5 Example

Example 1:



Wikipedia uses structural markup to identify headings, lists, and paragraphs, and CSS to define their visual properties. www.wikipedia.org

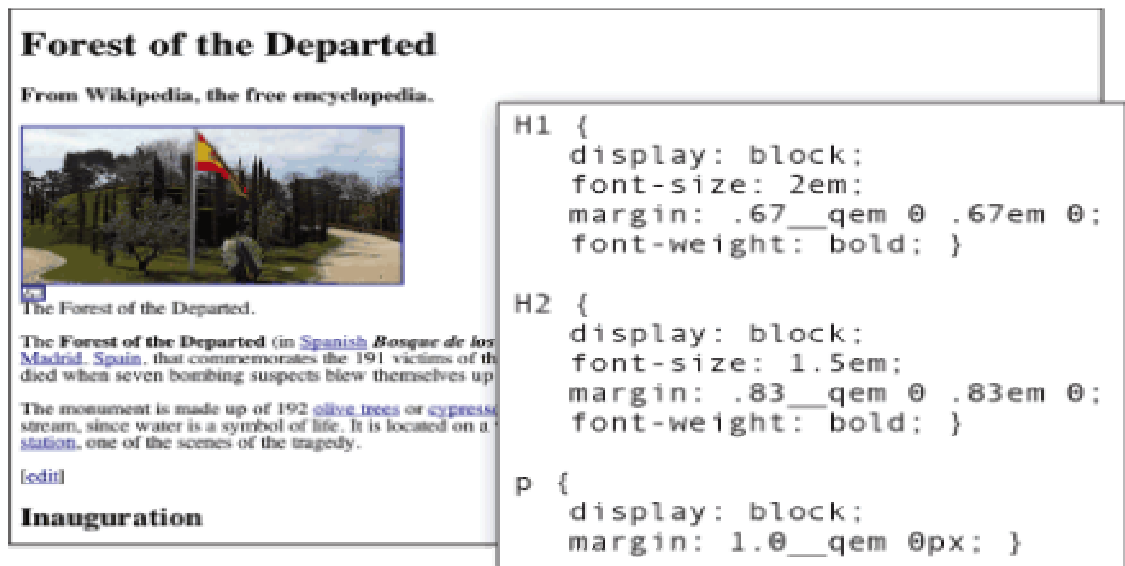
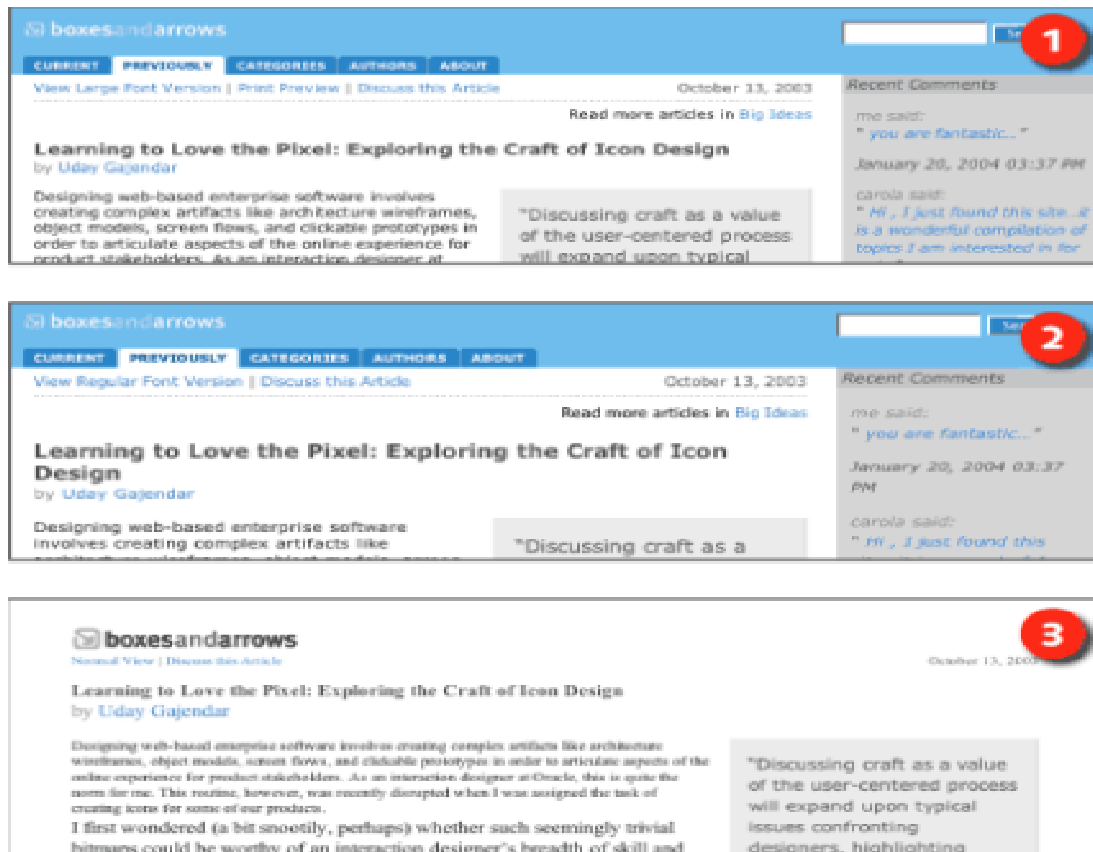


Figure 2.: When structured documents are displayed without styles, the client software determines the visual formatting. Here, the structured Wikipedia page is displayed without styles in Safari. Select Safari style definitions are shown in the inset. www.wikipedia.org

Example 2: HTML document is more readable with CSS

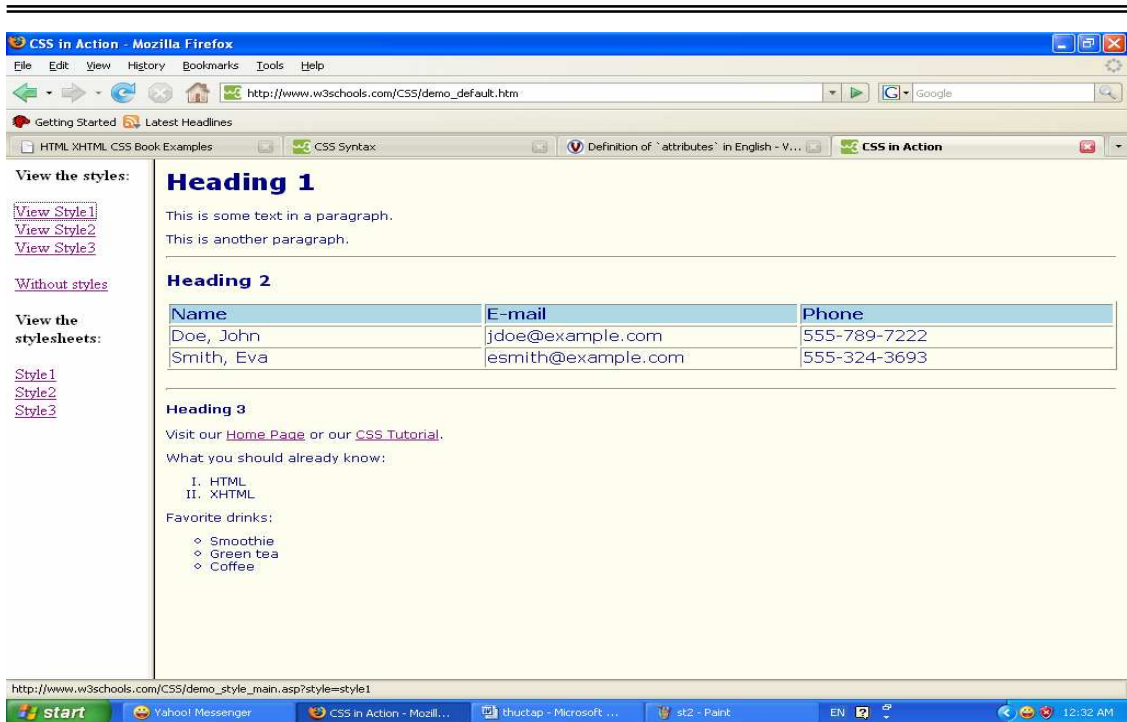
CSS has far more formatting options than standard HTML presentation markup. And since content and presentation are separate, one document can have many different designs simply by applying a different style sheet (Figure 3).



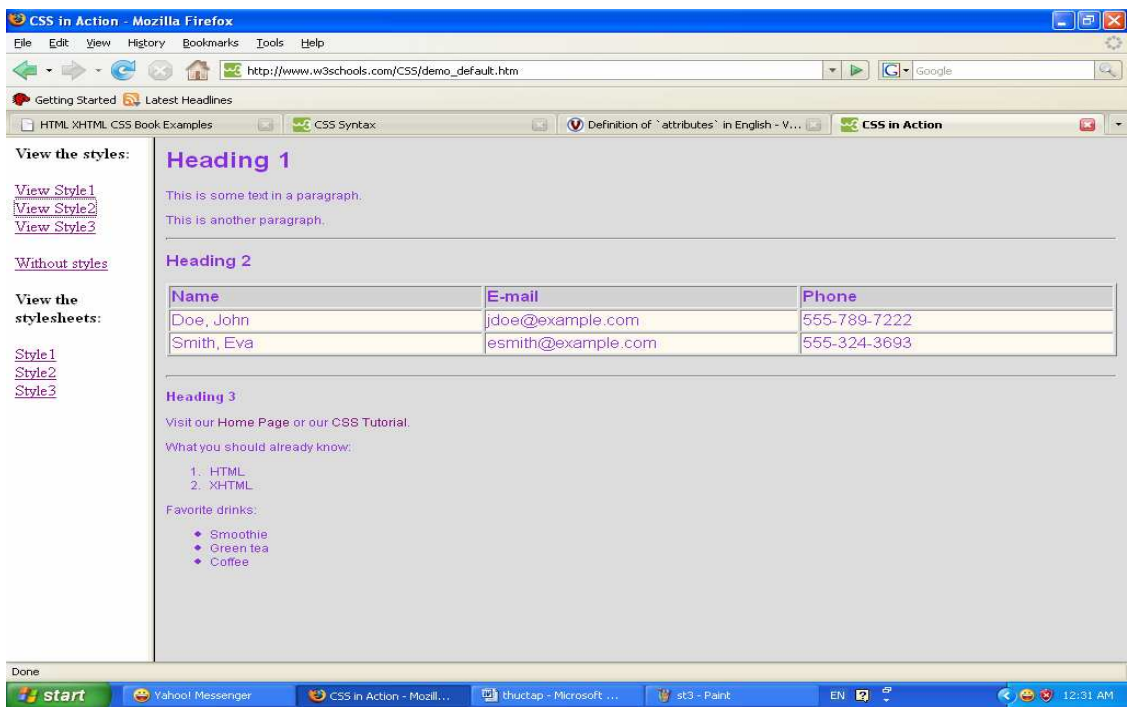
When content and presentation are separate, alternate formats are easy to provide by applying a different style sheet. Through linked style sheets, Boxes and Arrows allows users to choose between regular (1), large-font (2), and print (3) versions. www.boxesandarrows.com

Example 3: An HTML document can be displayed with different styles

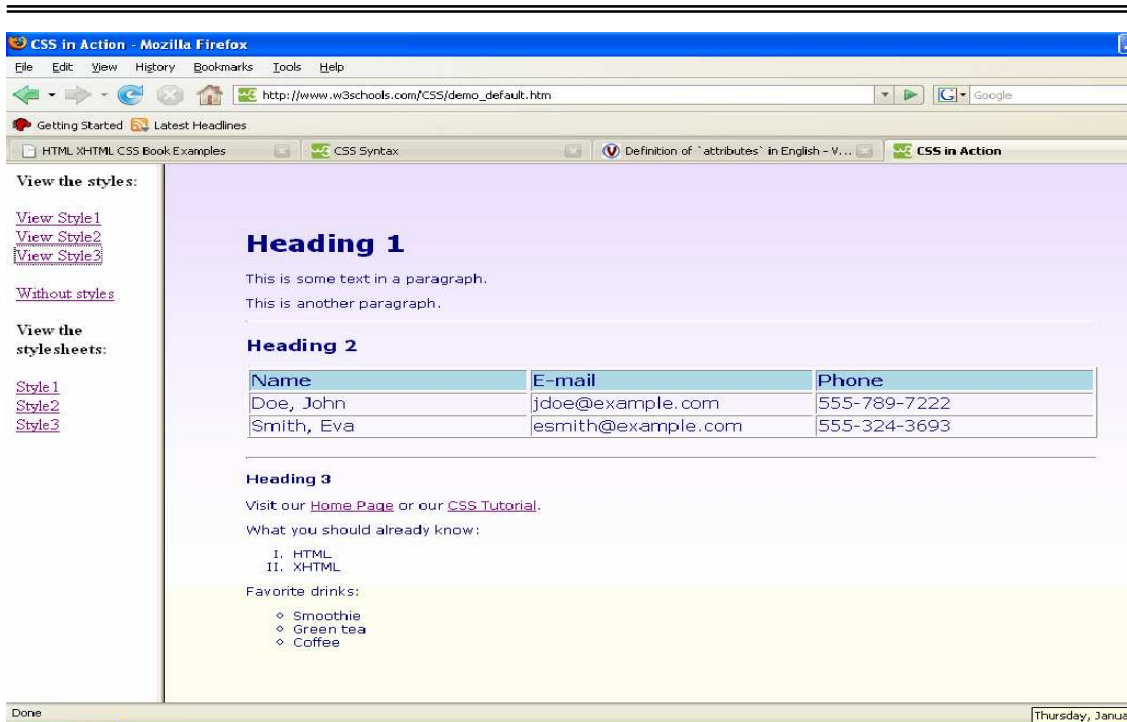
Style 1



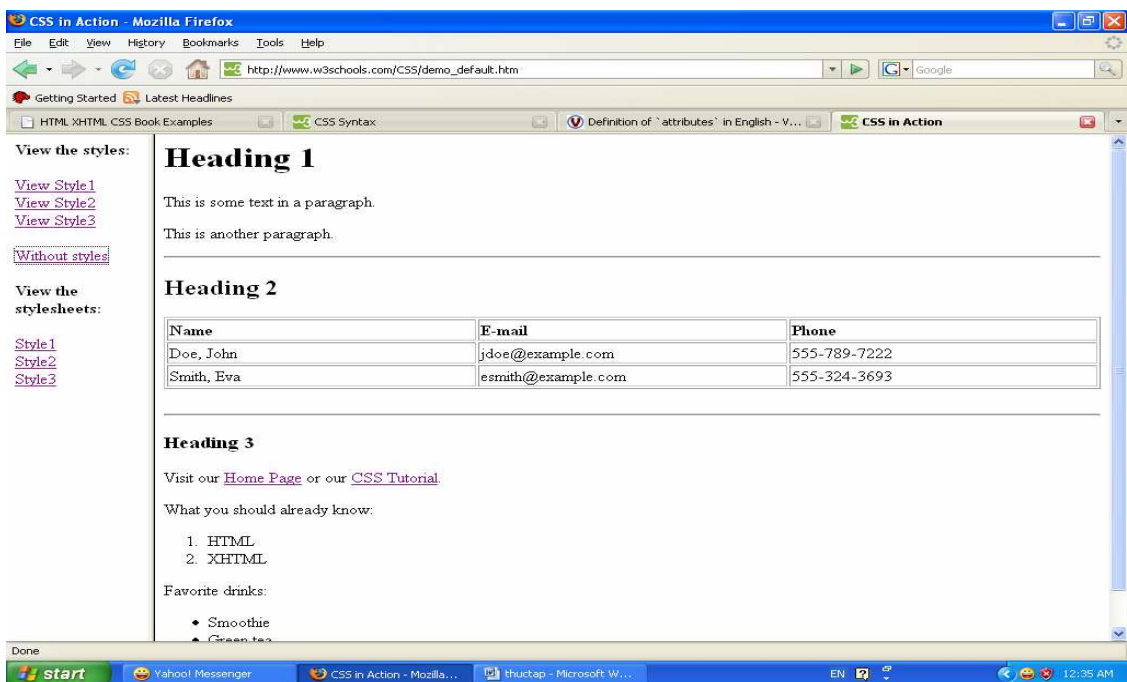
Style 2



Style 3



And without styles



2. Accessibility

2.1 Structure vs. Presentation

When designing a document or series of documents, content developers should strive first to identify the desired structure for their documents before thinking about how the documents will be presented to the user. Distinguishing the structure of a document from how the content is presented offers a number of advantages, including improved accessibility, manageability, and portability.

Identifying what is structure and what is presentation may be challenging at times. For instance, many content developers consider that a horizontal line communicates a structural division. This may be true for sighted users, but to unsighted users or users without graphical browsers, a horizontal line may have next to no meaning. For example, in HTML content developers should use the HTML 4.01 [\[HTML4\]](#) heading elements (H1-H6) to identify new sections. These may be *complemented* by visual or other cues such as horizontal rules, but should not be replaced by them.

The inverse holds as well: content developers should not use structural elements to achieve presentation effects. For instance in HTML, even though the BLOCKQUOTE element may cause indented text in some browsers, it is designed to identify a quotation, not create a presentation side-effect. BLOCKQUOTE elements used for indentation confuse users and search robots alike, who expect the element to be used to mark up block quotations.

The separation of presentation from structure in XML documents is inherent. As Norman Walsh states in "A Guide to XML" [\[WALSH\]](#),

HTML browsers are largely hardcoded. A first level heading appears the way it does because the browser recognizes the H1 tag. Again, since XML documents have no fixed tag set, this approach will not work. The presentation of an XML document is dependent on a stylesheet.

2.2 Text equivalents

Text is considered accessible to almost all users since it may be handled by screen readers, non-visual browsers, and braille readers. It may be displayed visually, magnified, synchronized with a video to create a caption, etc. As you design a document containing non-textual information (images, applets, sounds, multimedia presentations, etc.), supplement that information with textual equivalents wherever possible.

When a text equivalent is presented to the user, it fulfills essentially the same function (to the extent possible) as the original content. For simple content, a text equivalent may need only describe the function or purpose of content. For

complex content (charts, graphs, etc.), the text equivalent may be longer and include descriptive information.

Text equivalents must be provided for logos, photos, submit buttons, applets, bullets in lists, ASCII art, and all of the links within an image map as well as invisible images used to lay out a page.

2.2.1 Overview of technologies

How one specifies a text equivalent depends on the document language.

For example, depending on the element, HTML allows content developers to specify text equivalents through attributes (" alt" or "longdesc") or in element content (the OBJECT element).

Video formats, such as QuickTime, will allow developers to include a variety of alternative audio and video tracks. SMIL ([\[SMIL\]](#)) allows developers to synchronize alternative audio and video clips, and text files with each other.

In creating XML DTDs, ensure that elements that might need a description have some way of associating themselves with the description.

Some image formats allow internal text in the data file along with the image information. If an image format supports such text (e.g., Portable Network Graphics, see [\[PNG\]](#)) content developers may also supply information there as well.

2.2.2 Backward Compatibility

Content developers must consider backward compatibility when designing Web pages or sites since:

- Some user agents do not support some HTML features,
- People may use older browsers or video players,
- Compatibility problems may arise between software

Therefore, when designing for older technologies, consider these techniques:

- Provide inline text equivalents. For example, include a description of the image immediately after the image.

Provide links to long text equivalents either in a different file or on the same page. These are called description links or "d-links". The link text should explain that the link designates a description. Where possible, it should also explain the nature of the description. However, content developers concerned about how the

description link will affect the visual appearance of the page may use more discrete link text such as "[D]", which is recommended by NCAM (refer to [\[NCAM\]](#)). In this case, they should also provide more information about the link target so that users can distinguish links that share "[D]" as content (e.g., with the "title" attribute in HTML).

2.2.3 Alternative pages

Although it is possible to make most content accessible, it may happen that all or part of a page remains inaccessible. Additional techniques for creating accessible alternatives include:

1. Allow users to navigate to a separate page that is accessible, contains the same information as the inaccessible page, and is maintained with the same frequency as the inaccessible page.
2. Instead of static alternative pages, set up server-side scripts that generate accessible versions of a page on demand.
3. Refer to the examples for [Frames](#) and [Scripts](#).
4. Provide a phone number, fax number, e-mail, or postal address where information is available and accessible, preferably 24 hours a day

Here are two techniques for linking to an accessible alternative page:

1. Provide links at the top of both the main and alternative pages to allow a user to move back and forth between them. For example, at the top of a graphical page include a link to the text-only page, and at the top of a text-only page include a link to the associated graphical page. Ensure that these links are one of the first that users will tab to by placing them at the top of the page, before other links.
2. Use meta information to designate alternative documents. Browsers should load the alternative page automatically based on the user's browser type and preferences.

Not every user has a graphic environment with a mouse or other pointing device. Some users rely on keyboard, alternative keyboard or voice input to navigate links, activate form controls, etc. Content developers must ensure that users may interact with a page with devices other than a pointing device. A page designed for keyboard access (in addition to mouse access) will generally be accessible to users with other input devices. What's more, designing a page for keyboard access will usually improve its overall design as well.

Keyboard access to links and form controls may be specified in a few ways:

Image map links

Provide text equivalents for client-side image map areas, or provide redundant text links for server-side image maps. [Refer to the image map section for examples.](#)

Keyboard shortcuts

Provide keyboard shortcuts so that users may combine keystrokes to navigate links or form controls on a page. **Note.** Keyboard shortcuts -- notably the key used to activate the shortcut -- may be handled differently by different operating systems. On Windows machines, the "alt" and "ctrl" key are most commonly used while on a Macintosh, it is the apple or "clover leaf" key. Refer to the [Keyboard access for links](#) and [Keyboard Access to Forms](#) sections for examples.

Tabbing order

Tabbing order describes a (logical) order for navigating from link to link or form control to form control (usually by pressing the "tab" key, hence the name). Refer to the [Keyboard Access to Forms](#) section for examples.

2.3 Device-independent control for embedded interfaces

Some elements import objects (e.g., applets or multimedia players) whose interfaces cannot be controlled through the markup language. In such cases, content developers should provide alternative equivalents with accessible interfaces if the imported objects themselves do not provide accessible interfaces.

2.4 Navigation

A consistent style of presentation on each page allows users to locate navigation mechanisms more easily but also to skip navigation mechanisms more easily to find important content. This helps people with learning and reading disabilities but also makes navigation easier for all users. Predictability will increase the likelihood that people will find information at your site, or avoid it when they so desire.

Examples of structures that may appear at the same place between pages:

1. navigation bars
2. the primary content of a page
3. advertising

A navigation mechanism creates a set of paths a user may take through your site. Providing navigation bars, site maps, and search features all increase the likelihood that a user will reach the information they seek at your site. If your site is highly visual in nature, the structure might be harder to navigate if the user can't form a mental map of where they are going or where they have been. To help them, content developers should describe any navigation mechanisms. It is crucial

that the descriptions and site guides be accessible since people who are lost at your site will rely heavily on them.

When providing search functionality, content developers should offer search mechanisms that satisfy varying skill levels and preferences. Most search facilities require the user to enter keywords for search terms. Users with spelling disabilities and users unfamiliar with the language of your site will have a difficult time finding what they need if the search requires perfect spelling. Search engines might include a spell checker, offer "best guess" alternatives, query-by-example searches, similarity searches, etc.

2.5 Comprehension

The following sections discuss techniques for helping comprehension of a page or site.

2.5.1 Writing style

The following writing style suggestions should help make the content of your site easier to read for everyone, especially people with reading and/or cognitive disabilities. Several guides (including [\[HACKER\]](#)) discuss these and other writing style issues in more detail.

1. Strive for clear and accurate headings and link descriptions. This includes using link phrases that are terse and that make sense when read out of context or as part of a series of links (Some users browse by jumping from link to link and listening only to link text.) Use informative headings so that users can scan a page quickly for information rather than reading it in detail.
 2. State the topic of the sentence or paragraph at the beginning of the sentence or paragraph (this is called "front-loading"). This will help both people who are skimming visually, but also people who use speech synthesizers. "Skimming" with speech currently means that the user jumps from heading to heading, or paragraph to paragraph and listens to just enough words to determine whether the current chunk of information (heading, paragraph, link, etc.) interests them. If the main idea of the paragraph is in the middle or at the end, speech users may have to listen to most of the document before finding what they want. Depending on what the user is looking for and how much they know about the topic, search features may also help users locate content more quickly.
 3. Limit each paragraph to one main idea.
 4. Avoid slang, jargon, and specialized meanings of familiar words, unless defined within your document.
-
-

-
-
5. Favor words that are commonly used. For example, use "begin" rather than "commence" or use "try" rather than "endeavor."
 6. Use active rather than passive verbs.
 7. Avoid complex sentence structures.

To help determine whether your document is easy to read, consider using the Gunning-Fog reading measure (described in [\[SPOOL\]](#) with examples and the algorithm online at [\[TECHHEAD\]](#)). This algorithm generally produces a lower score when content is easier to read. As example results, the Bible, Shakespeare, Mark Twain, and TV Guide all have Fog indexes of about 6. Time, Newsweek, and the Wall St. Journal an average Fog index of about 11.

2.5.2 Multimedia equivalents

For people who do not read well or not at all, multimedia (non-text) equivalents may help facilitate comprehension. Beware that multimedia presentations do not **always** make text easier to understand. Sometimes, multimedia presentations may make it more confusing.

2.6 Content negotiation

There are a variety of strategies to allow users to select the appropriate content:

1. Include links to other versions of content, such as translations. For example, the link "Refer to the French version of this document" links to the French version.
2. Indicate content type or language through markup (e.g., in HTML use "type" and "hreflang").
3. Use content negotiation to serve content per the client request. For example, serve the French version of a document to clients requesting French.

2.7 Automatic page refresh

Content developers sometimes create pages that refresh or change without the user requesting the refresh. This automatic refresh can be very disorienting to some users. Instead, in order of preference, authors should:

1. Configure the server to use the appropriate HTTP status code (301). Using HTTP headers is preferable because it reduces Internet traffic and download times, it may be applied to non-HTML documents, and it may be used by agents who requested only a HEAD request (e.g., link checkers). Also, status codes of the 30x type provide information such as "moved"
-
-

-
-
- permanently" or "moved temporarily" that cannot be given with META refresh.
2. Replace the page that would be redirected with a static page containing a normal link to the new page.

2.8 Screen flicker

A flickering or flashing screen may cause seizures in users with photosensitive epilepsy and content developers should thus avoid causing the screen to flicker. Seizures can be triggered by flickering or flashing in the 4 to 59 flashes per second (Hertz) range with a peak sensitivity at 20 flashes per second as well as quick changes from dark to light (like strobe lights).

2.9 Bundled documents

Bundled documents can facilitate reading offline. To create a coherent package:

- Use metadata to describe the relationships between components of the package (refer to [link metadata](#) for HTML).
- Use archiving tools such as zip, tar and gzip, and StuffIt to create the package.

2.10 Validation

This section discusses strategies and techniques for testing Web documents to determine accessibility issues that have been resolved and those that haven't. These tests should highlight major access issues, and are valuable in reducing a number of accessibility barriers. However, some of these testing scenarios only replicate conditions caused by a disability; they do not simulate the full experience a user with a disability might have. In real-life settings, your pages may be less usable than you expected. Thus, one of the strategies recommends that content developers observe people with different disabilities as they attempt to use a page or site.

If, after completing the following tests and adjusting your design accordingly, you find that a page is still not accessible, it is likely that you should create an [alternative page](#) that is accessible.

2.10.1 Automatic validators

A validator can verify the syntax of your pages (e.g., HTML, CSS, XML). Correct syntax will help eliminate a number of accessibility problems since software can process well-formed documents more easily. Also, some validators can warn you of some accessibility problems based on syntax alone (e.g., a document is missing

an attribute or property that is important to accessibility). Note, however, that correct syntax does not guarantee that a document will be accessible. For instance, you may provide a text equivalent for an image according to the language's specification, but the text may be inaccurate or insufficient. Some validators will therefore ask you questions and step you through more subjective parts of the analysis. Some examples of automatic validators include:

1. An automated accessibility validation tool such as Bobby (refer to [\[BOBBY\]](#)).
2. An HTML validation service such as the W3C HTML Validation Service (refer to [\[HTMLVAL\]](#)).
3. A style sheets validation service such as the W3C CSS Validation Service (refer to [\[CSSVAL\]](#)).

2.10.2 Repair tools

Validators usually report what issues to solve and often give examples of how to solve them. They do not usually help an author walk through each problem and help the author modify the document interactively. The WAI Evaluation and Repair Working Group ([\[WAI-ER\]](#)) is working to develop a suite of tools that will help authors not only identify issues but solve them interactively.

2.10.3 User scenarios

Keep in mind that most user agents (browsers) and operating systems allow users to configure settings that change the way software looks, sounds, and behaves. With the variety of user agents, different users will have very different experiences with the Web. Therefore:

1. Test your pages with a text-only browser such as Lynx ([\[LYNX\]](#)) or a Lynx emulator such as Lynx Viewer ([\[LYNXVIEW\]](#)) or Lynx-me ([\[LYNXME\]](#)).
 2. Use multiple graphic browsers, with:
 - o sounds and images loaded,
 - o images not loaded,
 - o sounds not loaded,
 - o no mouse,
 - o frames, scripts, style sheets, and applets not loaded.
 3. Use several browsers, old and new. **Note.** Some operating systems or browsers do not allow multiple installations of the browser on the same machine. It may also be difficult to locate older browser software.
 4. Use other tools such as a self-voicing browser (e.g., [\[PWWEBSPEAK\]](#) and [\[HOMEPAGEREADER\]](#)), a screen reader (e.g., [\[JAWS\]](#) and [\[WINVISION\]](#)), magnification software, a small display, an onscreen keyboard, an alternative keyboard, etc. **Note.** If a Web site is usable with one of these products it does not ensure that the site will be usable by other
-
-

products. For a more detailed list of assistive technologies used to access the Web refer to ([\[ALTBROWSERS\]](#)).

2.10.4 Spell and grammar checks

A person reading a page with a speech synthesizer may not be able to decipher the synthesizer's best guess for a word with a spelling error. Grammar checkers will help to ensure that the textual content of your page is correct. This will help readers for whom your document is not written in their native tongue, or people who are just learning the language of the document. Thus, you will help increase the comprehension of your page.

2.11 Browser Support

Please refer to the W3C Web site ([\[WAI-UA-SUPPORT\]](#)) for information about browser and other user agent support of accessibility features.

In general, please note that HTML user agents ignore attributes they don't support and they render the content of unsupported elements.

2.12 Technologies Reviewed for Accessibility

"Web Content Accessibility Guidelines 1.0" suggests using W3C technologies since they have been reviewed for accessibility issues and therefore have accessibility features built in. The latest W3C technologies are available from the [W3C Technical Reports and Publications](#) page.

Brief overview of current W3C technologies:

- MathML for mathematical equations
- HTML, XHTML, XML for structured documents
- RDF for meta data
- SMIL to create multimedia presentations
- CSS and XSL to define style sheets
- XSLT to create style transformations
- PNG for graphics (although some are best expressed in JPG, a non-w3c spec)

2.13 Audio information

Auditory presentations must be accompanied by *text transcripts*, textual equivalents of auditory events. When these transcripts are presented synchronously with a video presentation they are called *captions* and are used by people who cannot hear the audio track of the video material.

Some media formats (e.g., QuickTime 3.0 and SMIL) allow captions and video descriptions to be added to the multimedia clip. SAMI allows captions to be added. The following example demonstrates that captions should include speech as well as other sounds in the environment that help viewers understand what is going on.

Until the format you are using supports alternative tracks, two versions of the movie could be made available, one with captions and descriptive video, and one without. Some technologies, such as SMIL and SAMI, allow separate audio/visual files to be combined with text files via a synchronization file to create captioned audio and movies.

Some technologies also allow the user to choose from multiple sets of captions to match their reading skills. For more information see the SMIL 1.0 ([\[SMIL\]](#)) specification.

Equivalentents for sounds can be provided in the form of a text phrase on the page that links to a text transcript or description of the sound file. The link to the transcript should appear in a highly visible location such as at the top of the page. However, if a script is automatically loading a sound, it should also be able to automatically load a visual indication that the sound is currently being played and provide a description or transcript of the sound.

2.14 Visual information and motion

Auditory descriptions of the visual track provide narration of the key visual elements without interfering with the audio or dialogue of a movie. Key visual elements include actions, settings, body language, graphics, and displayed text. Auditory descriptions are used primarily by people who are blind to follow the action and other non-auditory information in video material.

Note. If there is no important visual information, for example, an animated talking head that describes (through prerecorded speech) how to use the site, then an auditory description is not necessary.

For movies, provide auditory descriptions that are synchronized with the original audio. Refer to the section on [audio information](#) for more information about multimedia formats.

2.15 Collated text transcripts

Collated text transcripts allow access by people with both visual and hearing disabilities. They also provide everyone with the ability to index and search for information contained in audio/visual materials.

Collated text transcripts include spoken dialogue as well as any other significant sounds including on-screen and off-screen sounds, music, laughter, applause, etc. In other words, all of the text that appears in captions as well as all of the descriptions provided in the auditory description.

3. Link CSS with HTML

There are three ways to apply CSS to HTML.

3.1 In-line

In-line styles are plonked straight into the HTML tags using the `style` attribute.

They look something like this:

```
<p style="color: red">text</p>
```

This will make that specific paragraph red.

But, if you remember, the best-practice approach is that the HTML should be a stand-alone, **presentation free** document, and so in-line styles should be avoided wherever possible.

3.2 Internal

Embedded, or **internal** styles are used for the whole page. Inside the `head` tags, the **style** tags surround all of the styles for the page.

This would look something like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>CSS Example</title>
<style type="text/css">
    p {
        color: red;
    }

    a {
        color: blue;
    }
</style>
...
```

This will make all of the paragraphs in the page red and all of the links blue.

Similarly to the in-line styles, you should keep the HTML and the CSS files separate, and so we are left with our saviour...

3.3 External

External styles are used for the whole, multiple-page website. There is a **separate CSS file**, which will simply look something like:

```
p {  
    color: red;  
}  
  
a {  
    color: blue;  
}
```

If this file is saved as "web.css" then it can be linked to in the HTML like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html>  
<head>  
    <title>CSS Example</title>  
    <link rel="stylesheet" type="text/css" href="web.css" />  
...  
</head>  
</html>
```

For example, let's say that your style sheet is named **style.css** and is located in a folder named **style**. The situation can be illustrated like this:



The trick is to create a link from the HTML document (default.htm) to the style sheet (style.css). Such link can be created with one line of HTML code:

```
<link rel="stylesheet" type="text/css" href="style/style.css" />
```

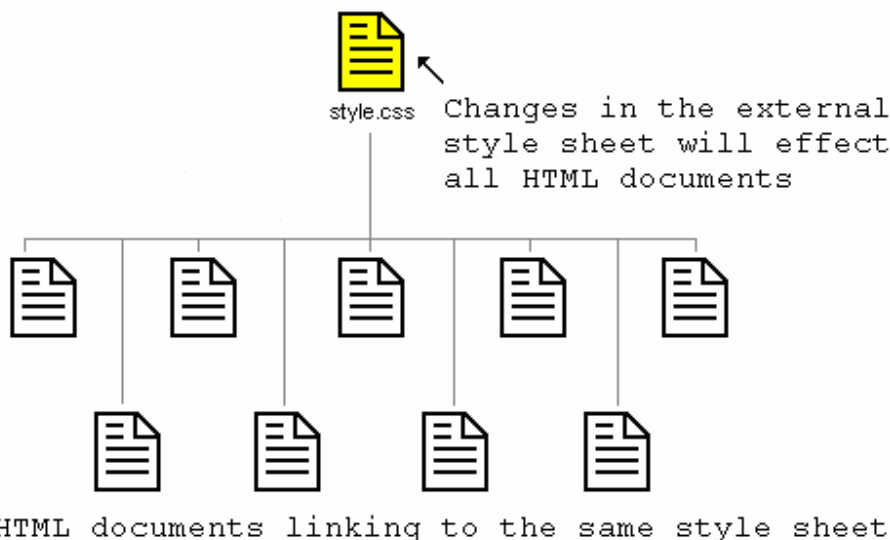
Notice how the path to our style sheet is indicated using the attribute href.

The line of code must be inserted in the header section of the HTML code i.e. between the <head> and </head> tags. Like this:

```
<html>
<head>
  <title>My document</title>
  <link rel="stylesheet" type="text/css" href="style/style.css" />
</head>
<body>
...

```

This link tells the browser that it should use the layout from the CSS file when displaying the HTML file. The really smart thing is that several HTML documents can be linked to the same style sheet. In other words, one CSS file can be used to control the layout of many HTML documents.



This technique can save you a lot of work. If you, for example, would like to change the background color of a website with 100 pages, a style sheet can save you from having to manually change all 100 HTML documents. Using CSS, the change can be made in a few seconds just by changing one code in the central style sheet.

4. Systax of CSS

4.1 What is CSS?

- CSS stands for Cascading Style Sheets
- Styles define how to display HTML elements
- Styles are normally stored in Style Sheets
- Styles were added to HTML 4.0 to solve a problem
- External Style Sheets can save a lot of work
- External Style Sheets are stored in CSS files
- Multiple style definitions will cascade into

4.2 CSS Syntax

The CSS syntax is made up of three parts: a selector, a property and a value:

```
selector {property:value}
```

```
selector {property: value;}
  ↑           ↑           ↙
  What HTML  The property The value of
  tag(s) does the property the property
  apply to   could for     could be red for
  (e.g. "body") example be  example ("#FF0000")
  the background
  color
  ("background-color")
```

The selector is normally the HTML element/tag you wish to define, the property is the attribute you wish to change, and each property can take a value. The property and value are separated by a colon, and surrounded by curly braces:

```
body {color:black}
```

Note: If the value is multiple words, put quotes around the value:

```
p {font-family:"sans serif"}
```

Note: If you want to specify more than one property, you must separate each property with a semicolon. The example below shows how to define a center aligned paragraph, with a red text color:

```
p {text-align:center;color:red}
```

To make the style definitions more readable, you can describe one property on each line, like this:

```
p
{
text-align:center;
color:black;
font-family:arial
}
```

Grouping

You can group selectors. Separate each selector with a comma. In the example below we have grouped all the header elements. All header elements will be displayed in green text color:

```
h1,h2,h3,h4,h5,h6
{
color:green
}
```

The class Selector

With the class selector you can define different styles for the same type of HTML element.

Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph. Here is how you can do it with styles:

```
p.right {text-align:right}
p.center {text-align:center}
```

You have to use the class attribute in your HTML document:

```
<p class="right">This paragraph will be right-aligned.</p>
<p class="center">This paragraph will be center-aligned.</p>
```

Note: To apply more than one class per given element, the syntax is:

```
<p class="center bold">This is a paragraph.</p>
```

The paragraph above will be styled by the class "center" AND the class "bold".

You can also omit the tag name in the selector to define a style that will be used by all HTML elements that have a certain class. In the example below, all HTML elements with class="center" will be center-aligned:

```
.center {text-align:center}
```

In the code below both the h1 element and the p element have class="center". This means that both elements will follow the rules in the ".center" selector:

```
<h1 class="center">This heading will be center-aligned</h1>  
<p class="center">This paragraph will also be center-aligned.</p>
```

Note: Do **NOT** start a class name with a number! It will not work in Mozilla/Firefox.

Add Styles to Elements with Particular Attributes

You can also apply styles to HTML elements with particular attributes.

The style rule below will match all input elements that have a type attribute with a value of "text":

```
input[type="text"] { background-color:blue }
```

The id Selector

You can also define styles for HTML elements with the id selector. The id selector is defined as a #.

The style rule below will match the element that has an id attribute with a value of "green":

```
#green {color:green}
```

The style rule below will match the p element that has an id with a value of "para1":

```
p#para1  
{  
text-align:center;  
color:red  
}
```

Note: Do **NOT** start an ID name with a number! It will not work in Mozilla/Firefox.

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. A comment will be ignored by browsers. A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
p
{
text-align:center;
/*This is another comment*/
color:black;
font-family:arial
}
```

4.3 The simple CSS example

Example 1: text

<pre><html> . <head> <style type="text/css"> body {color:blue} h1 {color:#00ff00} h2 {color:rgb(255,0,0)} </style> </head> . <body> <h1>This is heading 1</h1> <h2>This is heading 2</h2> <p>This is a paragraph. Notice that this text is blue. The default color for a page is defined in the body selector.</p> </body> </html></pre>	<p>This is heading 1</p> <p>This is heading 2</p> <p>This is a paragraph. Notice that this text is blue. The default color for a page is defined in the body selector.</p>
--	--

Example2: Font

<pre><html> <head> <style type="text/css"> p.serif(font-family:"Times New Roman",Georgia,Serif) p.sansserif(font-family:Arial,Verdana,Sans-serif) </style> </head> . <body> <h1>CSS font family example</h1> <p class="serif">This is a paragraph, shown in the Times New Roman font.</p> <p class="sansserif">This is a paragraph, shown in the Arial font.</p> </body> </html></pre>	<p>CSS font family example</p> <p>This is a paragraph, shown in the Times New Roman font.</p> <p>This is a paragraph, shown in the Arial font.</p>
--	---

Example 3:margin

```
<html>
<head>
<style type="text/css">
p.leftmargin {margin-left: 2cm}
</style>
</head>

<body>
<p>This is a paragraph with no margin specified</p>
<p class="leftmargin">This is a paragraph with a specified
left margin</p>
</body>

</html>
```

This is a paragraph with no margin specified

This is a paragraph with a specified left margin

Example 4: Dimension

```
<html>
<head>
<style type="text/css">
p
{
min-height: 100px
}
</style>
</head>
<body>

<p>This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>



</body>
</html>
```

This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.



Chapter 5

1. Dynamic Pages

1.1 What are Dynamic Pages

Dynamic pages consist of pages where the logic is maintained separately from the content. The content is stored in the database until the variable parameters of the dynamic url tells the database repository what items to pull into the page as it loads. Based on the url's parameters, the server will return different content.

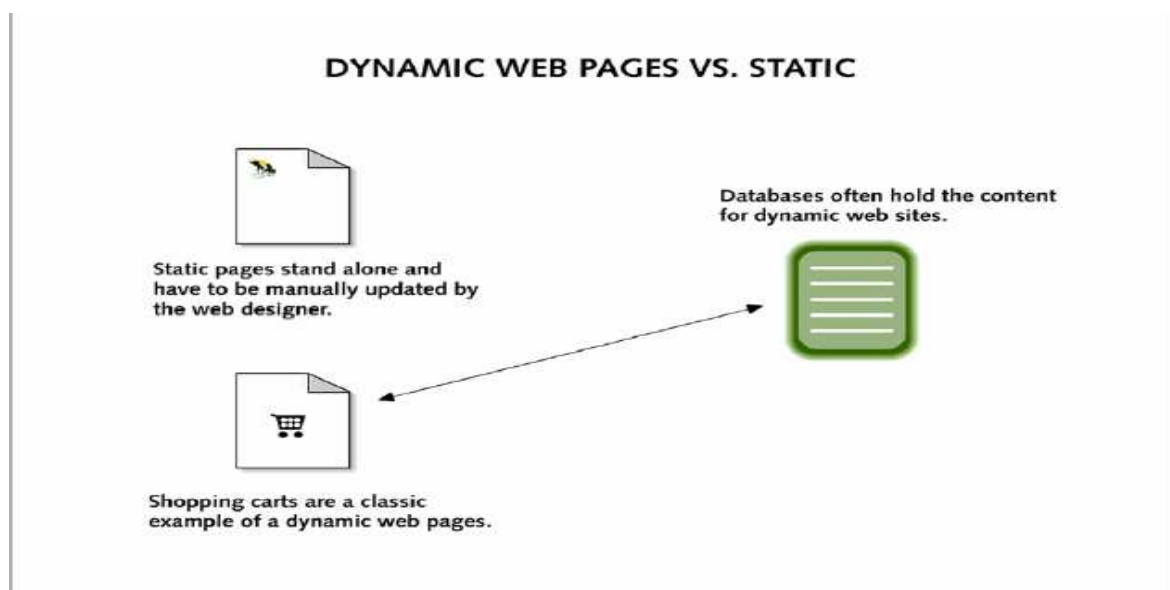
Dynamic pages are usually coded in such scripting languages as CGI, ASP, PHP, Perl, Cold Fusion, JSP and others. You can usually tell if an url is dynamically generated because it will include question marks, equal signs, percentage signs, and ampersands.

An example of a dynamic url is:

`http://www.thedomain.com/index.php?lang=NLD&name=home`

The important element in the url above, is the portion after the ?. It is the portion after the ? that contains the query string - the part that changes.

1.2 The difference between static and dynamic page web



Essentially, static pages are those on your site that send exactly the same response to every request; dynamic pages can customize the response on the server to offer personalization based on cookies and information it can get from the visitor.

An obvious example of a static page is an old style HTML document; the only way to change a HTML page is to upload a new or updated version in its place. Every time a static file is downloaded, the file contents that are sent to the browser are the same for everyone that accesses that file.

While most pages written in a particular scripting language will return a personalized response to each visitor, this is not always the case; the same is true for pages utilizing Server Side Includes. Suppose for a moment that you had two SHTML documents, both of which include a single file; the first including a simple HTML snippet for the menu, the second including an online poll script. The page including the menu would be static; the menu would be the same for everyone that downloaded it until either of the files were changed on the server. The page that included the poll however would be dynamic, as it would display different pages to people depending on their previous voting history.

Flash and Shockwave presentations are also classed as static content; despite the fact that user interaction can lead to different ways of presenting the same data, everyone will download the same file from the server.

1.3 The Advantage of Dynamic Pages

A dynamic page can be customized by a response on a server to help personalize your site to meet your customer's need. All page content will come from a database connected to the Web site.

Since the dynamic template is maintained separately from the content, it means that content changes can be made when needed. In addition, the web site can be updated without major maintenance, editing and reviewing, which translates into lower maintenance costs and time.

Many webmasters of large e-commerce sites tend to favor dynamic pages because it is customer friendly, and allows them to update their pages on a regular basis by specific time and date sensitive routines. Which is understandable, if their products and offers change on a daily basis.

On the affiliate side, I'm seeing more merchant webmasters offering the option of dynamic links for their affiliates. The affiliate can put the code on their website, and let the merchant create the update. This saves the affiliate time, by eliminating the manual manipulation of the url and uploading of their web pages on a daily basis; And second, they have the reassurance of knowing that if something is made

available by the merchant, the link will automatically change and be available to their customers in a timely manner. However, there are some affiliates that do not take a liking to this type of link. Why?

For the established affiliate, with more automation tools -- the merchant's dynamic links takes the control out of their hands. How? The dynamic link may provide specific links to products or services that the affiliate may or may not wish to see appear on their site.

So far, dynamic links looks like a workable solution -- saving time and money and offering only small nuisances. But are there any disadvantages to dynamic pages?

1.4 The Disadvantages of Dynamic Pages

Is there a downside to dynamic pages? Not to be wishy-washy, but -- Yes and No.

In days gone by, many of the web spiders could not read the url parameters to the right of the question mark in the dynamic url. Today, webmasters have seen some improvement.

Google and some of the other search engines can handle simple dynamic urls, but if the query parameters gets to long or complicated (having more than one ? for example) their crawler will ignore the link completely. If you need or want more information on what Google can and cannot do with dynamic links, and the possible procedures and alternatives of getting your dynamic links indexed you can check out [Google Webmaster Center](#) for the details.

Yahoo, on the other hand, suggests that you use dynamic links only in directories that are not intended to be crawled or indexed. Nicely put, the answer is a simple "I don't think so." Thus, whether you have a dynamic web page or website, you may find yourself having problems with getting your web pages crawled, let alone getting them indexed.

So, where do the search engines stand on the issue of dynamic links today? Some web spiders are taking the plunge and crawling websites with dynamic links, which is a positive. But on the negative side, there are still some web spiders that will not venture near a dynamically-created page for fear of getting stuck in the page and being lead through some poorly written code and thus, causing a possible major server crash.

With that being said, what can you do to get your dynamic web pages spidered? Do you have to bit the bullet and change back to static or is there other possibilities out there that will get your dynamic url indexed.

1.5 Example Dynamic pages

An example of a dynamic *google.com.vn*

Dynamic componens:

- Inner search

help user search information quickly

The purpose of the Inner Search Centre is to provide the individual with the tools to develop their own awareness and personal healing experience. Our focus is the journey to the heart centre and a re-alignment with the higher self and your creator source. Discover the Inner Beauty that you have always been. All private sessions and workshops are designed to help you help yourself.

- Member account: You don't have permission to access in Member Area
- Forum,Mail,blog

Static component

- Tools
- Images
- Icon

2 Client-side and Server-side

2.1 Server side

With server-side scripting, completing an activity involves sending information to another computer ([server](#)) across the internet. The server then runs a program that process the information and returns the results, typically a webpage.

Search engines use server-side processing. When a keyword is sent, a program on a server matches the word or phrase entered against an index of website content. (To complete the same search as a client-side process would require the browser to download the entire search engine program and index.)

1)The browser sends an HTTP request.

2)The server retrieves the requested file with the script.

3)The server executes the script or program which typically outputs an HTML web page.

4)The server sends the HTML output to the client's browser.

5)Example: www.google.com

Server-side scripting languages include [ASP](#) and [PHP](#).

2.2 Client side

Client side (that is to say: in the browser) or what is commonly called DHTML ... dynamic HTML.

DHTML is basically taking HTML and JavaScript (sometimes VB script) to make the web page change it's own content (as far as the viewer is concerned) without having to reload or load a new page.

Examples of DHTML would include drop down menus, 'floating' images that hover over the rest of the page etc ... if you [look around](#), you will find plenty on the web.

Client-side scripting languages include [JavaScript](#).

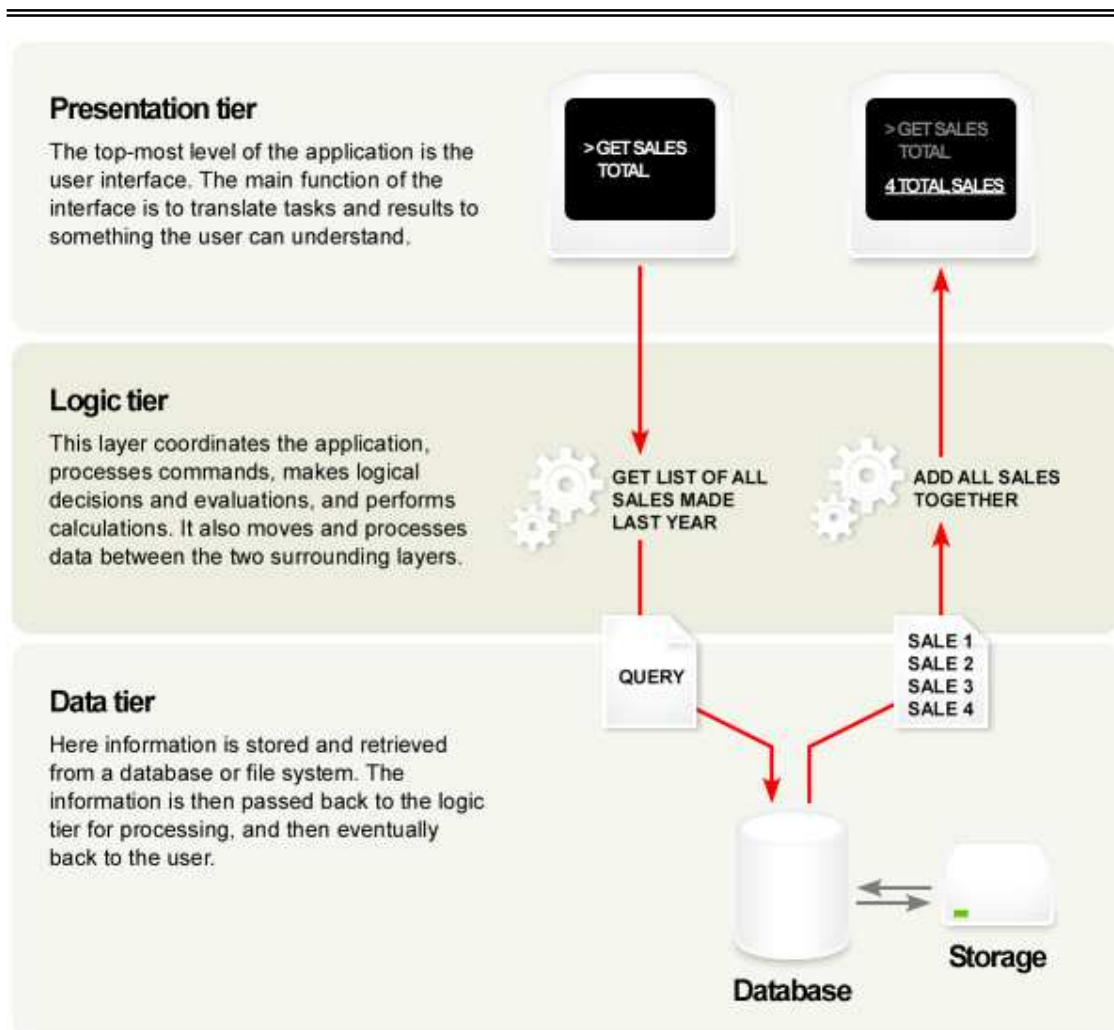
3. Three-Tier sytem

[client-server architecture](#) in which the [user interface](#), [functional process logic](#) ("business rules"), [computer data storage](#) and data access are developed and maintained as independent [modules](#), most often on separate [platforms](#).

The three-tier model is considered to be a [software architecture](#) and a [software design pattern](#).

Apart from the usual advantages of modular [software](#) with well defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements or [technology](#) change. For example, a change of [operating system](#) in the *presentation tier* would only affect the user interface code.

Typically, the user interface runs on a desktop [PC](#) or [workstation](#) and uses a standard [graphical user interface](#), functional process logic may consist of one or more separate modules running on a workstation or [application server](#), and an [RDBMS](#) on a [database server](#) or [mainframe](#) contains the computer data storage logic. The middle tier may be multi-tiered itself (in which case the overall architecture is called an "n-tier architecture").



The 3-Tier architecture has the following three tiers:

3.1 Presentation Tier

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network.

3.2 Application Tier (Business Logic/Logic Tier)

The logic tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.

3.3 Data Tier

This tier consists of Database Servers. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

Example: SIS in HUT <http://sis.hut.edu.vn/>

-
-
- Presentation Tier:

forms and pages using display database: *CourseBL.aspx*, *GroupListBL.aspx*, *ClassEnrollmentBL.aspx*...

- Application Tier:

functions using *insert,select,update,delete* data from database

- Data Tier:

is those class using retrieved data from tables of database:
*class,student,department,..*are class in Data Tier

Chapter 6.

1. Scripting in a web browser

1.1 JavaScript

JavaScript is an [interpreted](#) programming or [script](#) language from Netscape. It is somewhat similar in capability to Microsoft's [Visual Basic](#), Sun's [Tcl](#), the UNIX-derived [Perl](#), and IBM's [Rexx](#). In general, script languages are easier and faster to code in than the more structured and [compiled](#) languages such as [C](#) and [C++](#). Script languages generally take longer to process than compiled languages, but are very useful for shorter programs.

JavaScript is used in Web site development to do such things as:

- Automatically change a formatted date on a Web page
- Cause a linked-to page to appear in a popup window
- Cause text or a graphic image to change during a [mouse rollover](#)

JavaScript uses some of the same ideas found in [Java](#), the compiled [object-oriented programming](#) derived from C++. JavaScript code can be imbedded in [HTML](#) pages and interpreted by the Web browser (or client). JavaScript can also be run at the server as in Microsoft's [Active Server Pages](#) before the page is sent to the requestor. Both Microsoft and Netscape browsers support JavaScript, but sometimes in slightly different ways.

1.2 Applet

Applet is java program that can be embedded into HTML pages. Java applets runs on the java enables web browsers such as mozilla and internet explorer. Applet is designed to run remotely on the client browser, so there are some restrictions on it. Applet can't access system resources on the local computer. Applets are used to make the web site more dynamic and entertaining.

Java applets are executed in a sandbox by most web browsers, preventing them from accessing local data. The code of the applet is downloaded from a web server and the browser either embeds the applet into a web page or opens a new window showing the applet's user interface. The applet can be displayed on the web page by making use of the deprecated applet HTML element [1], or the recommended object element [2]. This specifies the applet's source and the applet's location statistics.

1.2.1 Advantages of Applet:

-
-
- Applets are cross platform and can run on Windows, Mac OS and Linux platform
 - Applets can work all the version of Java Plugin
 - Applets runs in a sandbox, so the user does not need to trust the code, so it can work without security approval
 - Applets are supported by most web browsers
 - Applets are cached in most web browsers, so will be quick to load when returning to a web page
 - User can also have full access to the machine if user allows

1.2.2 Disadvantages of Java Applet:

- Java plug-in is required to run applet
- Java applet requires JVM so first time it takes significant startup time
- If applet is not already cached in the machine, it will be downloaded from internet and will take time

- Its difficult to desing and build good user interface in applets

1.3 Flash

Over the releases of new versions of Flash, Macromedia has made Flash more and more controllable via programming, where they have it positioned as a competitor to HTML to build interactive web sites and applications such as an e-commerce store. Macromedia argues that Flash is the way to go instead of HTML because of the following reasons:

- Flash movies load faster and save on download time because Flash is vector based whereas HTML is not.
- Flash intelligently 'caches' it's movies so they don't have to be reloaded.
- Flash gives the user (the person viewing/using the Flash movie) a more responsive 'rich-client' like experience.

All of these points are true, but they can be true for HTML pages as well (except for the vectors). I will address these points now:

Flash pages can be made to load faster, but most of the time, the way they are designed in the real world, they do not. That is not a Flash problem, it is more an issue of the Flash developers going nuts with fancy and heavy Flash movies.

HTML caches pages as well, once images are downloaded they are held in your browser's cache. The cached images are then used instead of downloading them from the server again.

With new technology like ASP.net and Java Server Faces, HTML now can react just like a 'rich-client' application. Even without these new tools, properly designed HTML for most dynamic sites can provide a good user experience.

1.3.1 What do I have against Flash?

Before I start trashing Flash, I have to first say that I think it is a great tool, but not in all things and certainly not in the all-encompassing way that Macromedia would suggest. Here's why:

Flash handles text very poorly. The web for the most part is about text, we go to the web to read about things, whether it be articles like this one, or what is in your shopping cart, or the latest baseball stats, it is all text.

Flash text rendering/displaying is much slower than HTML and noticeably less clear. Macromedia knows this and that's why they include the ability to display basic HTML in Flash and that's why on their own site, they make heavy use of HTML.

Flash development usually takes much longer than the HTML equivalent. This has been helped to a great degree starting with the release of FLASH MX where they essentially provided HTML form components, but it is still slower to create a Flash site than an HTML site.

1.3.2 What I like about Flash development?

1. You can do some really nice work in Flash that would be difficult and sometimes impossible in HTML alone. Things like complex animations and playing video spring to mind.
2. Practically no browser issues: For the most part, Flash movies will work the same if the user is on Netscape or IE, on Mac or PC. The browser issues (where people coming to your site have different browsers that can 'break' your pages) are quickly becoming a thing of the past since most people (thank the web gods) are using IE. But even today, I still have to deal with people who may be using some old browser that can break all but the simplest of HTML code.

1.3.3 So when should you use Flash?

In my humble opinion, I would use Flash to create a presentation that is similar to a television commercial. This type of presentation is something where the user sits back and enjoys the show as the Flash movie delivers the message to the client with animation, sound, and possibly video. Please do not get this confused with those ubiquitous 'intro' animations that still plague many Flash sites. Rather I am

talking about informative movies that the user can decide to view to learn about something like a product or a service.

I can also see Flash being used in straight animations, like what your kids watch Saturday morning. One last use where I find Flash handy, is with so called 'rich-ui' components like calendars or fancy navigational systems. The only danger here is that if the user doesn't have the proper Flash plug-in, they won't be able to use those components and as such in many cases (like with a Flash based navigation bar) the user will not be able to use your web site!

Instead of Macromedia's vision, I see Flash being used selectively to enhance an HTML based site.

2.Javascript

2.1 Syntax javascript

2.1.1 JavaScript Variables

Like other programming languages, JavaScript variables are a container that contains a value - a value that can be changed as required.

For example, you could prompt your website users for their first name. When they enter their first name you could store it in a variable called say, `firstName`. Now that you have the user's first name assigned to a variable, you could do all sorts of things with it like display a personalised welcome message back to the user for example. By using a variable to store the user's first name, you can write one piece of code for all your users.

Declaring JavaScript variables

First, you need to declare your variables. You do this using the `var` keyword. You can declare one variable at a time or more than one. You can also assign values to the variables at the time you declare them.

Different methods of declaring JavaScript variables

```
// declaring one javascript variable  
var firstName;
```

```
// declaring multiple javascript variables  
var firstName, lastName;
```

```
// declaring and assigning one javascript variable
```

```
var firstName = 'Homer';
```

```
// declaring and assigning multiple javascript variables
```

```
var firstName = 'Homer', lastName = 'Simpson';
```

Using JavaScript variables

Although there are many uses for JavaScript variables, here is a quick and dirty example:

```
<script language="javascript" type="text/javascript" >
```

```
<!-- hide me
```

```
var firstName = prompt("What's your first name?", "");
```

```
// end hide -->
```

```
<!-- hide me
```

```
document.write(firstName);
```

```
// end hide -->
```

```
</script>
```

The above example opens a JavaScript prompt, prompting the user for their first name. It will then write the name to the page (in practice, you would output the name somewhere between the `<body></body>` tags).

I suspect you can find a much better use for your javascript variables but this simply to demonstrate how easily you can store data inside a variable - data that could change at any moment.

Rules for JavaScript Variables

- Can contain any letter of the alphabet, digits 0-9, and the underscore character.
- No spaces
- No punctuation characters (eg comma, full stop, etc)
- The first character of a variable name cannot be a digit.
- JavaScript variables' names are case-sensitive. For example *firstName* and *FirstName* are two different variables

2.1.2 JavaScript Functions

In JavaScript, you will use functions a lot. A function (also known as a *method*) is a self-contained piece of code that performs a particular "function". You can

recognise a function by its format - it's a piece of descriptive text, followed by open and close brackets.

Sometimes there will be text in between the brackets. This text is known as an *argument*. An argument is passed to the function to provide it with further info that it needs to process. This info could be different depending on the context in which the function is being called.

Arguments can be extremely handy, such as allowing your users to provide information (say via a form) that is passed to a function to process. For example, your users could enter their name into a form, and the function would take that name, do some processing, then present them with a personalised message that includes their name.

A function doesn't actually do anything until it is called. Once it is called, it takes any arguments, then performs it's function (whatever that may be).

Writing a function in JavaScript

It's not that hard to write a function in JavaScript. Here's an example of a JavaScript function.

Write the function:

```
<script type="text/javascript">
<!--
function displayMessage(firstName) {
    alert("Hello " + firstName + ", hope you like JavaScript functions!")
}
//-->
</script>
```

Call the function:

```
<form>
First name:
<input type="input" name="yourName" />
<input
    type="button"
    onclick="displayMessage(form.yourName.value)"
    value="Display Message" />
</form>
```

This should work like this:

First name:

Explanation of code

Writing the function:

1. We started by using the *function* keyword. This tells the browser that a function is about to be defined
2. Then we gave the function a name, so we made up our own name called "displayMessage". We specified the name of an argument ("firstName") that will be passed in to this function.
3. After the function name came a curly bracket {. This opens the function. There is also a closing bracket later, to close the function.
4. In between the curly brackets we write all our code for the function. In this case, we use JavaScript's built in `alert()` function to pop up a message for the user.

Calling the function:

1. We created an HTML form with an input field and submit button
2. We assigned a name ("yourName") to the input field
3. We added the *onclick event handler* to the button. This event handler is called when the user clicks on the button (more about event handlers later). This is where we call our JavaScript function from. We pass it the value from the form's input field. We can reference this value by using "form.yourName.value".

2.1.4 Control structures in JavaScript

The JavaScript loops and conditions are similar to those of the C language but with greater flexibility and some extensions.

if else

The syntax is:

```
if(condition) { }
```

or

```
if(condition) { } else { };
```

The brackets are optional if there is a single instruction while parentheses are always required.

When the evaluation of the condition returns "true", the instructions are executed otherwise it is the *else* part when it is present.

Example:

```
if(a == 5)
{
    document.write("a is 5");
}
```

It would be possible as in C to assign a variable inside the condition, a practice that should be avoided.

The for loop

The syntax is:

```
for(var = initializer; condition; increment)
{
    ...instructions...
}
```

Example:

```
for(var i = 0; i < 10; i++)
{
    document.write(i + "<br>");
}
```

For in

This structure allows parsing the contents of an object to return the list of its properties. If it is an array, the properties are the indices of the array. To get the contents of the object, the array must be indexed with the property to retrieve its value.

Example:

```
arr = new Array("a", "b", "c");
for (x in arr)
{
    document.write(x + " " + arr[x]);
}
```

This code displays:

- 0) a
 - 1) b
 - 2) c
-
-

For each in (pm)

For each gives directly the content of the object and works as the *foreach* PHP control. This structure has been added to JavaScript 1.6 and Internet Explorer 7 recognizes only version 1.5, it will not work with this browser. Do not use on a public site, so.

Example with the same array:

```
for each(x in arr)
{
    document.write(x);
}
```

While

Loop running as a given condition is true, implying that the conditional expression contains a variable that is modified in the body of the loop.

```
while(condition) { }
```

Example:

```
var i = 0;
while(i < 3)
{
    document.write(arr[i]);
    i++;
}
```

a
b
c

It is easy to forget to increment the variable of the condition, which causes an infinite loop and blocks the browser. Use therefore with cautious. The *for* loop of the following example will have the same result and is therefore preferable, as is *for in*:

```
for(i = 0; i < 3; i++)
{
    document.write(arr[i]);
}
```

Break and continue

The *break* instruction can exit the loop, while *continue* moves to the next iteration. In the example, is created an endless loop with a condition "true" that will be always *true* of course, and it relies on the *break* command to end the loop:

```
var arr = ["a", "b", "c", "d", "e"];
var x = 0;
while(true)
{
  if (x == 2) { x++; continue; }
  if (x == arr.length) break;
  document.write(arr[x]);
  x++;
}
```

The "c" string is not displayed because the loop continues when the index reaches 2. The loop stops when the size of the table is reached, thanks to the *break* instruction.

```
a
b
d
e
```

If a *while* control can easily turn into an infinite loop, things get even worse with the use of the *continues* command as it can bypass the instruction to increment the variable of the condition and also skip the *break* instruction.

The ability to associate a label still does not help us to avoid this problem because the label must be declared prior to use *continue*.

```
label:
...instructions...
continue label;
```

What makes this option of little interest.
do while

do while is similar to the *while* structure with the condition postponed to end of the loop.

The content of the loop will always be executed at least once.

```
do { } while(condition)
```

Example:

```
var i = 0;
do
{
  document.write(arr[i]);
  i++;
} while(i < 3);
```

```
  a
b
c
```

In this case, there is no difference with *while*. The difference appears only if the condition is never met. For example ($i > 4$) does nothing with *while* and displays "a" with *do while*.

Switch case

Executes a processing depending on the value of a conditional expression.

Syntax:

```
switch(expression)
{
  case value:
    ... instructions ...
    break;
  case value:
    ...instructions...
    break;
  ....
}
```

The cases are compared in turn and the first value that corresponds to the expression is retained, and the associated code is executed. The *break* instruction marks the end of code for the case, so if *break* is omitted, the code of the following case is executed in turn.

Conclusion

JavaScript inherits the unsafe syntax and structures of the C language, which includes the aberration of the assignment within a condition, with effects amplified in a client-server environment. We must be attentive to possible infinite loops and prefer to use *for and for .. in*.

2.2 Using javascript to varify data in a form

2.2.1 Description

This article discusses how to use JavaScript to validate important types of form data, including names, addresses, URLs, email addresses, phone numbers, zip codes, expiration dates and credit card numbers (Visa, Master Card, Discover, and American Express, in both Canadian and US formats, with either 13, 14, 15 or 16 digit account numbers). Each data validation function returns an array of valid inputs that were detected, and has the ability to filter and reformat data to desired appearances and standards. If no valid input is detected, then an error code is returned. In addition to providing definitions for each error code number, the JavaScript form validation script also provides associated human-readable error messages which explain the error after it has occurred.

Also, a user input validation function is provided which stops falsified user information from being submitted to business Web sites. This function is easy to add to any Web form by creating a list of form objects, and registering the function as the `onsubmit` event handler for the form. The programming logic allows for relations to be expressed between associated form fields when performing user input validation. For added flexibility, the validation process allows the form creator to permit specific data entry fields to contain unverified or unusual data at the time the form is successfully validated and submitted.

2.2.2 Introduction

This validation script offers a collection of validation functions, and an easy way to use them in your form with JavaScript. Ideally, validation should be helpful to users and make it easier for customers to fill out forms. Be aware that there are some precautions to take note of, which are discussed later on.

Within a form there are fields for text, and/or selector menus. For some forms with only a few values it's easy to offer options via a select menu. Here, data validation isn't as important because the only options given are the ones allowed. For other data fields, like zip codes, listing all variations is impossible, so the easiest way is to use a text field. Be aware that a text field allows anything to be typed, so it's often necessary to validate it by determining whether the value is reasonable. A zip code, for instance, needs to be at least five digits, meaning that five spaces could easily be recognized as an invalid zip code.

The script is written to make validation easy to use. While primarily intended for text fields, it's also applicable to other fields which have pre-programmed values. When working with text fields it's able to validate the field value itself, but when working with other fields it's able to validate the entire field in combination with other fields and functions.

The JavaScript validation programming interface consists of four components:

-
-
- Low-level processing functions for numerical values, text, and whitespace.
 - Validation functions for each type of data
 - Validation handler used for `onsubmit`, which supports generic form validation using a list of form elements, or fine-tuned validation by using a validation profile customized for your form
 - Definitions of error codes and user-friendly error code messages

We display an example form along with its source code to show how the validation program is used. Then we explain how to use the built-in validation functions, how to add a customized validation function, how to add your own error codes and error messages, and finally, how to register the validation function as the `onsubmit` event handler for your form.

2.2.3 Data Validation Precautions

A serious concern with data validation is that a validation script sometimes is unable to recognize acceptable values, or that it may be non-functional depending on browser settings or other issues. If an error occurred when loading a script in a page, a customer might not be able to submit the order. In addition, it tends to make security managers forget that malicious users can intentionally disable scripts. Nothing from a form should ever be blindly trusted if a critical area of security might be compromised if an unexpected value was submitted.

For businesses there's another problem that is just as important as security. Customers shouldn't be prevented from placing orders due to a validation script which didn't consider all cases of valid input, or which commits some other logical error.

An example is Frys.com, where improper form validation prevents many people from submitting orders. Validation code tries to prevent duplicate order submissions in the final step of ordering. Here, a JavaScript logic error results in the form's being disabled before any order is submitted in many cases; on Opera, this problem is repeatable every time, so it's actually impossible to order from Frys.com with the Opera browser due to this bug.

So far as I know, this problem hasn't been corrected, and it's been present for more than a year. From a business standpoint, this is just as serious as security errors. Web sites should be lenient about permitting users to submit information which may in fact be correct, even when the JavaScript validation code thinks it's wrong. Business ordering systems should allow the user to submit the form after confirming their data for the third time. The JavaScript error code can be submitted in a hidden field.

Even though these validation functions have been refined over seven years and they can be considered to be production grade, one should still take extra

precautions to make sure that people can submit their data. There will still be cases when the validation script might be wrong or when there should still be a way around it.

2.2.4 Available validation functions

Each validation function sets an error code if there's an error, and it has to return an array of valid values (possibly with prettifications and reformatting) if there's no error. These validation functions were written (for the most part) before this validation script or article existed. They've been tested for years to make sure that the validation script is of production quality.

Note that the actual return values of all of them are arrays; it's possible for some to return more than one value in the array, such as with *email()* for email addresses. Also note that the return value might not be an array if there has been an error code set.

zip	Validation macro to define for a field for the zip values, either five digits or nine digits; as with other macros, it recognizes several variants, including spaces or no spaces, and with the case of a zip code, an optional hyphen. It is possible for multiple zip codes to be verified.
tel	Telephone numbers, including long distance and country codes, and all kinds of separators.
cardno	Checks any common type of credit card number.
text	Checks that text (this does not mean alphabetical characters only) is present.
words	Checks that at least two words of text are present.
email	Validates all kinds of email addresses. It is possible for multiple email addresses to be validated.
expires	checks a four-digit expiration date, that it has not expired, and that it is not an invalid amount into the future (using the computer's own clock, which is occasionally mis-set, resulting in errors, but this is OK, since the customers will be allowed to judge for themselves after receiving one warning).
url	Matches the wide range of possible URL values. Of course, it is possible that a URL may point to a non-existent location. It may return multiple values.

2.2.5 Using the Validation Functions

Let's show an example of what the *email()* function does. This code:

```
javascript:alert(email('hi@bye.com me@you.you'))
```

will result in the array

```
['hi@bye.com', 'me@you.you']
```

Now we explain how these are connected to a form.

Creating a Validation Variable

```
check_form_example = {};
```

Inside of this variable (which is an *object literal*) you place the validation macro function with the name of the appropriate form field, like this:

```
1check_form_example = {  
2zipcode: { verify: zip }  
3};
```

This would correspond to a text field in your form such as `<input type="text" name="zipcode">`.

2.2.6 Specifying Custom Messages

You can specify a custom error message if the zip code is invalid, which better fits the context of your form. (Maybe someone has a previous zip code and a new zip code.)

```
zipcode: { verify: zip, message: 'Enter the zip code of your new home.' }
```

Note: This message will always be shown, even if there are different error codes. For this reason you might not want to specify a custom error message directly, but you might decide to add your own error codes with new error code messages.

Also, you may feel comfortable overriding the default error message texts, which are within the reserved error code numbers 1-30.

2.2.7 Form Validation Command

In order to validate your form by this set of requirements, you execute the function `validate([form], check_form_example)`, which performs all the validation steps as specified. If the form fields within this variable, `check_form_example`, are all validated, the function returns a value of `true`. This allows it to be in the submission pre-check of a form directly:

```
<form onsubmit="return validate(this, check_form_example)">
```

In this context, `this` is the correct code which refers to the actual form (`this` refers to the element in which an event handler is placed; the element in which the event handler is placed is the form itself).

As you may know already, JavaScript defines that if the `onsubmit` event handler function returns a value of `false`, the form will not be submitted. If the `validate()` function has returned `false`, the form isn't submitted, and the user is able to correct the error of which they were alerted. If the `validate` function returns `true`, the form is submitted as normal.

2.2.8 Creating Custom Validation Functions

What do you do if your Web site only accepts Visa, MasterCard and Discover? The validation script will by default accept any type of credit card number, so you need to write a custom validation function.

Here's what you can do, noticing that Visa account numbers begin with 4, MasterCards begin with 5, and Discover cards begin with 6.

```
1// this code is copied from http://www.codelib.net/home/jkm/checksum.html
2
3function cardtypeOK(x) { // x is the number as text, like "123423..."
4  var a='456', i;
5  for (i=0; i<a.length; i++) if (x.charAt(0) == a.charAt(i)) break;
6  if (i == a.length) return false;
7  return true;
8}
```

In this example '456' are the beginning digits of credit cards your business can accept (in this case Visa, MC, and Discover). Note that a faster way of checking the digits is this:

```
if ('456'.indexOf(x.charAt(0)) != -1)
```

or even this

```
if (1 + '456'.indexOf(x.charAt(0)))
```

Then create a validation variable for the credit card information. This validation variable could be used to validate the submission of credit card information in the

same way that the previous validation variable was used to validate the submission of the form containing billing address information.

```
1 billing_info_check = {
2   card: { verify: function(s) {
3     s = cardno(s)
4     if (!err && !cardtypeOK(s[0]))
5       err = 32;
6     return [s];
7   } },
8   cardname: { verify: words,
9   message: 'Please enter the full name as it appears on the card.' },
10  expiry: { verify: expires }
11 }
```

You could also add the same attributes, `card`, `cardname`, `message`, `expiry` to the existing validation variable if your card information was part of the same form as the billing address.

2.2.9 Smart Validation Behavior

The validation script tries to avoid repeating the same error message twice. When a certain error code has been recorded for a certain field, the validation function won't show the same error message to the user. If the user changes the field but a different error message results, the user will be prompted with the new error message. You have the option of overriding this behavior by specifying the attribute `force` to have a value of 1. Three of the fields in the example form above have this behavior. Their validation parameters are specified like this:

```
1zip: { verify: zip, filter: 1, force: 1 },
2phone: { verify: tel, filter: 1, force:1 },
3email: { verify: email, filter: 1, force: 1 }
```

This was the idea of Greg Dietsche, because sometimes loops could result when evaluating optional fields:

"When validating against a list such as this one which has two optional fields, occasionally a loop can ensue if neither of the optional fields (in this case, email and subject) are filled in by the user..."

*To solve that problem, I made the following changes to the validation script. Basically the idea is that if the user has been shown a prompt for an *optional* field once, then they will never see the prompt again."*

This was a nice suggestion, so it was taken and added to. Now the error state is maintained for *each* field, and optional fields bring up an alert message if they have a possible error, and if the error number is different from the error that may have been "detected" before.

Essentially, this lets a user leave the value of the field as they wish, if they want to ignore the error message.

Notice also that there is a `filter` attribute. When this attribute is set, valid input will be reformatted to have a standard appearance. The script does its best to ensure that there is no data loss when reformatting input, so if part of a field is not recognized, it is left unaltered.

You can extend the filtering behavior by using functions like `String.toUpperCase()`. The value you return from a custom validation function will be used to reformat the data.

2.2.10 Technical Specifications of the Validation Function

This section provides specifications for the `validate()` function used as the `onsubmit` event handler.

The event handler is called with the following syntax

```
<form ... onsubmit="return validate(this, check)">
```

The variable `check` is the validation variable which has been described above.

Every element of the form with a *name* contained in `check` will be submitted to the function contained in `check[name].verify`, if such a function is present. The argument will be the value of the element if the element is text or textarea, otherwise the element object itself.

If an error code is set by this function `check[name].verify`, and if one of the following is true—the value of `check[name].force` is true, or the field has an error different from an error it may have had immediately prior to this invocation of validation—then `present_element` is set to this element, `check[name].err` and `present_error` are set to the error number, and any

error message is alerted, the element is focused, and false is returned from `validate()`. The browser will then prevent form submission.

Otherwise, after the end of the form is reached, true is returned from `validate`, and form submission is permitted.

Chapter 7

1. CGI

1.1 What is CGI?

This is an overview of what CGI is all about. It does not go into programming details, but will fill you in on the ideas behind it. In addition to the examples below CGI can be used for forums, polls, rotating banners, etc. Pretty much anything you want your system to do.

These days a lot of the programming that was done with CGI is now being done with *php*, but there are still about 40 million cgi scripts out there and a few things that only CGIs can do, so far. Just that legacy of those millions of old scripts means that its not going anywhere soon.

Obviously, when running scripts your unix hosting service has to be capable of *cgi hosting* otherwise you'll have problems. Most unix hosts *will* run cgi scripts so you'll find that cgi hosting and unix hosting are usually the same thing. Check your hosting service to make sure.

Article by Richard Lowe Jr.

Let's unlock a little bit of the mystery about something called CGI. If it helps any, CGI means Common Gateway Interface. This is a method which is used to exchange data between the server (the hardware and software that actually allows you to get to your web site) and a web client (your browser). CGI is actually a set of standards where a program or script (a series of commands) can send data back to the web server where it can be processed.

Typically, you use standard HTML tags to get data from a person, then pass that data to a CGI routine. The CGI routine then performs some action with the data.

Method	Where is the script code located ?	Where is the script code executed ?	Required HTML file extensions
CGI	In files in the CGI-BIN directory on the server.	On the server.	.shhtm or .shtml
PHP, ColdFusion, ASP	Embedded in the HTML document.	On the server.	.php .cfm .asp
Javascript	Embedded in the HTML document.	On the user's PC by their browser.	n/a
Java	In files on the server.	On the user's PC by their browser.	n/a

Figure 1: CGI and other languages

some of the more common uses of CGI include:

- Guestbooks - The CGI routine is responsible for accepting the data, ensuring it is valid, sending an email acknowledgement back to the writer, perhaps sending an email to the webmaster, and creating the guestbook entry itself.
- Email Forms - A simple CGI forms routine just formats the data into an email and sends it back to the webmaster. More complicated routines can maintain a database, send an acknowledgement and validate data.
- Mailing List Maintenance - These routines allow visitors to subscribe and unsubscribe from a mailing list. In this case, the CGI routine maintains a database of email addresses, and the better ones send acknowledgements back to the visitor and webmaster.

A CGI routine can be anything which understands the CGI standard. A popular CGI language is called PERL, which is simple to understand and use (well, compared to other languages). PERL is a scripting language, which means each time a PERL routine is executed the web server must examine the PERL commands to determine what to do. In contrast, a compiled language such as C++ or Visual Basic can be directly executed, which is faster and more efficient.

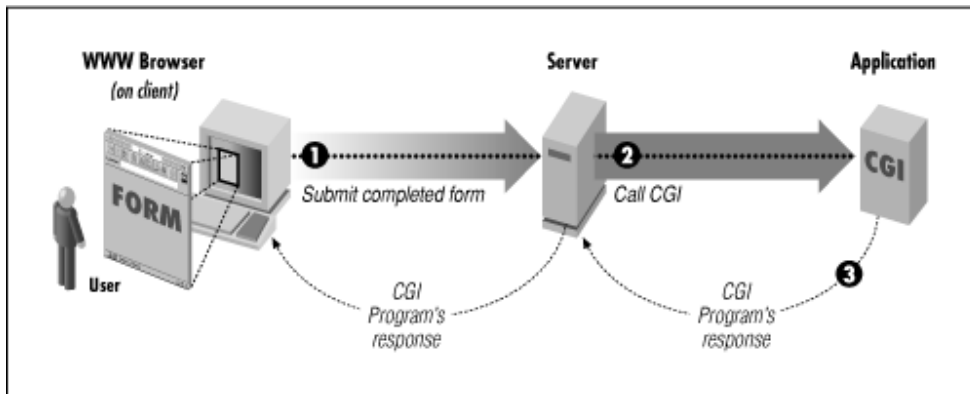


Figure2: Simple diagram of CGI

1.2 How CGI works

1. You (the webmaster) specify a form tag which includes the name of the CGI routine.
2. You create HTML tags which retrieves data from your visitors.
3. Each of the input tags includes a variable name. The data which is retrieved from the visitor (or directly set if the tag includes the "hidden" qualifier) is placed in the variable name.
4. When the visitor presses the "submit" button, the CGI routine which was specified in the form tag is executed. At this time, the CGI routine "takes control", meaning the browser essentially is waiting for it to complete.
5. This CGI routine can get data from variable names. It retrieves the data and does whatever action is required.
6. When the CGI routine finishes, it returns control back to the web client (the browser).

1.3 Example web using CGI languages used for CGI

Example 1: Program that sends an HTML page tailored to the type of browser.

```
#include "cgi-lib.h"
#include "html-lib.h"
int main()
{
  if (accept_image())
    show_html_page("/index-img.html");
  else
    show_html_page("/index-txt.html");
}
```

Example 2: A sample form in HTML that uses the CGI program query results.

```
<form method=POST action="/cgi-bin/query-results">
<p>Name: <input type=text name="name">
<p>Age: <input type=text name="age">
<p>E-mail: <input type=text name="email">
<p><input type=submit>
</form>
```

Example 3

2. Form, GET and POST

There are two ways to send data from a web form to a CGI program: GET and POST. These *methods* determine how the form data is sent to the server.

With the GET method, the input values from the form are sent as part of the URL and saved in the QUERY_STRING environment variable. With the POST method, data is sent as an input stream to the program. We'll cover POST in the next chapter, but for now, let's look at GET.

You can set the QUERY_STRING value in a number of ways. For example, here are a number of direct links to the env.cgi program:

Try opening each of these in your web browser. Notice that the value for QUERY_STRING is set to whatever appears after the question mark in the URL itself. In the above examples, it's set to "test1", "test2", and "test3" respectively.

You can also process simple forms using the GET method. Start a new HTML document called envform.html, and enter this form:

Program 1: envform.html - Simple HTML Form Using GET

```
<html><head><title>Test Form</title></head>
<body>

<form action="env.cgi" method="GET">
Enter some text here:
<input type="text" name="sample_text" size=30>
<input type="submit"><p>
</form>

</body></html>
```

Working example: <http://www.cgi101.com/book/ch3/envform.html>

Save the form and upload it to your website. Remember you may need to change the path to env.cgi depending on your server; if your CGI programs live in a "cgi-bin" directory then you should use action="cgi-bin/env.cgi".

Bring up the form in your browser, then type something into the input field and hit return. You'll notice that the value for QUERY_STRING now looks like this:

```
sample_text=whatever+you+typed
```

The string to the left of the equals sign is the name of the form field. The string to the right is whatever you typed into the input box. Notice that any spaces in the string you typed have been replaced with a +. Similarly, various punctuation and other special non-alphanumeric characters have been replaced with a %-code. This is called *URL-encoding*, and it happens with data submitted through either GET or POST methods.

You can send multiple input data values with GET:

```
<form action="env.cgi" method="GET">
First Name: <input type="text" name="fname"
size=30><p>
Last Name: <input type="text" name="lname"
size=30><p>
<input type="submit">
</form>
```

This will be passed to the env.cgi program as follows:

```
$ENV{QUERY_STRING} = "fname=joe&lname=smith"
```

The two form values are separated by an ampersand (&). You can divide the query string with Perl's `split` function:

```
my @values = split(/&/,$ENV{QUERY_STRING});
```

`split` lets you break up a string into a list of strings, splitting on a specific character. In this case, we've split on the "&" character. This gives us an array named `@values` containing two elements: ("fname=joe", "lname=smith"). We can further split each string on the "=" character using a `foreach` loop:

```
foreach my $i (@values) {
    my($fieldname, $data) = split(//=, $i);
    print "$fieldname = $data<br>\n";
}
```

```
}
```

This prints out the field names and the data entered into each field in the form. It does not do URL-decoding, however. A better way to parse QUERY_STRING variables is with CGI.pm.

3. PHP

3.1 PHP Introduction

PHP is a server-side scripting language.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML/XHTML
- JavaScript

What is PHP?

- PHP stands for **PHP: Hypertext Preprocessor**
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

-
-
- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support

3.2 Php Syntax

PHP code is executed on the server, and the plain HTML result is sent to the browser.

Basic PHP Syntax

A PHP scripting block always starts with `<?php` and ends with `?>`. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with `<?` and end with `?>`.

For maximum compatibility, we recommend that you use the standard form (`<?php`) rather than the shorthand form.

```
<?php
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>
```

```
<?php
echo "Hello World";
?>
```

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

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

Note: The file must have a .php extension. If the file has a .html extension, the PHP code will not be executed.

Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

3.3 PHP MySQL Connect to a Database

The free MySQL database is very often used with PHP.

Create a Connection to a MySQL Database

Before you can access data in a database, you must create a connection to the database.

In PHP, this is done with the `mysql_connect()` function.

Syntax

```
mysql_connect(servername,username,password);
```

Parameter	Description
servername	Optional. Specifies the server to connect to. Default value is "localhost:3306"
username	Optional. Specifies the username to log in with. Default value is the name of the user that owns the server process
password	Optional. Specifies the password to log in with. Default is ""

Note: There are more available parameters, but the ones listed above are the most important. Visit our full [PHP MySQL Reference](#) for more details.

Example

In the following example we store the connection in a variable (`$con`) for later use in the script. The "die" part will be executed if the connection fails:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

// some code
?>
```

Closing a Connection

The connection will be closed automatically when the script ends. To close the connection before, use the `mysql_close()` function:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

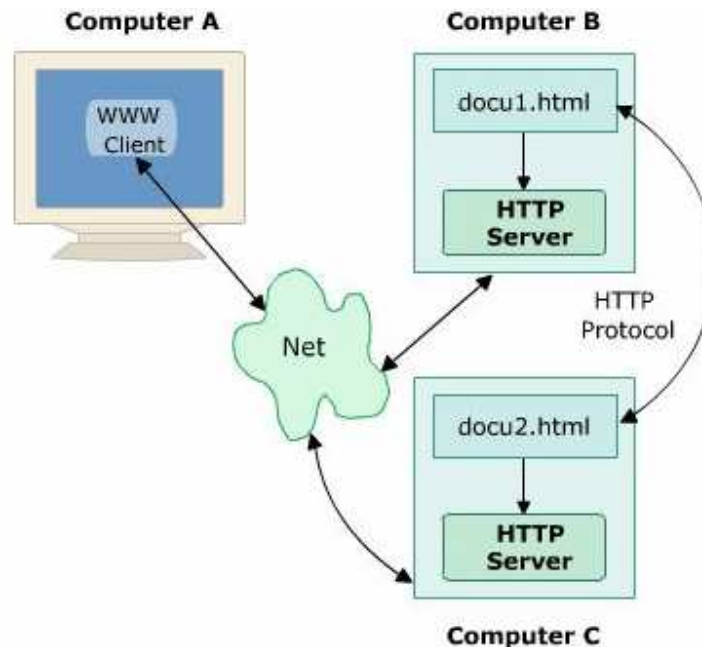
// some code
```

```
mysql_close($con);  
?>
```

Chapter 8.

1.SESSION MANAGEMENT

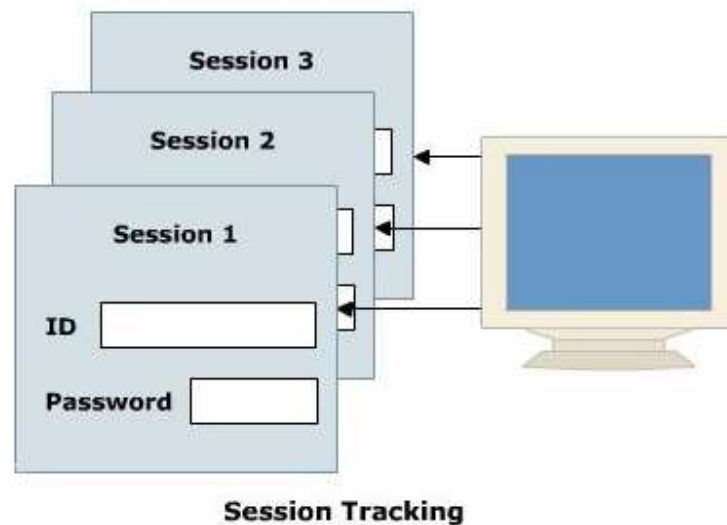
1.1.Stateless Nature of HTTP



Client Server Model using HTTP Protocol

- A protocol is set of rules, which govern the syntax, semantic, and synchronisation of communication.
 - When the configuration setting, transactions and information are not tracked by a protocol, then it's called a stateless protocol. For such a protocol, connections last for only one transaction.
 - HTTP is based on the Client-Server Model. An HTTP Client such as a Web Browser, opens the connection and sends a request message to an HTTP server asking for resources. The server responds with the message and the requested resources. Once the resource is delivered, the server closes the connection. Thus, no connection information is stored, and hence HTTP is referred to as a stateless protocol.
 - Advantage. Hosts do not need to retain information about users between requests. This simplifies server design because it does not need to dynamically allocate storage to manage conversations in progress or worry about freeing it when a client connection dies in mid-transaction.
 - Disadvantage. It may be necessary to include more information in each request; this extra information will need to be interpreted by the server each time. And another, no acknowledgement about information has reached the client. Hence the loss of data, if any, is not known.
-
-

1.2. Need of Tracking Client Identity and State



- In online shop site, items are displayed in different pages based on classification such as stationary, hardware, music, and books. When a customer does online shopping, he may select items from various pages and put them in a shopping cart.
- As HTTP is a stateless protocol, when the customer clicks on a new page, the information about the previously selected items is lost. As a result, it is necessary to keep track of successive requests made by the same user.
- The server should have a tracking mechanism that allows the customer to maintain the information with the server as long as the customer does not log out from the website.

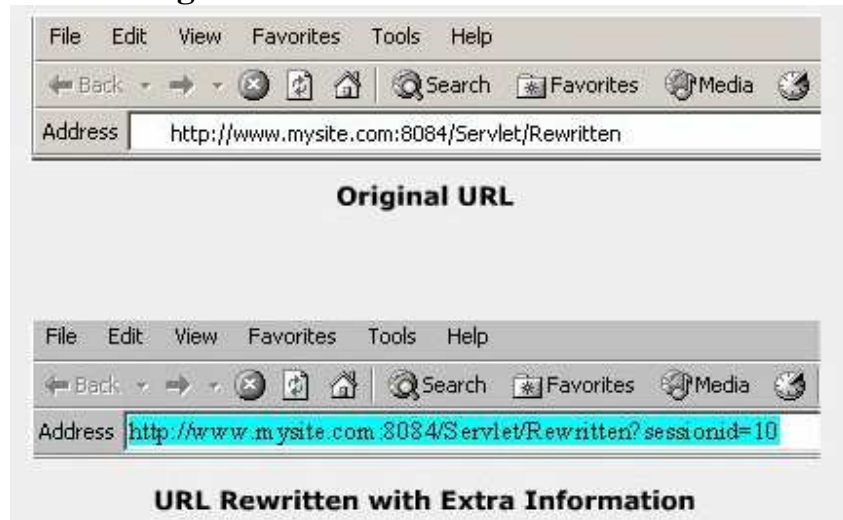
1.3. Session Concept

- **Session** is the time period between the start and the end of the user's interaction with an application.
 - **Session variables** are used to store information about the user's session. This session is available to all pages in the application. Generally, the information stored in session variables is username, password, user's preferences, and so on. Session variables are cleared as soon as the user's session in the site comes to an end.
 - **Session identifier** is used to identify a session because it is unique. When a session state is enabled, each request for a page in the application is examined for the session ID value. If the session ID value is not supplied, a
-
-

new session is start. The session ID is then sent for that session to the browser along with response.

1.4. Some of Session Tracking Techniques

1.4.1. URL Rewriting



- Url Rewriting technique add some extra data at the end of URL to identify the session.
- The extra information can be in the form of extra path information or add parameters. The user does not see extra information on the surface as such but he/she clicks on the link, he/she not ask for resource but because of information after the “?” in URL, he or she is actually sending data to the program. Generally, the extra information appended is the unique session ID. Tracking can be done by retrieveing the session ID

1.4.2. Hidden Form Field

```

</Body>
<H1> How to use a Hidden form fields </H1>
<FORM ACTION="SettingHiddenFields.jsp" METHOD="post">
  Please Enter your First name : <input type ="text"
  name = "Firstname" value = "">
  <input type="hidden" name="hidden" value="You are most Welcome">
  <input type="submit" value="submit">
</FORM>
</Body>

```

Hidden Form Fields

```

<H1>Reading Hidden Controls</H1>
String string = request.getParameter("name");
String text = "";
if(request.getParameter("text1") != null) {
  out.println(string + "The hidden text is:" +request.getParameter("text1"));
  text = request.getParameter("text1");
}
<FORM ACTION="GettinHiddenField.jsp" METHOD="post">

```

JSP Code Accessing Hidden Form Fields

- The hidden form field method inserts the session identifier into the hidden form field in html of each page. This browser includes hidden field when the form is submitted. The session can be extracted by application by searching for these field.
- Hidden field present nothing on HTML page and the information in invisible to the user. The hidden field can be hold any kind of data. Every time the server reads, it should read all the parameter pass to it from the previous form and all the value as new hidden field in any new form that is generates. In the way, information is passed to another thereby maintain the connection between two pages.

1.4.3.Cookies (See more on Chap2)

- Cookie is a small piece of information sent by server to the client web browser that can be later read back by server. A cookie contain one or more name-value pair, which exchange in request and response header.

Example Code 1. Session in PHP

```

<?php
  session_start();
  $session_id = session_id();
  echo 'Your session id is '.$session_id.'<br>';
  if (isset($_SESSION['visit']))

```

```
    {
        $_SESSION['visit']++;
    }
    else
    {
        $_SESSION['visit']=1;
    }
echo 'You have visited this page ' .$_SESSION['visit'] . ' times';
?>
```

Example Code 2 Session in ASP.NET

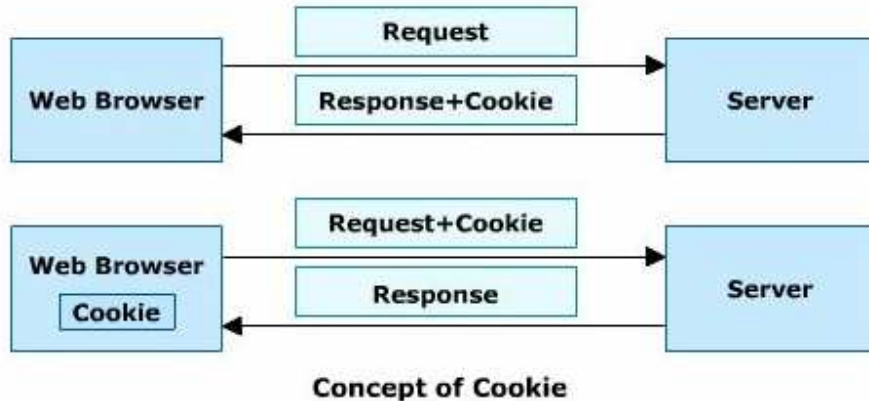
```
void Session_Start(object sender, EventArgs e)
{
    // Code that runs when a new session is started
    Session["visit"] = 0;
    Session.Timeout = 1500;
    Response.Write("Session ID is:" + Session.SessionID+"\n");
}
protected void Page_Load(object sender, EventArgs e)
{
    Session["visit"]=Convert.ToInt16(Session["visit"])+1;
    Response.Write("Hello, you come here"+Session["visit"]);
}
```

Example Code 3 Session in JSP

```
public void doGet(HttpServletRequest request,
HttpServletRequest response) throws ServletException,
IOException
{
//If the user want to add item, remember it by
//adding a cookie
    if (value!=null)
    {
        itemID=value[0];
        Cookie getItem=new Cookie("Buy",itemID);
        getItem.setComment("User has indicated a desire"+
        "to buy this book from book store");
        response.addCookie(getItem);
    }
}
```

2.COOKIE

2.1.Cookie Overview



- In [computing](#), a cookie (also browser cookie, computer cookie, tracking cookie, web cookie, internet cookie, and HTTP cookie) is a small string of text stored on a user's computer by a [web browser](#). A cookie consists of one or more name-value pairs containing bits of information such as user preferences, shopping cart contents, the identifier for a server-based [session](#), or other data used by websites.
- It is sent as an HTTP header by a web server to a web client (usually a browser) and then sent back unchanged by client each time it accesses that server. A cookie can be used for authenticating, session tracking (state maintenance), and maintaining specific information about users, such as site preferences or the contents of their electronic shopping carts. The term "cookie" is derived from "magic cookie", a well-known concept in UNIX computing which inspired both the idea and the name of browser cookies. Some alternatives to cookies exist, each has its own uses, advantages, and drawbacks.

2.2.Cookie function

- Normal cookies are used to remember the user who is visiting the website in order to show the appropriate content. Without them, some websites would cease to function. Cookies are also used to remember the "signed on" status of users.
 - HTTP cookies are used by Web servers to differentiate users and to maintain data related to the user during navigation, possibly across multiple visits. HTTP cookies were introduced to provide a way to implement a "shopping cart" (or "shopping basket"), a virtual device into which a user can store items they want to purchase as they navigate the site.
 - Allowing users to log in to a website is another use of cookies. Users typically log in by inserting their credentials into a login page; cookies
-
-

allow the server to know that the user is already authenticated, and therefore is allowed to access services or perform operations that are restricted to a user who is not logged in.

- Many websites also use cookies for personalization based on users' preferences. Sites that require authentication often use this feature, although it is also present on sites not requiring authentication. Personalization includes presentation and functionality. For example, the Wikipedia website allows authenticated users to choose the webpage skin they like best; the Google search engine allows users (even non-registered ones) to decide how many search results per page they want to see.

2.3.Type of Cookie

2.3.1.Session Cookies

- A session cookies is also referred to as a transient cookie. Session cookies are store temporarily in the memory. Once the browser is closed, the session cookie can not be retained. The next time, when the same visitor visit the same site, he or she will be traded as a new visitor.
- Instead of collecting information from the user's computer, session cookies are generally used by those Web Applications in which user need to be indentified as they move from page to page. For example, If you are a member of online book library, once you have logged in, you can browse through any number of books.

2.3.2.“Persistent” Cookies – Permenant Cookie- Stored Cookie

- Consider a login page of any popular e-mail service provider such as Hotmail or Yahoo Mail. Once you have enter your username and password, you are prompted to indicate if you want to your login details to be remembered in the computer. If you say yes, then their details are stored in the cookie. The next time you start login your details automatically appear in their respective places. The Persistent Cookies use to track user's browsing habits. They have an expiry date, and store in hard disk until expiry date or until they manually deleted by user.

Example code 4 “Persistent” Cookies in PHP

```
<?php
if ($_POST['ok']=='OK')
{
    setcookie('user',$_POST['user'],time()+20);
    setcookie('age',$_POST['age'],time()+20);
}
if (isset($_COOKIE['user']))
```

```
{
    echo 'Welcome back, '.$_COOKIE['user'].'</br>';
    echo 'Your age is '.$_COOKIE['age'].'</br>';
}
?>
```

3.SERVLET

3.1.Static Web vs Dynamic Web

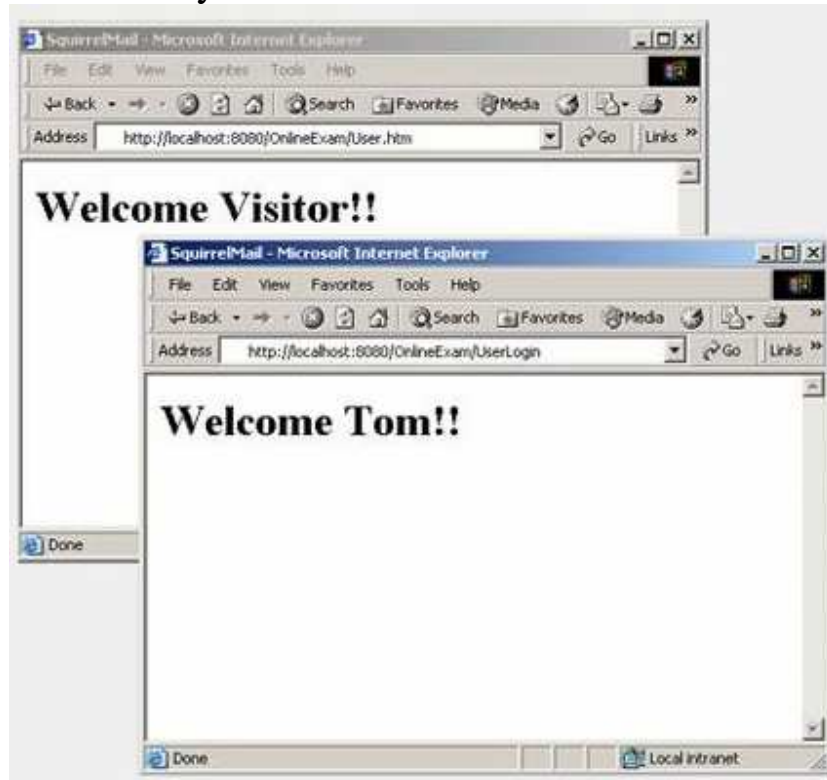
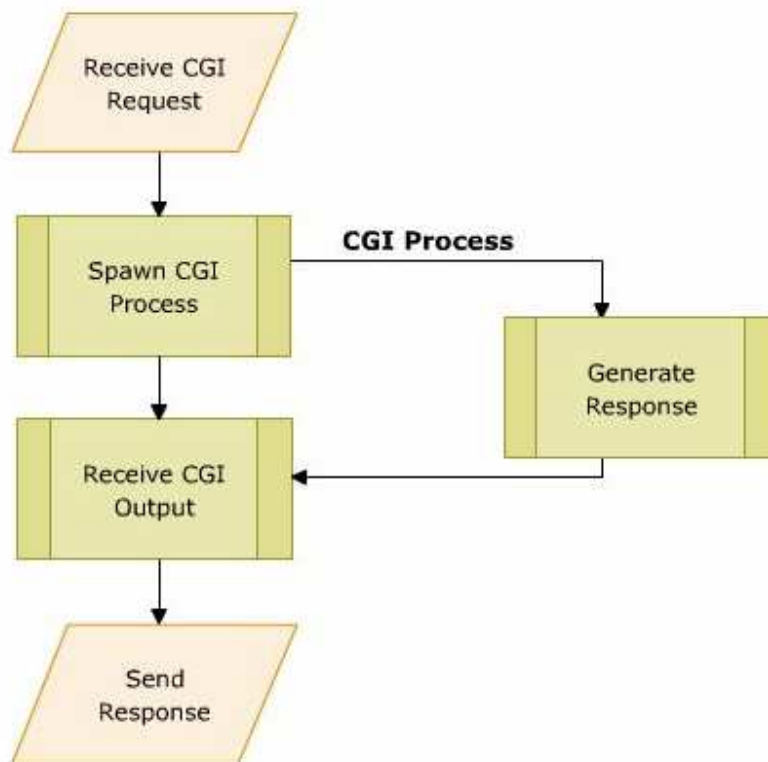


Figure 3.1 Dynamic vs Static

- **Static Web** contents can be designed and maintained explicitly by the Web Developers. It's not feasible to maintain the Static Web content from User-End. Any subsequent change requires to be done by the Web Designers. Information such as username can not be retained by the browser. Only general information is shown in the subsequent pages.
- **Dynamic Web** content can be maintained by Client-end and changes which are implemented can be managed and modified by the client. Once Dynamic Web is created it requires very less or no maintenance cost since there is no requirement of approaching the web designer for making change. User specific information can be retained by the browser and displayed to all the subsequent pages.

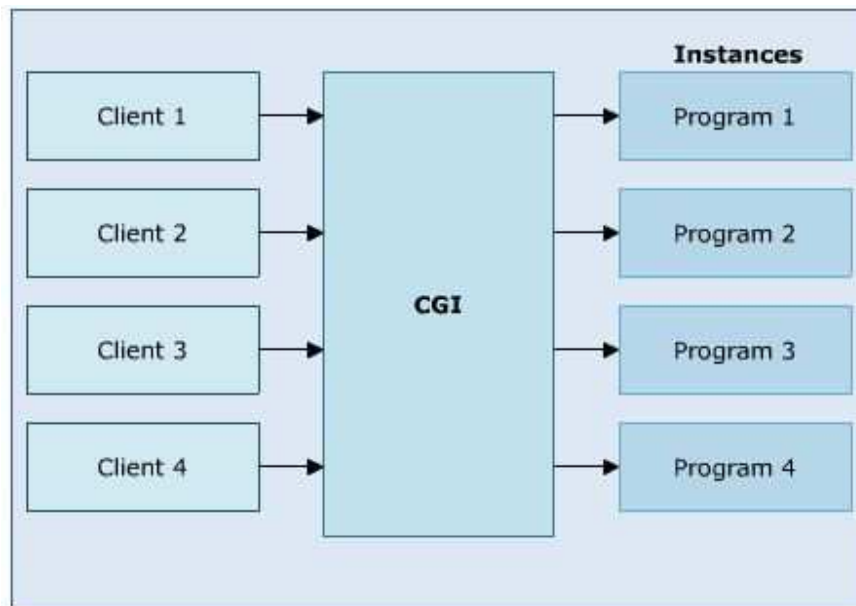
3.2.CGI-Solution for dynamic content generation.



Server Process for running CGI

- CGI is set of standards followed to interface application form client-side to the web server. The page viewed at the client-end is simple HTML page containing the static content.
- A program that gives dynamic output thereby executing in real time is written at server-side. To write programs in the server side using standard CGI is used.

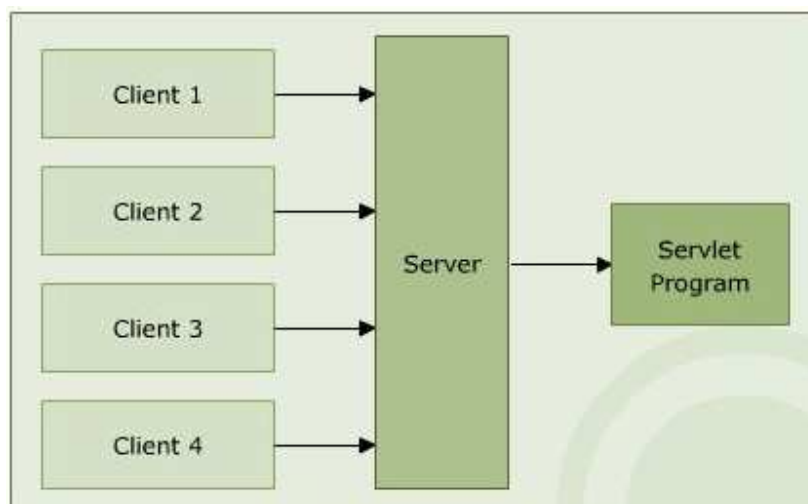
3.3.Shortcomings of CGI



CGI process

- Number of process can be executed by a Web Server is limit to each server. Each time is process is executed, different instance of same processes is created in the server side there by resulting in **overload of server** and in **reducing efficiency of server**.
- The widely accepted platform for writing CGI script is Perl. Each time the server receive a request, the Perl interpreter need to be reloaded. This reduce the efficiency of server.

3.4.Servlet- A Solution to CGI-Problem.



- Servlet are java codes which you to add dynamic content to Web Server. The content generate after excution of servlet is basically HTML page. Servlet gets auto on refresh on receiving a request each time. A servlet

initialisation code is used only for initialising the servlet for the first time, there by proving advantageous over CGI.

- The multi threading property of servlets help in handing separate requests by allocating resource to separate threads. This give boost to the performance thereby increasing the efficiency of servlets

3.5.Merits and Demerits of Servlet.

Advantages	Disadvantages
Enhanced Efficiency	Low-level HTML Documentation
Ease of Use	Unclear Session Management
Powerful	
Portable	
Safe and Cheap	

3.6.Web container Concept

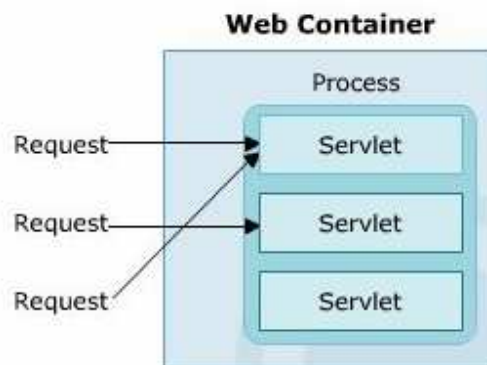


Figure 3.2 Web Container

The web container is the program that manage excution of servlet and JSP page. The web container take request for Web Server and pass it to Servlet for processing. It maintain the Servlet life cycle. The performance of a servlet upon the efficiency of Web container.

3.7.Servlet's Life Cycle

- The Servlet class is loaded by the container during start-up.
 - The container calls the `init()` method. This method initializes the servlet and must be called before the servlet can service any requests. In the entire life of a servlet, the `init()` method is called only once.
 - After initialization, the servlet can service client requests. Each request is serviced in its own separate thread. The container calls the `service()` method of the servlet for every request. The `service()` method determines the kind of request being made and dispatches it to an appropriate method to handle
-
-

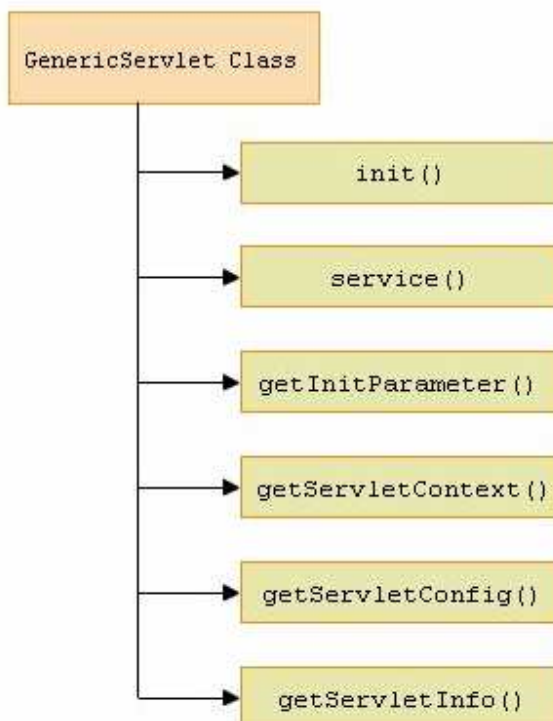
the request. The developer of the servlet must provide an implementation for these methods. If a request for a method that is not implemented by the servlet is made, the method of the parent class is called, typically resulting in an error being returned to the requester.

- Finally, the container calls the `destroy()` method which takes the servlet out of service. The `destroy()` method like `init()` is called only once in the lifecycle of a Servlet.

3.8.HTTP Request Processing Life Cycle

- Receive: The servlet instance is receive.
- Include: The instance variable is set to the request object.
- Authenticate: The servlet is authenticate against the request object.
- Pre-process: Require pre-process is done with Servlet.
- Generate Response: Response is generate after pre-process.
- Post-process: Require post-process is done.
- Reuseability check: Servlet is check for reuseability and if required held back.
- Replace: Servlet is replaced in the servlet manager.

3.9.Implement Servlet in Java



`init()`: Initialises the servlet.

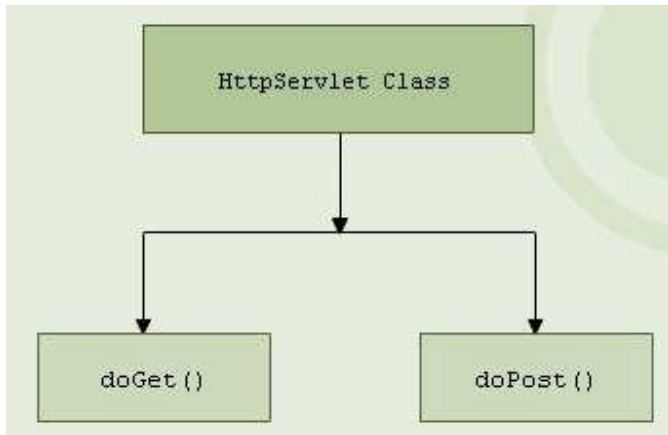
`service()`: Call the container to response the servlet request.

`getInitParameter()`: return the string contain the value of named initialisation parameter.

`getServletContext()`: return the Servlet context in which this servlet instance is running.

`getServletConfig()`: return the servlet Configuration of servlet instance.

`getServletInfo()`: return useful servlet instance such as name of creator, version, copyright.



Extend GenericServlet, must override atleast one of the following object. DoGet(), DoPost(), DoDelete(), init(), destroy() and getServletInfo().

doGet(): Call by the server to handle the get request made by client. This method is call through service().

doPost(): Call by the server to handle the post request made by client.

Example code of Servlet

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \" +
            \"Transitional//EN\">\n" +
            "<html>\n" +
            "<head><title>Hello WWW</title></head>\n" +
            "<body>\n" +
            "<h1>Hello WWW</h1>\n" +
            "</body></html>");
    }
}
```

4.JAVA SERVER PAGES

4.1.What is a JSP Page

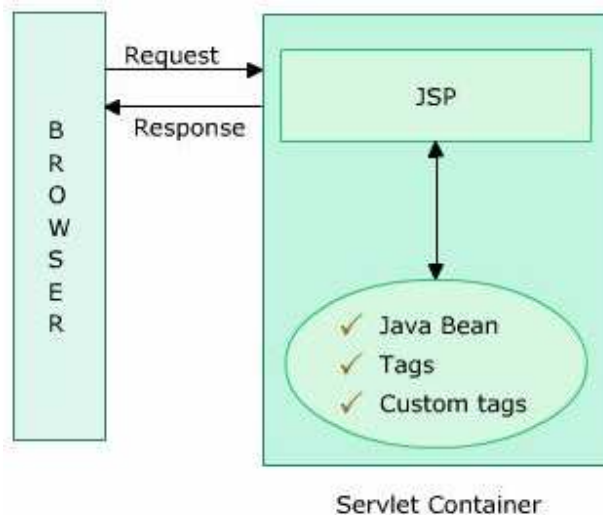


Figure 4.1 JSP Page

- A JSP Page contain HTML tags, which is display the static content on the Web Pages. Also contain tag which use to generate dynamic content. The tag is categorised into standard tags and custom tags.
- The standard JSP tags are used for invoking the operation on JavaBean component and processing requests.
- The custom JSP tags perform operations, such as processing forms, accessing database and other Enterprise services, such as email and direction.
- The information given in the tags in a JSP page is processed and will generate the dynamic content on the Web Page.

4.2.Benefits of JSP

- **JSP separate content generation from presentation** logic in JSP page and content logic on server in JavaBean.
 - **Emphasising reusable componet.** JSP page use reusable components such as JavaBean Program, which can be use by multiple programs. Java bean enable to perform complex functions in JSP page. These beans can be shared and exchanged among web proqramers.
 - **Simplified page development.** Allow a Web proqramer with limited knowledge in scripting language to design a page. Proqramer can also generate dynamic content using standard tags provided by JSP.
-
-

-
-
- **Access and Instantiate JavaBeans components.** Support using JavaBeans components with JSP. Can easily create and initialise beans and get/set values of this.

4.3. Servlet and JSP

- JSP technology simplifies the process of creating page by separating the web presentation from web content. In most of applications, the response send to client is a combination of template-data and dynamic generated data. In this situation, it much easy to work with JSP pages than to do every with Servlet.
- Servlet are well-suited for handling binary data dynamically, for example for upload file or for creating dynamic image, since they need not contain any display logic.

4.4. Architecture of JSP

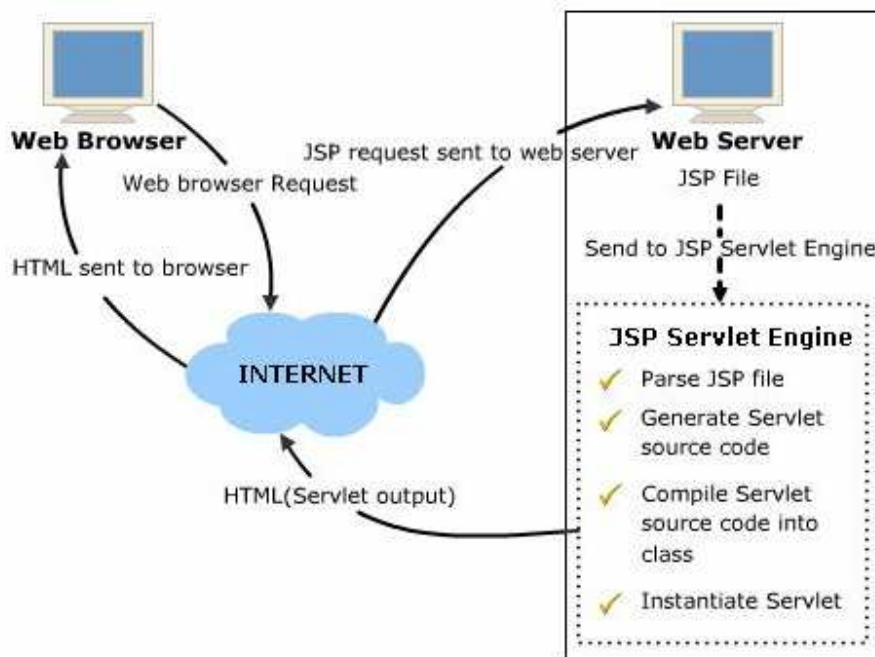
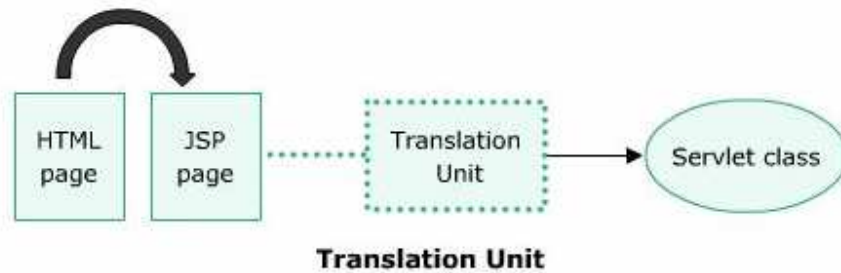


Figure 4.2 Architecture of JSP

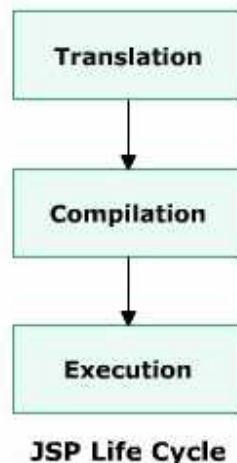
- Whenever the web browser sends a request for JSP page to Web Server through the internet, the Web Server pass JSP file to JSP Servlet Engine.
 - The JSP file is parsed and a servlet is generated from JSP file. This servlet source code is compiled into class and is instantiated by the init() and service() methods. The HTML output from the servlet is send through the Internet and HTML results are displayed on the user's Web Browser.
-
-

4.5. Translation Unit



- A JSP can include the content of other HTML pages or other JSP files. The translation unit process this JSP page and convert it to Servlet class. The translation unit consists of JSP source file as well as all of its static include files.

4.6. JSP Life Cycle



- Translation is the first phase of JSP Life Cycle. In this phase a servlet code to implement JSP tag is automatic generated, compiled and load into servlet container. There are three methods are used in this phase.
 - `jspInit()` : This method is Invoked when the JSP page is initialise. A JSP page can override this method by include a definition for it in declaration elements. A JSP page should always redefine the `init()` method from a servlet.
 - `_jspService()`: this method conresponse to the body of JSP page. This method is defined automatically by JSP container and never be defined by JSP page.
 - `jspDestroy()`: this method is invoked when JSP page is going to be destroyed.
-
-

-
-
- Compilation, the servlet class require to compile the entire JSP is generated. If any change are made in the code to generate dynamic content, then the JSP page need to be compiled and execute again.
 - Execution of page is carried out with the help of directives. Page directives controls various excution parameters and are used for buffering output and handling errors.

4.7.JSP Scripting elements

4.7.1.Scriptlets

- A scriptlets use to embeded Java code within HTML code. The java code is insert into Servlet. The scriptlets are executed when the request of client is being processed. The scriptlets are used to add complex data to HTML form.

Syntax: <% Java code fragment %>

Snippet:

```
<% int localStackBasedVariable = 1;
out.println(localStackBasedVariable); %>
```

4.7.2.Expression tag

- An **expression tag** places an expression to be evaluated inside the java servlet class. Expressions should not be terminated with a semi-colon.

Syntax: <%= Java Expression %>

Snippet: <%= "expanded inline data " + 1 %>

4.7.3.Declaration tag

- A **declaration tag** places a variable definition inside the body of the java servlet class. Static data members may be defined as well. Also inner classes should be defined here.

Syntax: <%= Java Declartion code %>

Snippet:

```
<%! int serverInstanceVariable = 1; %>
<%!
/**
```

```
* Converts the Object into a string or if
* the Object is null, it returns the empty string.
*/
public String toStringOrBlank( Object obj ){
    if(obj != null){
        return obj.toString();
    }
    return "";
}
%>
```

4.7.4. Comments

- We can use the following tag to give **comments** in jsp.

Syntax: <%-- Give your comments here --%>

Snippet: <%-- My comment --%>

4.8. JSP Directives

- JSP Directives control the processing of entire page. The directives identify the packages to be imported and the interface to be implemented. However, these directives do not procedure any output. They inform the JSP engine about the actions to be performed on the JSP page.

Syntax: <%@ directiveName attribute="value" %>

4.8.1. Page Directives

- The page directives used to define and manipulate a number of importance attributes that affect the entire of JSP pages. A page directive is written at the beginning of JSP pages.
 - A JSP pages can contain any number of page directives. All directives in the page is processed together during translation and result is applied together to the JSP page.
 - **import:** Results in a Java import statement being inserted into the resulting file.
 - **contentType:** specifies the content that is generated. This should be used if HTML is not used or if the character set is not the default [character set](#).
 - **errorPage:** Indicates the page that will be shown if an exception occurs while processing the HTTP request.
 - **isErrorPage:** If set to true, it indicates that this is the error page. Default value is *false*.
 - **isThreadSafe:** Indicates if the resulting servlet is [thread](#) safe.
-
-

-
-
- **autoFlush:** To autoflush the contents. A value of true, the default, indicates that the buffer should be flushed when it is full. A value of false, rarely used, indicates that an exception should be thrown when the buffer overflows. A value of false is illegal when also using buffer="none".
 - **session:** To maintain session. A value of true (the default) indicates that the predefined variable session (of type HttpSession) should be bound to the existing session if one exists, otherwise a new session should be created and bound to it. A value of false indicates that no sessions will be used, and attempts to access the variable session will result in errors at the time the JSP page is translated into a servlet.
 - **buffer:**To set Buffer Size. The default is 8k and it is advisable that you increase it.
 - **isELIgnored:** Defines whether EL expressions are ignored when the JSP is translated.
 - **language:** Defines the scripting language used in scriptlets, expressions and declarations. Right now, the only possible value is "java".
 - **extends:** Defines the superclass of the class this JSP will become. You won't use this unless you REALLY know what you're doing - it overrides the class hierarchy provided by the Container.
 - **info:** Defines a String that gets put into the translated page, just so that you can get it using the generated servlet's inherited getServletInfo() method.
 - **pageEncoding:** Defines the character encoding for the JSP. The default is "ISO-8859-1"(unless the contentType attribute already defines a character encoding, or the page uses XML document syntax).

Syntax: <% @ directive attribute="value" %>

Snippet:

```
<% @ page import="java.util.*" %> <%-- example import --%>
<% @ page contentType="text/html" %> <%-- example contentType --%>
<% @ page isErrorPage="false" %> <%-- example for non error page --%>
<% @ page isThreadSafe="true" %> <%-- example for a thread safe JSP --%>
<% @ page session="true" %> <%-- example for using session binding --%>
<% @ page autoFlush="true" %> <%-- example for setting autoFlush --%>
<% @ page buffer="20kb" %> <%-- example for setting Buffer Size --%>
```

4.8.2.Include Directives

- Used to insert content of another resource in JSP page at runtime. The remote resource is a file in textbase format such as text, HTML, JSP. The
-
-

remote resource should be given with the proper relative file path in the include directive.

- At the translation phase the content of include file is parsed by JSP Engine and content or output of that included file is inserted in the current web document.

Syntax: `<%@ include file="value" %>`

Snippet: `<%@ include file="somefile.jspf" %>`

4.8.3.TagLib Directive

- The taglib allow JSP pages to create custom tags, which are defined by user. Custom tags have access to all the object that are available for JSP page.
- A Tag Library is a group of custom tags that are extend the functionality of a JSP page one after the other. The taglib directive specifies name of tag library, which contains compiled Java code for the tag to be used. Using this tag library, the JSP Engine determines what action is to be taken when a particular tag appears in JSP page.

Syntax: `<%@ taglib prefix="tagprefix" uri="taglibraryURI" %>`

Snippet: `<%@ taglib prefix="myprefix" uri="taglib/mytag.tld" %>`

4.9.JSP Action

- JSP Actions allow the transfer of control between pages. You can dynamic insert a file, reuse JavaBean component, forward the user to another page or generate HTML for Java Plugin.
- **<jsp:include>** give you a choice to include either a static or dynamic file in a JSP file at the time of page request. The result a different for each type of inclusion.
- The content is included in the call JSP file if the file is static, otherwise it act on request and send back a result that is included in the JSP page.

Syntax: `<jsp:include page="weburl" flush="true"/>`

Snippet:

```
<html>
  <head></head>
  <body>
    <jsp:include page="mycommon.jsp" >
      <jsp:param name="extraparam" value="myvalue" />
    </jsp:include>
    name:<%=request.getParameter("extraparam")%>
```

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

- **<jsp:forward>** use to redirect the request object containing the client-request from one JSP file to another file. The target file can be an HTML file, a JSP file, or a servlet.
- In the source the code after `<jsp:forward>` element is not processed. To pass parameter names and values to target file, we can use `<jsp:param>`.

Syntax: `<jsp:forward page="url" />`

Snippet:

```
<jsp:forward page="subpage.jsp" >
  <jsp:param name="forwardedFrom" value="this.jsp" />
</jsp:forward>
```

- **<jsp:param>** allows you to pass one or more name value pairs as parameter to an include or forward resources like JSP page, servlet or other resource that can process parameter.
- **<jsp:plugin>** in the client Web Browser, the `<jsp:plugin>` play or displays an object, using Java plugin, that are available in the browser or downloaded from a specified URL.

Syntax:

```
<jsp:plugin type="applet|bean" height="%" width="%"
codebase="classFileDirectory"
code="classFileName" />
```

Snippet:

```
<jsp:plugin type=applet height="100%" width="100%"
archive="myjarfile.jar,myotherjar.jar"
codebase="/applets"
code="com.foo.MyApplet" >
<jsp:params>
<jsp:param name="enableDebug" value="true" />
</jsp:params>
<jsp:fallback>
Your browser does not support applets.
</jsp:fallback>
</jsp:plugin>
```

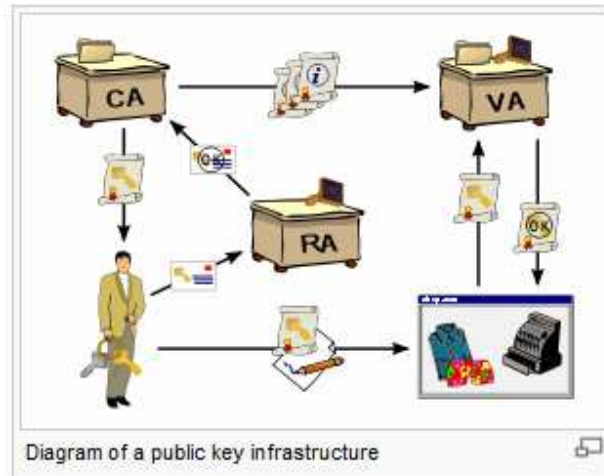
-
-
- **<jsp:fallback>** this text message conveys the user that plug-in could not start known as fall back. If the plugin starts, but the applet or bean does not, the plugin usually display the popup window explain the error to user.



Chapter 9.

1.Public Key Infrastructure

1.1.What is PKI ?



- The **Public Key Infrastructure (PKI)** is a set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates.
- PKI ensures a secure method for exchanging sensitive information over unsecured networks. In addition, PKI provides authenticated, private and non-reputable communications.
- PKI makes use of the technology known as public key cryptography. Public key cryptography uses a pair of keys to scramble and decipher messages, a public key and a private key. The public key is widely distributed, whereas the private key is held secretly by an individual. Messages are protected from malicious people by scrambling them with the public key of the recipient. Only the recipient can decrypt the message by using his / her private key, thus retaining the privacy of the message. The public key is distributed with a digital certificate that contains information that uniquely identifies an individual (for example name, email address, the date the certificate was issued, and the name of the certificate authority which issued it). Also, by using digital certificates we can digitally sign messages to protect the integrity of the information itself and achieve non-repudiation (digitally signing a transaction is legally binding and no party can deny his /her participation).

1.2.Components of PKI

-
-
- Digital certificates are issued by organizations called **Certification Authorities (CAs)** i.e. Verisign. The user identity must be unique for each CA. The binding is established through the registration and issuance process, which, depending on the level of assurance the binding has, may be carried out by software at a CA, or under human supervision.
 - Requests for certificates are usually processed by organizations called **Registration Authorities (RAs)**. An RA's responsibility is to evaluate each request, investigate the profile of each applicant and inform the appropriate CA about the trusting level of the client. After the trust verification of the applicant, CA issues the certificate. I have to mention that it is common for RA to be included within the CA environment as one component.
 - The operation of CAs and RAs are governed by appropriate policies, the **Certificate Policy (CP)** and the **Certificate Practice Statement (CPS)**. The first provides rules for naming certificate holders, the cryptographic algorithms that will be used, the minimum allowable length of encryption keys, etc. The latter details how the Certification Authority will implement the Certificate Policy (CP) into its procedures.
 - SomeCAs
-
-

Certificate Name	Security Device
Thawte Personal Freemail CA	Builtin Object Token
▲Thawte Consulting cc	
Thawte Premium Server CA	Builtin Object Token
Thawte Server CA	Builtin Object Token
▲thawte, Inc.	
thawte Primary Root CA	Builtin Object Token
thawte Extended Validation SSL CA	Software Security Device
▲The Go Daddy Group, Inc.	
Go Daddy Secure Certification Authority	Software Security Device
Go Daddy Class 2 CA	Builtin Object Token
▲The USERTRUST Network	
PositiveSSL CA	Software Security Device
Network Solutions Certificate Authority	Software Security Device
UTN-USERFirst-Hardware	Builtin Object Token
UTN - DATACorp SGC	Builtin Object Token
UTN-USERFirst-Network Applications	Builtin Object Token
UTN-USERFirst-Client Authentication and ...	Builtin Object Token
UTN-USERFirst-Object	Builtin Object Token
WebSpace-Forum Server CA	Software Security Device
▲TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği...	
TÜRKTRUST Elektronik Sertifika Hizmet Sa...	Builtin Object Token
▲Unizeto Sp. z o.o.	
Certum CA	Builtin Object Token
▲ValiCert, Inc.	
Starfield Secure Certification Authority	Software Security Device
http://www.valicert.com/	Builtin Object Token
http://www.valicert.com/	Builtin Object Token
http://www.valicert.com/	Builtin Object Token
▲VeriSign, Inc.	
Sun Microsystems Inc SSL CA	Software Security Device

1.3. Issuing a certificate

- A CA issues [digital certificates](#) that contain a [public key](#) and the identity of the owner. The matching private key is not similarly made available publicly, but kept secret by the end user who generated the key pair. The certificate is also an attestation by the CA that the public key contained in the certificate belongs to the person, organization, server or other entity noted in the certificate. A CA's obligation in such schemes is to verify an applicant's credentials, so that users and relying parties can trust the information in the CA's certificates. CAs use a variety of standards and tests to do so.
- If the user trusts the CA and can verify the CA's signature, then he can also verify that a certain public key does indeed belong to whomever is identified in the certificate.

1.4.CA architectures

In this section I will present the different types of CA architectures that are generally considered when implementing a PKI. PKI may be constructed as a:

- Single architecture
- Hierarchical architecture
- Mesh architecture

Every architecture is distinct from the others in respect to the following:

- The numbers of CAs in the PKI system
- Where users place their trust (known as a user's trust point)
- Trust relationships between CAs within a multi-CA PKI

1.4.1.Single Architecture

- A single architecture is the most basic PKI model that contains only a single (you wouldn't expect more, would you?) CA. All the users of the PKI place their trust on this CA. The CA will be responsible in handling all the users requesting a certificate. As there is only one CA, every certification path will begin with its public key.

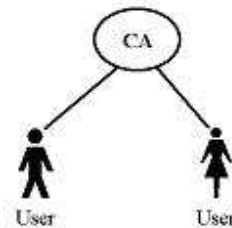


Figure 5.1 Single PKI Architecture

1.4.2.Hierarchical Architecture

- A hierarchical architecture is constructed with subordinate CA relationships. In this configuration, all users trust a single "root" CA. The root CA issues certificates to subordinate CAs only, whereas subordinate CAs may issue certificates to users or other CAs. The trust relationship is specified in only one direction. In this PKI architecture, every certification path begins with the root CA's public key.
-
-

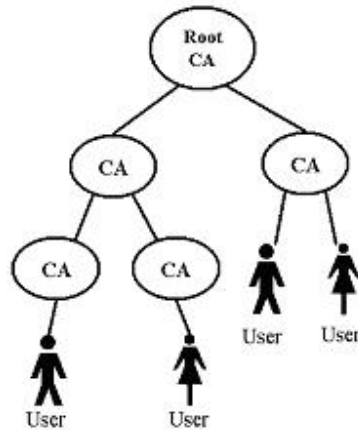


Figure 5.2 Hierarchical PKI Architecture

1.4.3.Mesh Architecture

- A mesh architecture does not include only one CA that is trusted by all entities in the PKI system. CAs can be connected with cross certification creating a "web of trust" where end entities may choose to trust any CA in the PKI. If a CA wishes to impose constraints on certain trust relationships, it must specify appropriate limitations in the certificates issued to its peers.

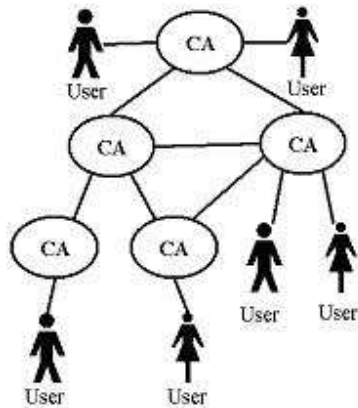


Figure 5.3 Mesh PKI Architecture

1.5.PKI Architecture Overview

[Applications]		
System Security-Enabling Services	Secure Protocols	Security Policy Services
	Protocol Security Services	
	Long-Term Key Services	Supporting Services
Cryptographic Services		
Cryptographic Primitives		

Figure: PKI Architecture Overview

The PKI Architecture components are grouped into the following broad functional categories:

- **System Security-enabling Services** provide the functionality which allows a user's or other principal's identity to be established and associated with their actions in the system.
- **Cryptographic Primitives and Services** provide the cryptographic functions on which public-key security is based (including secret-key primitives, such as the Data Encryption Standard (DES)).
- **Long-term Key Services** permit users and other principals to manage their own long-term keys and certificates and to retrieve and check the validity of other principals' certificates.
- **Protocol Security Services** provide security functionality (data origin authentication, data integrity protection, data privacy protection, non-repudiation) suitable for use by implementors of security-aware applications, such as secure protocols.
- **Secure Protocols** provide secure inter-application communications for security-unaware and "mildly" security-aware applications.
- **Security Policy Services** provide the policy-related information which must be carried in secure protocols to enable access control, and provide access control checking facilities to security-aware applications which must enforce policy.
- **Supporting Services** provide functionality which is required for secure operation, but is not directly involved in security policy enforcement.

1.6.Digital Signature

1.6.1. Need of Digital Signature

- Digital signatures are used to digitally sign objects or message. Digital signature are not only verify the content of message but also help to identify the creator of message. It is impossible to forge a digital signature or alter the content of signed message without invalidating the signature. Hence the signature is use for two purposes.
- **Ensure message content integrity.** Some mathematic calculations are performed repeatedly in message to generate a digital signature. The signature is appened at the end of each message before transmission over a network. If an encrypt message is tamperetd with, the digital signature becomes invalid.
- Verifying the authenticity of message sender. A digital signature ensure that an encrypt message can not be deciphered by unintended recipients. Thus a sender can ensure about the authenticity of recipient. The sender public and private keys are mathematically related. A recipient can also ensure the authenticity of the sender by using the sender public key for decryption.

1.6.2. What the digital signature made of ?

- Digital Signature is generated by Public Key Cryptography, using public key and private key to decrypt and encrypt message.
- Public key. Each sender has a unique public key which is accessible easily to the recipients. A public key is used by recipient to decrypt message. Hence maintaining the authenticity of public key is while transmitting the message is very important.
- Private key. Sender encrypt message with his private key. If receiver can decrypt with sender's public key, the data must be from the sender.

1.6.3. Working of digital signature

Encryption

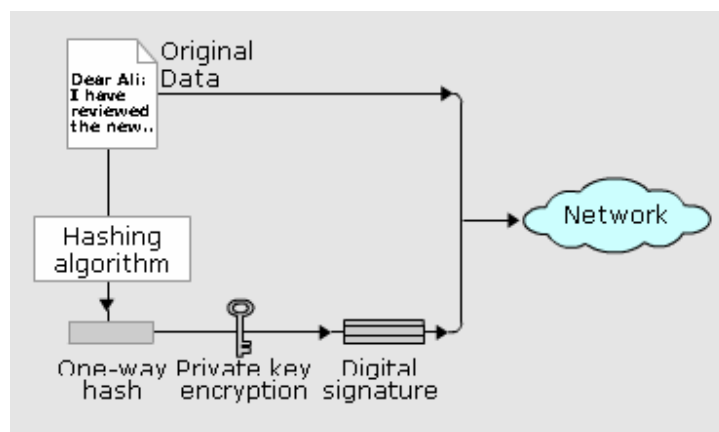


Figure 5.4 Encryption

- A hash or a message digest is prepared using the hash algorithm.
- The hash data or the message digest is encrypted using sender's private key.
- The digital signature and the public key of sender is append at the end of message.

Decryption

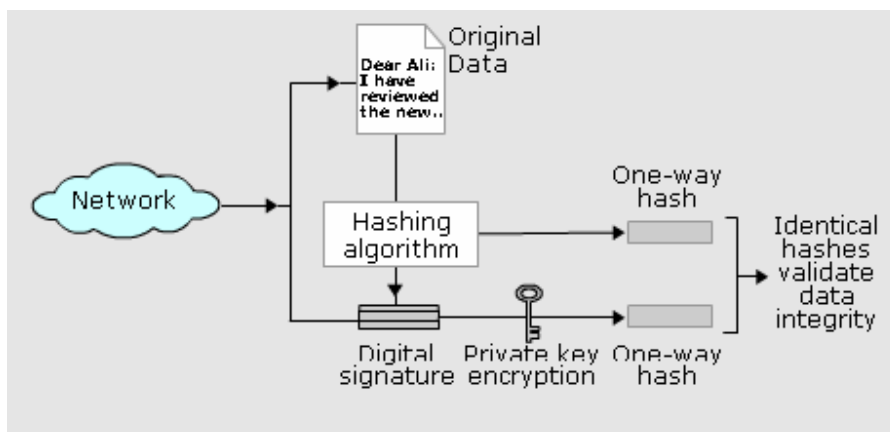


Figure 5.5 Decryption

- The receiver receives message and the digitally signed message digest.
- The receiver separately calculates a message digest for received message.
- The receiver use sender's public key to decrypt the signed message digest that was received and compare this to independently calculated message digest.
- If the two digest not math, the data may has been tampered or data may not be authentic or data not have been intended for the receiver.

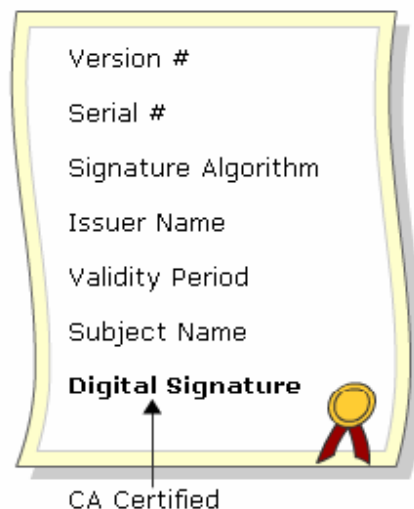
1.6.4. Validating data integrity

- In data communication, data integrity is said to be maintain if there is no different between data that is sent and received. The receiver check data integrity by calculate new hash value on the received message. The receiver also decrypts the signed hash using the sender's public key to ensure the authenticity of the message. The computed hash is compare to the hash decrypted from the digital signature. If two hash is not math, the signature has been created with private key not correspond to the public key present by the message sender.

1.6.5. Drawback of digital signature

-
-
- **Non-repudiation.** Repudiation means disclaiming responsibility for a sent message. A digital signature make non-repudiation difficult but if the sender claims to have lost its private key then the authenticity of all messages having digital signatures using that key would have been compromised.
 - **Time Stamping.** Digital signature can contain any record of date and time when particular document was signed. Hence, a reader can not be sure that a signer has not misused the digital signature for an older message.

1.7.Digital Certificates



Impersonation using the false public key can be reduce with the used of digital certificates. Digital certificate prevent impersonation by storing widely known and distributed public key, information such as name, email address and other application-specific data about the certificate owner. Thus the digital certificate is an identity document issued by the certificate authority to authenticate a message sender.

1.7.1.Standards and features of digital certificates

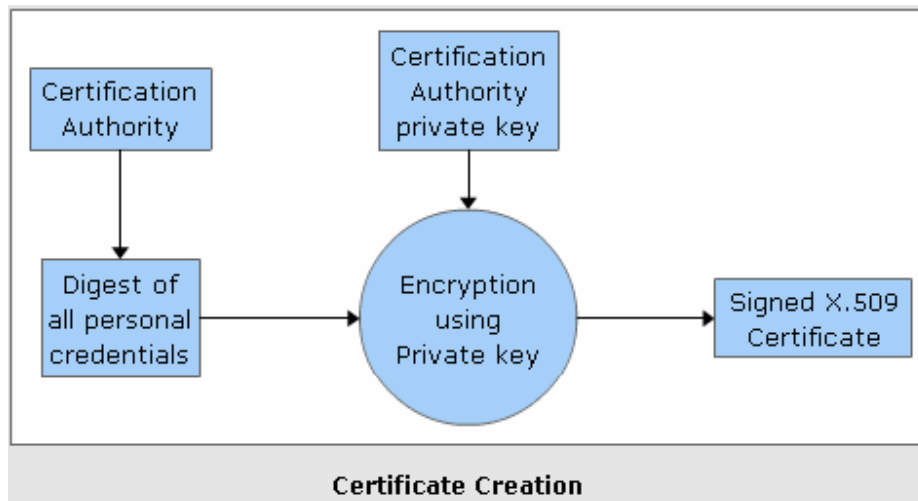
- A Digital certificate is a structure with digital signature. The data structure also contains information like the public key, identity of key owner and the name of certification authority who guarantees the authenticity of key owner. A signature is viewed as trusted when it generated by the CA since the digitally signed data can not be altered without detection. Certificate Extensions can be used to customize certificates to satisfy end-user.

Commonly, two types of standards are used for issuing digital certificates.

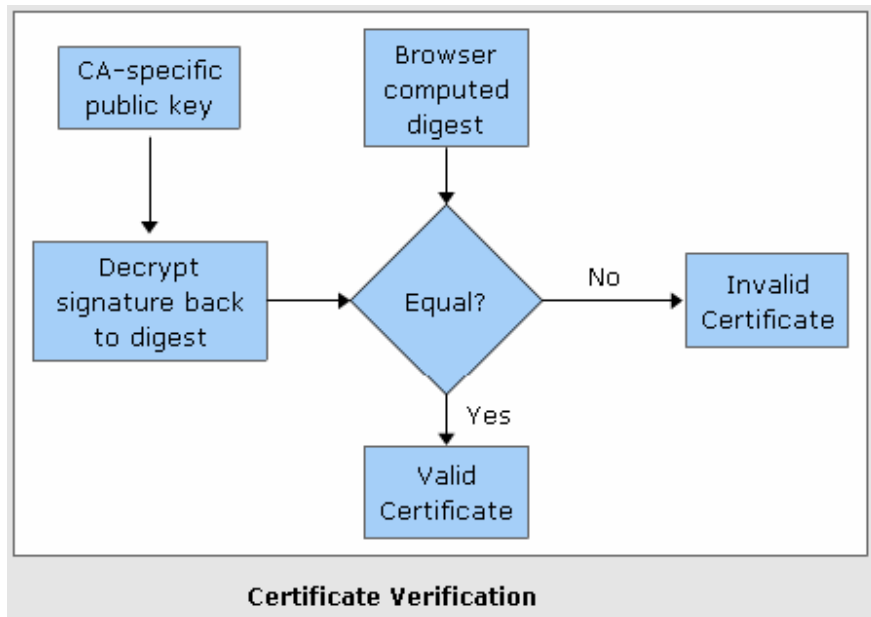
-
-
- X.509 is created by the international telephone standards body and is issued by Microsoft's Authenticode, Netscape's Object Signing, and Marimba's channel signing, to authenticate the originator of Internet Objects.
 - PGP (Pretty good Privacy) is developed by Phil Zimmermann. PGP is used for encrypting, compressing and authenticating email message and attachments.

1.7.2. Verify the authenticity of the sender

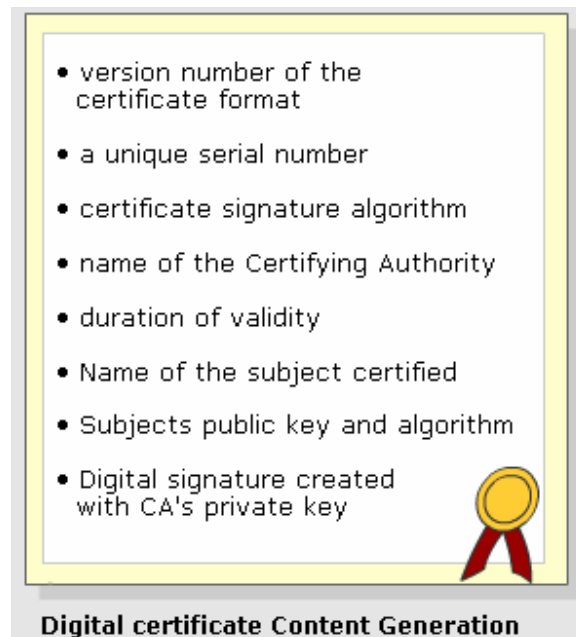
- The first step involves validation of credential of the party applying for digital certificates. A digital ID is used along with a public key encryption system for credentials verification. The certification authority verify that public key belongs to the specific individual or company. The verification depth depends on the level of certificates and the CA itself.



- The second step is the creation of certificate. If the validation process is completed successfully, the CA create an X.509 certificate. The CA then sign the certificate by creating a digest (hash) of all the fields in the certificate and encrypt the hash value with its private key. The encrypt digest is placed into a X.509 certificate and the certificate is said to be signed.
 - The private key of CA is kept very secure, because false certificates can be created if the key is misplace.
-
-



- A third step is verify the certificate. A recipient generally verify the signed certificate using web browser. A web browser typically maintains the list of popular CA and this public keys, the appropriate key is used to decrypt the signature back to the digest. The browser re-computes it owns digest from the plain text in the certificate and verify it with the decrypted digest. If the two digest math then the certificate is said to be valid and the public key in the certificate is accepted as a valid public key of the subject.



- The fourth step is generate the content of digital certificate. The certificate is combine with a signed message or signed executable file. A public key is

used to verify the signatures. If a secure two-way communication session is required the subject's public key is used.

- A certificate generally contain
 - Version number of certificate format.
 - A Unique serial number.
 - Certificate signature algorithm.
 - Name of the CA
 - Duration of Validity
 - Name of subject certified
 - Subject public key and algorithm
 - Digital signature created with CA's private key.
- The fifth step is sign and verify the digital certificate. The signed certificate can now be used to verify the authenticity of the sender.



2. Secure Socket Layer

2.1. What is SSL?

- The Secure Sockets Layer (SSL) is a commonly-used [protocol](#) for managing the security of a message transmission on the Internet. SSL has recently been succeeded by Transport Layer Security ([TLS](#)), which is based on SSL. SSL uses a program [layer](#) located between the Internet's Hypertext Transfer Protocol ([HTTP](#)) and Transport Control Protocol ([TCP](#)) layers.
- The "sockets" part of the term refers to the [sockets](#) method of passing data back and forth between a client and a server program in a network or between program layers in the same computer.
- SSL uses the public-and-private key [encryption](#) system from [RSA](#), which also includes the use of a [digital certificate](#).
- Several versions of the protocols are in wide-spread use in applications like [web browsing](#), [electronic mail](#), [Internet faxing](#), [instant messaging](#) and [voice-over-IP \(VoIP\)](#).

2.2. SSL Feature

- SSL is supported by most Web Server and browser.
- Only trusted digital certificates are needed to protect Web Application through SSL.
- In Client-Server operations, the SSL protocol use the third party – the CA to identify one-end or both end of communication.
- SSL encrypt data transmission and incorporates a mechanism to detect any change in data transmission. This help prevent evavesdropping or tampering with sensitive data during transmission.

2.3. SSL with client browser and server

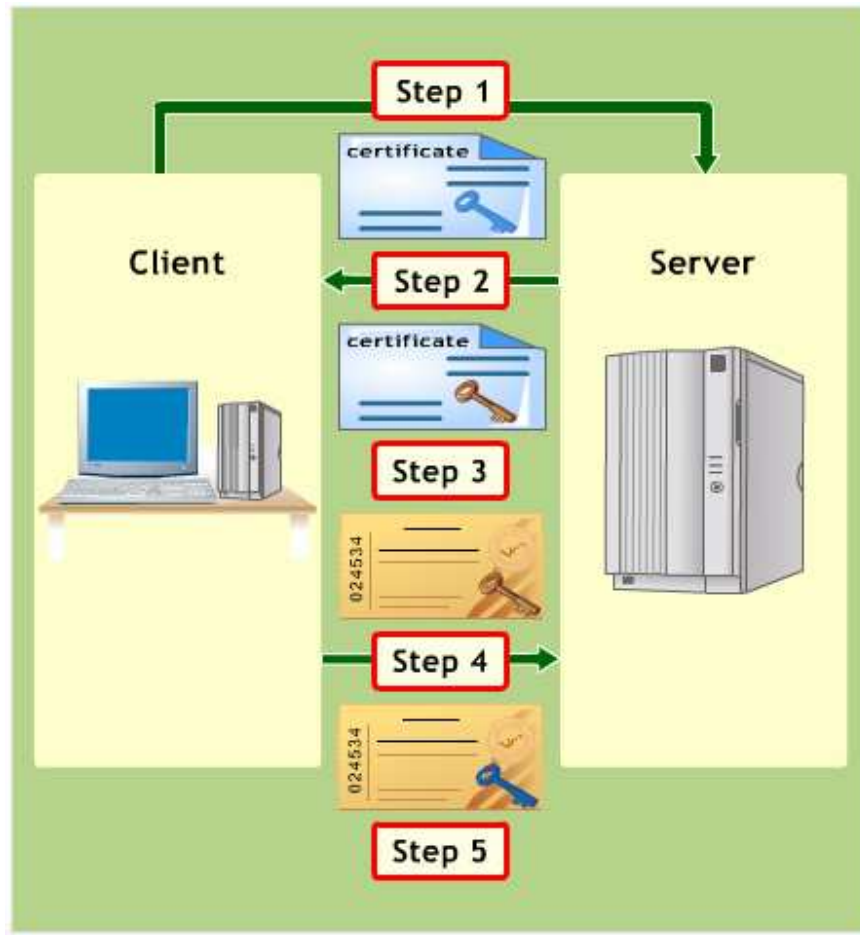


Figure 6.1 Client and Server with SSL

SSL uses the public key and private key to encrypt. A public key is known to everyone, and the private key is known only to the recipient of the message. A typical SSL handshake has five steps.

- Step 1. Client contacts the Web Browser.
- Step 2. The server sends back the certificate, encrypted with a trusted third-party private key.
- Step 3. The browser decrypts the certificate with a trusted third-party public key.
- Step 4. The browser uses a trusted third-party public key to encrypt a session ticket. The ticket is sent back to the server.
- Step 5. The web server receives the ticket and decrypts the session ticket with its private key. The server and the web browser use the same session ticket for future encryption in transmission.

Obtaining the SSL certificates

Consider a web application in which you want to implement SSL for login page. To use SSL you need to obtain a certificate. To get an SSL certificate, a certificates signing request has to be submitted. SCR is data file that holds details of the request party to a CA.

2.4.Detail in SSL Handshake

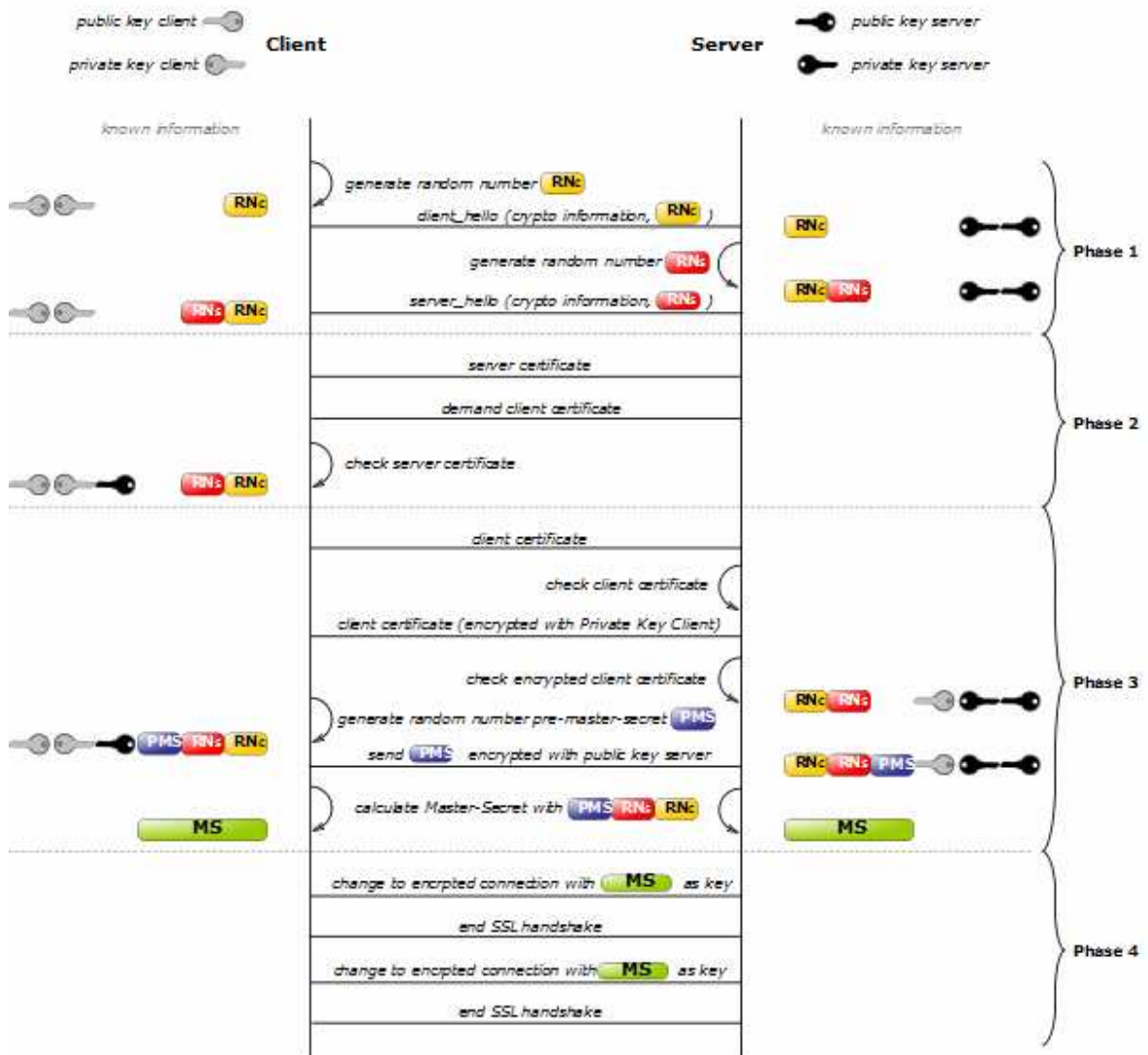


Figure 6.2 Scenario SSL

2.4.1.Simple TLS/SSL handshake

A simple connection example, illustrating a handshake where the server is authenticated by its certificate (but not the client), follows:

-
-
- Negotiation phase:
 - A client sends a **ClientHello** message specifying the highest TLS/SSL protocol version it supports, a random number, a list of suggested cipher suites and compression methods.
 - The server responds with a **ServerHello** message, containing the chosen protocol version, a random number, cipher suite, and compression method from the choices offered by the client. The server may also send a session id as part of the message to perform a resumed handshake.
 - The server sends its **Certificate** message (depending on the selected cipher suite, this may be omitted by the server).
 - The server sends a **ServerHelloDone** message, indicating it is done with handshake negotiation.
 - The client responds with a **ClientKeyExchange** message, which may contain a PreMasterSecret, public key, or nothing. (Again, this depends on the selected cipher.)
 - The client and server then use the random numbers and PreMasterSecret to compute a common secret, called the "master secret". All other key data for this connection is derived from this master secret (and the client- and server-generated random values), which is passed through a carefully designed "pseudorandom function".
 - The client now sends a **ChangeCipherSpec** record, essentially telling the server, "Everything I tell you from now on will be authenticated (and encrypted if encryption parameters were present in the server certificate)." The ChangeCipherSpec is itself a record-level protocol with content type of 20.
 - Finally, the client sends an authenticated and encrypted **Finished** message, containing a hash and MAC over the previous handshake messages.
 - The server will attempt to decrypt the client's Finished message, and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.
 - Finally, the server sends a **ChangeCipherSpec**, telling the client, "Everything I tell you from now on will be authenticated (and encrypted with the server private key associated to the public key in the server certificate, if encryption was negotiated)."
 - The server sends its authenticated and encrypted **Finished** message.
 - The client performs the same decryption and verification.
 - Application phase: at this point, the "handshake" is complete and the application protocol is enabled, with content type of 23. Application messages exchanged between client and server will also be authenticated and optionally encrypted exactly like in their Finished message.
-
-

2.4.2. Client-authenticated TLS/SSL handshake

The following full example shows a client being authenticated (in addition to the server like above) via TLS using certificates exchanged between both peers.

- Negotiation phase:
 - A client sends a **ClientHello** message specifying the highest TLS/SSL protocol version it supports, a random number, a list of suggested cipher suites and compression methods.
 - The server responds with a **ServerHello** message, containing the chosen protocol version, a random number, cipher suite, and compression method from the choices offered by the client. The server may also send a session id as part of the message to perform a resumed handshake.
 - The server sends its **Certificate** message (depending on the selected cipher suite, this may be omitted by the server).
 - The server requests a certificate from the client, so that the connection can be mutually authenticated, using a **CertificateRequest** message.
 - The server sends a **ServerHelloDone** message, indicating it is done with handshake negotiation.
 - The client responds with a **Certificate** message, which contains the client's certificate.
 - The client sends a **ClientKeyExchange** message, which may contain a PreMasterSecret, public key, or nothing. (Again, this depends on the selected cipher.) This PreMasterSecret is encrypted using the public key of the server certificate.
 - The client sends a **CertificateVerify** message, which is a signature over the previous handshake messages using the client's certificate's private key. This signature can be verified by using the client's certificate's public key. This lets the server know that the client has access to the private key of the certificate and thus owns the certificate.
 - The client and server then use the random numbers and PreMasterSecret to compute a common secret, called the "master secret". All other key data for this connection is derived from this master secret (and the client- and server-generated random values), which is passed through a carefully designed "pseudorandom function".
 - The client now sends a **ChangeCipherSpec** record, essentially telling the server, "Everything I tell you from now on will be authenticated (and encrypted if encryption was negotiated)." The ChangeCipherSpec is itself a record-level protocol, and has type 20, and not 22.
-
-

-
-
- Finally, the client sends an encrypted **Finished** message, containing a hash and MAC over the previous handshake messages.
 - The server will attempt to decrypt the client's Finished message, and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.
 - Finally, the server sends a **ChangeCipherSpec**, telling the client, "Everything I tell you from now on will be authenticated (and encrypted if encryption was negotiated)."
 - The server sends its own encrypted **Finished** message.
 - The client performs the same decryption and verification.
 - Application phase: at this point, the "handshake" is complete and the application protocol is enabled, with content type of 23. Application messages exchanged between client and server will also be encrypted exactly like in their Finished message.

2.4.3. Resumed TLS/SSL handshake by Session ID

Public key operations (e.g., RSA) are relatively expensive in terms of computational power. TLS provides a secure shortcut in the handshake mechanism to avoid these operations. In an ordinary full handshake, the server sends a session id as part of the **ServerHello** message. The client associates this session id with the server's IP address and TCP port, so that when the client connects again to that server, it can use the session id to shortcut the handshake. In the server, the session id maps to the cryptographic parameters previously negotiated, specifically the "master secret". Both sides must have the same "master secret" or the resumed handshake will fail (this prevents an eavesdropper from using a session id). The random data in the **ClientHello** and **ServerHello** messages virtually guarantee that the generated connection keys will be different than in the previous connection. In the RFCs, this type of handshake is called an abbreviated handshake. It is also described in the literature as a restart handshake.

1. Negotiation phase:
 - A client sends a **ClientHello** message specifying the highest TLS protocol version it supports, a random number, a list of suggested cipher suites and compression methods. Included in the message is the session id from the previous TLS connection.
 - The server responds with a **ServerHello** message, containing the chosen protocol version, a random number, cipher suite, and compression method from the choices offered by the client. If the server recognizes the session id sent by the client, it responds with the same session id. The client uses this to recognize that a resumed handshake is being performed. If the server does not recognize the session id sent by the client, it sends a different value for its session
-
-

-
-
- id. This tells the client that a resumed handshake will not be performed. At this point, both the client and server have the "master secret" and random data to generate the key data to be used for this connection.
2. The client now sends a **ChangeCipherSpec** record, essentially telling the server, "Everything I tell you from now on will be encrypted." The ChangeCipherSpec is itself a record-level protocol, and has type 20, and not 22.
 - o Finally, the client sends an encrypted **Finished** message, containing a hash and MAC over the previous handshake messages.
 - o The server will attempt to decrypt the client's Finished message, and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.
 3. Finally, the server sends a **ChangeCipherSpec**, telling the server, "Everything I tell you from now on will be encrypted."
 - o The server sends its own encrypted **Finished** message.
 - o The client performs the same decryption and verification.
 4. Application phase: at this point, the "handshake" is complete and the application protocol is enabled, with content type of 23. Application messages exchanged between client and server will also be encrypted exactly like in their Finished message.

Apart from the performance benefit, resumed sessions can also be used for single sign-on as it is guaranteed that both the original session as well as any resumed session originate from the same client. This is of particular importance for the [FTP over TLS/SSL](#) protocol which would otherwise suffer from a man in the middle attack in which an attacker could intercept the contents of the secondary data connections.

2.5. TSL/SSL record protocol

This is the general format of all TLS records.

+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	Content type			
Bytes 1..4	Version (Major)	(Minor)	Length (bits 15..8)	(bits 7..0)
Bytes 5..(m-1)	Protocol message(s)			
Bytes m..(p-1)	MAC (optional)			
Bytes p..(q-1)	Padding (block ciphers only)			

Content type. This field identifies the Record Layer Protocol Type contained in this Record.

Content types		
Hex	Dec	Type
0x14	20	ChangeCipherSpec
0x15	21	Alert
0x16	22	Handshake
0x17	23	Application

Version. This field identifies the major and minor version of TLS for the contained message. For a ClientHello message, this need not be the *highest* version supported by the client.

Versions		
Major Version	Minor Version	Version Type
3	0	SSLv3
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2

Length. The length of Protocol message(s), not to exceed 2^{14} bytes (16 KiB).

Protocol message(s). One or more messages identified by the Protocol field. Note that this field may be encrypted depending on the state of the connection.

MAC and Padding. A [message authentication code](#) computed over the Protocol message, with additional key material included. Note that this field may be encrypted, or not included entirely, depending on the state of the connection. No MAC or Padding can be present at end of TLS records before all cipher algorithms and parameters have been negotiated and handshaked, and then confirmed by sending a CipherStateChange record (see below) for signaling that these parameters will take effect in all further records sent by the same peer.

2.6. Handshake protocol

Most messages exchanged during the setup of the TLS session are based on this record, unless an error or warning occurs and needs to be signaled by an Alert protocol record (see below), or the encryption mode of the session is modified by another record (see ChangeCipherSpec protocol below).

+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	22			
Bytes 1..4	Version (Major)	Version (Minor)	Length (bits 15..8)	Length (bits 7..0)
Bytes 5..8	Message type	Handshake message data length (bits 23..16)		Handshake message data length (bits 15..8)
Bytes 9..(n-1)	Handshake message data			
Bytes n..(n+3)	Message type	Handshake message data length (bits 23..16)	Handshake message data length (bits 15..8)	Handshake message data length (bits 7..0)
Bytes (n+4)..	Handshake message data			

Message type. This field identifies the Handshake message type.

Message Types	
Code	Description
0	HelloRequest
1	ClientHello
2	ServerHello
11	Certificate
12	ServerKeyExchange
13	CertificateRequest
14	ServerHelloDone
15	CertificateVerify
16	ClientKeyExchange
20	Finished

Handshake message data length. This is a 3-byte field indicating the length of the handshake data, not including the header.

Note that multiple Handshake messages may be combined within one record.

2.7. Alert protocol

This record should normally not be sent during normal handshaking or application exchanges. However, this message can be sent at any time during the handshake and up to the closure of the session. If this is used to signal a fatal error, the session will be closed immediately after sending this record, so this record is used

to give a reason for this closure. If the alert level is flagged as a warning, the remote can decide to close the session if it decides that the session is not reliable enough for its needs (before doing so, the remote may also send its own signal).

+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	21			
Bytes 1..4	Version (Major)	(Minor)	Length 0	2
Bytes 5..6	Level	Description		
Bytes 7..(p-1)	<u>MAC</u> (optional)			
Bytes p..(q-1)	Padding (block ciphers only)			

Level. This field identifies the level of alert. If the level is fatal, the sender should close the session immediately. Otherwise, the recipient may decide to terminate the session itself, by sending its own fatal alert and closing the session itself immediately after sending it. The use of Alert records is optional, however if it is missing before the session closure, the session may be resumed automatically (with its handshakes).

Normal closure of a session after termination of the transported application should preferably be alerted with at least the Close notify Alert type (with a simple warning level) to prevent such automatic resume of a new session. Signaling explicitly the normal closure of a secure session before effectively closing its transport layer is useful to prevent or detect attacks (like attempts to truncate the securely transported data, if it intrinsically does not have a predetermined length or duration that the recipient of the secured data may expect).

Alert level types		
Code	Level type	Connection state
1	warning	Connection or security may be unstable.
2	fatal	Connection or security may be compromised, or an unrecoverable error has occurred.

Description. This field identifies which type of alert is being sent.

Alert description types			
Code	Description	Level types	Note
0	Close notify	warning or fatal	
10	Unexpected message	fatal	
20	Bad record MAC	fatal	
21	Decryption failed	fatal	TLS only, reserved
22	Record overflow	fatal	TLS only

30	Decompression failure	fatal	
40	Handshake failure	fatal	
41	No certificate	warning or fatal	SSL v3 only, reserved
42	Bad certificate	warning or fatal	
43	Unsupported certificate	warning or fatal	
44	Certificate revoked	warning or fatal	
45	Certificate expired	warning or fatal	
46	Certificate unknown	warning or fatal	
47	Illegal parameter	fatal	
48	Unknown CA	fatal	TLS only
49	Access denied	fatal	TLS only
50	Decode error	fatal	TLS only
51	Decrypt error	warning or fatal	TLS only
60	Export restriction	fatal	TLS only, reserved
70	Protocol version	fatal	TLS only
71	Insufficient security	fatal	TLS only
80	Internal error	fatal	TLS only
90	User cancelled	fatal	TLS only
100	No renegotiation	warning	TLS only
110	Unsupported extension	warning	TLS only

2.8.ChangeCipherSpec protocol

+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	20			
Bytes 1..4	Version (Major)	(Minor)	Length 0	1
Byte 5	CCS protocol type			

CCS protocol type. Currently only 1.

2.9.Application protocol

+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	23			
Bytes 1..4	Version (Major)	(Minor)	Length (bits 15..8)	(bits 7..0)
Bytes 5..(m-1)	Application data			
Bytes m..(p-1)	MAC (optional)			

Bytes Padding (block ciphers only)
p..(q-1)

Length. Length of Application data (excluding the protocol header, and the MAC and padding trailers).

MAC. 20 bytes for the [SHA-1](#)-based [HMAC](#), 16 bytes for the [MD5](#)-based HMAC.

Padding. Variable length ; last byte contains the padding length.

2.10.Example of Certificate

The image shows a screenshot of a certificate viewer interface. It is divided into three main sections:

- Certificate Hierarchy:** A tree view showing the following structure:
 - StartCom Certification Authority
 - StartCom Class 1 Primary Intermediate Client CA
 - StartCom Free Certificate Member
- Certificate Fields:** A list of certificate fields with 'Issuer' selected and highlighted in blue. The fields listed are:
 - Certificate
 - Version
 - Serial Number
 - Certificate Signature Algorithm
 - Issuer
 - Validity
 - Not Before
 - Not After
 - Subject
 - Subject Public Key Info
- Field Value:** A text area displaying the following values:
 - CN = StartCom Class 1 Primary Intermediate Client CA
 - OU = Secure Digital Certificate Signing
 - O = StartCom Ltd.
 - C = IL

Figure 6.4 SSL Certificate for Client

This certificate has been verified for the following uses:

SSL Server Certificate

Issued To

Common Name (CN) addons.mozilla.org
Organization (O) Mozilla Corporation
Organizational Unit (OU) Mozilla Add-ons
Serial Number 01:00:00:00:00:01:1C:7B:96:3A:0B

Issued By

Common Name (CN) GlobalSign Extended Validation CA
Organization (O) GlobalSign
Organizational Unit (OU) Extended Validation CA

Validity

Issued On 9/20/2008
Expires On 9/21/2010

Fingerprints

SHA1 Fingerprint 6A:DA:C1:8C:40:03:2A:B9:72:70:07:C9:83:29:13:3A:86:02:34:05
MD5 Fingerprint 27:09:B7:25:4F:63:6C:1E:32:72:2C:4D:6C:EE:BA:8A

Figure 6.5 SSL Certificate for Server

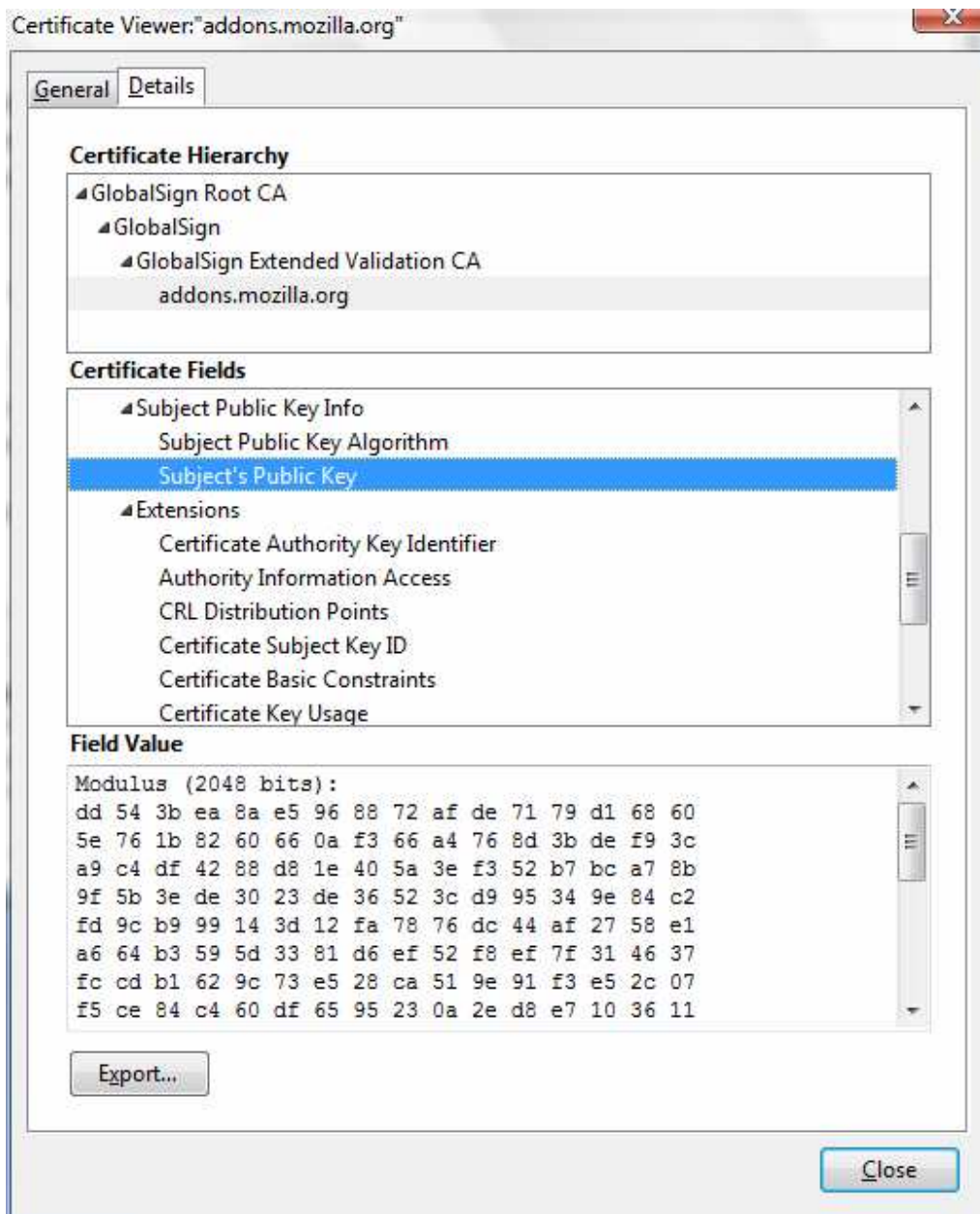


Figure 6.6 Subject's Public Key

3.HTTPS

3.1.Overview of HTTPS



- **Hypertext Transfer Protocol Secure (HTTPS)** is a combination of the [Hypertext Transfer Protocol](#) and a [cryptographic protocol](#). HTTPS connections are often used for payment transactions on the [World Wide Web](#) and for sensitive transactions in corporate information systems.
- HTTP operates at the highest layer of the [TCP/IP model](#), the Application layer; but the security protocol operates at a lower sublayer, encrypting an HTTP message prior to transmission and decrypting a message upon arrival.
- As opposed to [HTTP URLs](#) which begin with "http://" and use [port 80](#) by default, HTTPS URLs begin with "https://" and use [port 443](#) by default.

3.2.HTTPS Function

- Strictly speaking, HTTPS is not a separate protocol, but refers to use of ordinary [HTTP](#) over an [encrypted Secure Sockets Layer \(SSL\)](#) or [Transport Layer Security \(TLS\)](#) connection. This ensures reasonable protection from [eavesdroppers](#) and [man-in-the-middle attacks](#), provided that adequate cipher suites are used and that the server certificate is verified and trusted.
 - To prepare a web server to accept HTTPS connections, the administrator must create a [public key certificate](#) for the web server. This certificate must be signed by a trusted [certificate authority](#) for the web browser to accept it. The authority certifies that the certificate holder is indeed the entity it claims to be. Web browsers are generally distributed with the [signing certificates of major certificate authorities](#) so that they can verify certificates signed by them.
-
-

-
-
- A certificate may be revoked before it expires, for example because the secrecy of the private key has been compromised. Newer browsers such as [Firefox](#), [Opera](#), and [Internet Explorer](#) on [Windows Vista](#) implement the [Online Certificate Status Protocol](#) (OCSP) to verify that this is not the case. The browser sends the certificate's serial number to the certificate authority or its delegate via OCSP and the authority responds, telling the browser whether or not the certificate is still valid.
 - The system can also be used for client [authentication](#) in order to limit access to a web server to authorized users. To do this, the site administrator typically creates a certificate for each user, a certificate that is loaded into his/her browser. Normally, that contains the name and e-mail address of the authorized user and is automatically checked by the server on each reconnect to verify the user's identity, potentially without even entering a password.

3.3. Browser integration



Secure Connection Failed

svn.boost.org uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is unknown.

(Error code: sec_error_unknown_issuer)

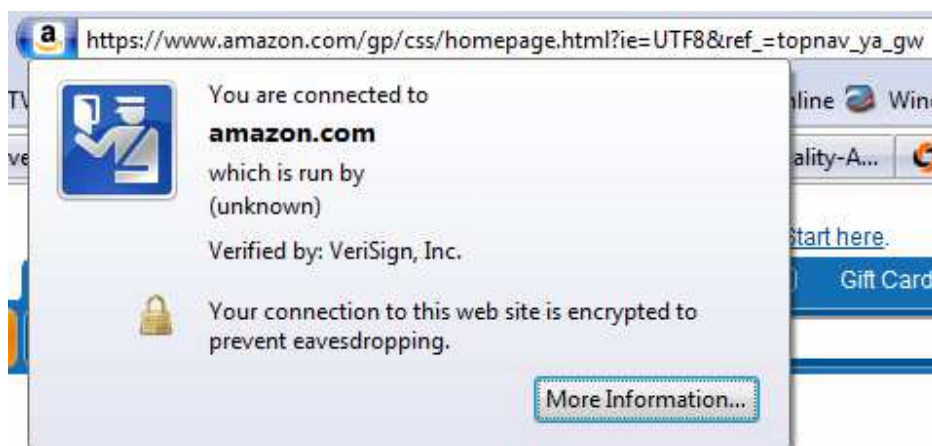
- This could be a problem with the server's configuration, or it could be someone trying to impersonate the server.
- If you have connected to this server successfully in the past, the error may be temporary, and you can try again later.

[Or you can add an exception...](#)

- When connecting to a site with an invalid certificate, older browsers would present the user with a dialog box asking if they wanted to continue. Newer browsers display a warning across the entire window. Newer browsers also prominently display the site's security information in the [address bar](#).
- [Extended validation](#) certificates turn the address bar green in newer browsers. Most browsers also pop up a warning to the user when visiting a site that contains a mixture of encrypted and unencrypted content.

3.4. Application of HTTPS

3.4.1. Online payment and Online Shopping



E-commerce payment system

- An e-commerce payment system facilitates the acceptance of [electronic payment](#) for [online transactions](#). Also known as [Electronic Data Interchange](#) (EDI), e-commerce payment systems have become increasingly popular due to the widespread use of the internet-based shopping and banking.
 - In the early years of [B2C](#) transactions, many consumers were apprehensive of using their credit and debit cards over the internet because of the perceived increased risk of [fraud](#). Recent research shows that 30% of people in the United Kingdom still do not shop online because they do not trust online payment systems. However, 54% do believe that it is safe to shop online which is an increase from 26% in 2006.
 - There are numerous different payments systems available for online merchants. These include the traditional credit, debit and charge card but also new technologies such as [digital-wallets](#), [e-cash](#), [mobile payment](#) and
-

-
-
- [e-checks](#). Another form of payment system is allowing a 3rd party to complete the online transaction for you. These companies are called [Payment Service Providers \(PSP\)](#), a good example is [Paypal](#) or [WorldPay](#). (Note Paypal also offers its own payment system).
- A **payment system** is a system (including physical or electronic infrastructure and associated procedures and protocols) used to settle [financial transactions](#) in [bond markets](#), [currency markets](#), and [futures](#), [derivatives](#) or [options](#) markets, or to transfer funds between [financial institutions](#). Due to the backing of modern [fiat currencies](#) with [government bonds](#), payment systems are a core part of modern monetary systems.
 - **Electronic money** (also known as **e-money**, **electronic cash**, **electronic currency**, **digital money**, **digital cash** or **digital currency**) refers to [money](#) or [scrip](#) which is exchanged only [electronically](#). Typically, this involves use of [computer networks](#), the [internet](#) and [digital stored value](#) systems. [Electronic Funds Transfer \(EFT\)](#) and [direct deposit](#) are examples of electronic money. Also, it is a collective term for [financial cryptography](#) and technologies enabling it.

Online shopping

- Online shopping is the process consumers go through to purchase products or services over the [Internet](#). An online shop, eshop, e-store, internet shop, webshop, [webstore](#), online store, or virtual store evokes the physical analogy of buying [products](#) or [services](#) at a [bricks-and-mortar retailer](#) or in a [shopping mall](#).
- The metaphor of an [online catalog](#) is also used, by analogy with [mail order](#) catalogs. All types of stores have retail web sites, including those that do and do not also have physical storefronts and paper catalogs. Online shopping is a type of [electronic commerce](#) used for [business-to-business](#) (B2B) and [business-to-consumer](#) (B2C) transactions.

Credit Cards and Smart Cards

- Over the years, credit cards have become one of the most common forms of payment for e-commerce transactions. In North America almost 90% of online [B2C](#) transactions were made with this payment type. Turban et al. goes on to explain that it would be difficult for an online retailer to operate without supporting credit and debit cards due to its widespread use. Increased security measures such as the use of the [card verification number \(CVN\)](#) which detects fraud by comparing the verification number on the printed on the signature strip on the back of the card with the information on file with the cardholder's issuing bank.
 - Also online merchants have to comply with stringent rules stipulated by the credit and debit card issuers (Visa and Mastercard) this means that merchants must have security protocol and procedures in place to ensure
-
-

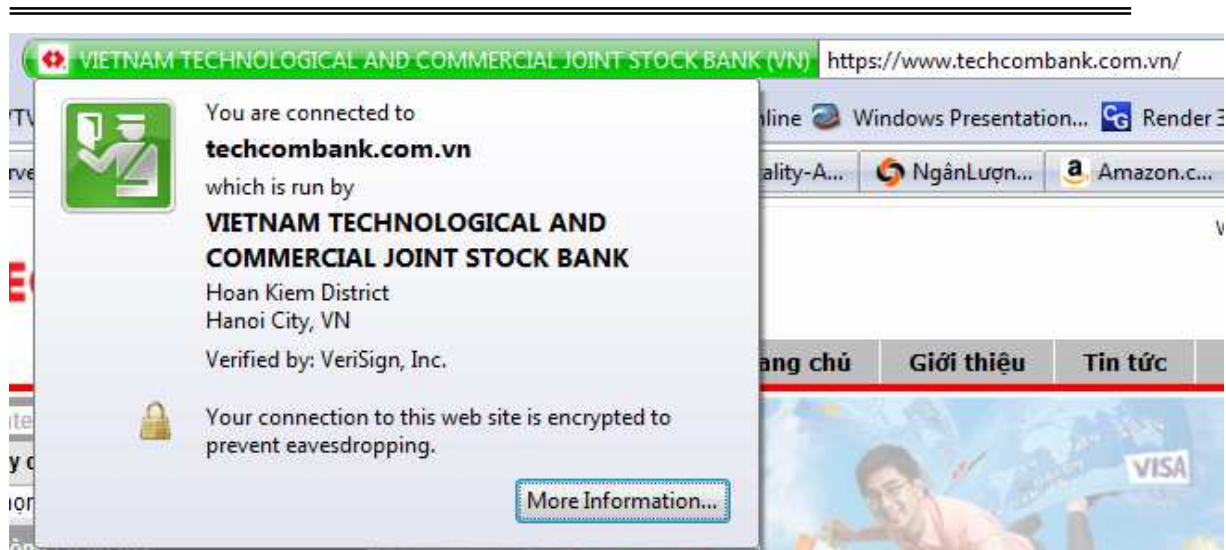
transactions are more secure. This can also include having a certificate from an authorised [certification authority \(CA\)](#) who provides PKI infrastructure for securing credit and debit card transactions.

- Despite this widespread use in North America, there are still a number of countries such as China, India and Pakistan that have some problems to overcome in regard to credit card security. In the meantime, the use of smartcards has become extremely popular. A [Smartcard](#) is similar to a credit card; however it contains an embedded 8-bit microprocessor and uses electronic cash which transfers from the consumers' card to the sellers' device. A popular smartcard initiative is the VISA Smartcard. Using the VISA Smartcard you can transfer electronic cash to your card from your bank account, and you can then use your card at various retailers and on the internet.

Payment service provider (PSP)

- A payment service provider (PSP) offers merchants online services for accepting electronic payments by a variety of payment methods including [credit card](#), bank-based payments such as [direct debit](#), [bank transfer](#), and real-time bank transfer based on [online banking](#). Some PSPs provide unique services to process other next generation methods ([Payment systems](#)) including cash payments, wallets such as [PayPal](#), prepaid cards or vouchers, and even paper or e-check processing.
- Typically, a PSP can connect to multiple [acquiring banks](#), card, and payment networks. In many cases the PSP will fully manage these technical connections, relationships with the external network, and bank accounts. This makes the merchant less dependent on financial institutions and free from the task of establishing these connections directly - especially when operating internationally.
- Furthermore, a full service PSP can offer [risk management](#) services for card and bank based payments, transaction payment matching, reporting, fund remittance and fraud protection in addition to [multi-currency](#) functionality and services.
- PSP fees are typically levied in one of two ways: As a percentage of each transaction or a low fixed cost per transaction.

3.4.2. Internet Banking



Online banking (or **Internet banking**) allows customers to conduct financial transactions on a secure website operated by their retail or [virtual bank](#), [credit union](#) or [building society](#).

The common features fall broadly into several categories

- Transactional (e.g., performing a financial transaction such as an account to account transfer, paying a bill, wire transfer... and applications... apply for a loan, new account, etc.)
 - [Electronic bill presentment and payment - EBPP](#)
 - [Funds transfer](#) between a customer's own [checking](#) and [savings accounts](#), or to another customer's account
 - [Investment](#) purchase or sale
 - [Loan](#) applications and transactions, such as repayments
- Non-transactional (e.g., online statements, check links, cobrowsing, chat)
 - [Bank statements](#)
- Financial Institution Administration - features allowing the financial institution to manage the online experience of their end users
- ASP/Hosting Administration - features allowing the hosting company to administer the solution across financial institutions

Features commonly unique to business banking include

- Support of multiple users having varying levels of authority
- Transaction approval process
- Wire transfer

Features commonly unique to Internet banking include

-
- Personal financial management support, such as importing data into personal [accounting software](#). Some online banking platforms support [account aggregation](#) to allow the customers to monitor all of their accounts in one place whether they are with their main bank or with other institutions.

3.4.3. Manage Certificate



VeriSign is the trusted provider of Internet infrastructure services for the networked world. The ability to know and trust the parties with which you do business and communicate has become critical in the networked world.

- VeriSign facilitates as many as **50 billion** authoritative Domain Name System (DNS) queries a day, and has been providing this service since 1998 with **100% availability**.
 - VeriSign plans to increase capacity of the .com and .net DNS by **10 times by 2010** to provide the security and stability required for global Internet-based transactions.
-

-
-
- VeriSign is the SSL Certificate provider of choice for over **95% of the Fortune 500** and the world's **40 largest banks**.
 - The VeriSign Secured® Seal, the most recognized symbol of trust on the Internet (TNS Study, 2006), is served over 150 million times a day.
 - VeriSign has issued over **1.4 million** VeriSign® Identity Protection (VIP) credentials to consumers for strong authentication on a network of leading Web sites.
-
-

Chapter 10.

1.Clustering

1.1.Cluster



A **computer cluster** is a group of linked [computers](#), working together closely so that in many respects they form a single computer. The components of a cluster are commonly, but not always, connected to each other through fast [local area networks](#). Clusters are usually deployed to improve performance and/or availability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

1.2.Cluster categorizations

1.2.1.High-availability (HA) clusters

- [High-availability clusters](#) (also known as Failover Clusters) are implemented primarily for the purpose of improving the availability of services which the cluster provides. They operate by having redundant [nodes](#), which are then used to provide service when system components fail. The most common size for an HA cluster is [two nodes](#), which is the minimum requirement to provide redundancy. HA cluster implementations attempt to use redundancy of cluster components to eliminate [single points of failure](#).
-
-

-
-
- There are many commercial implementations of High-Availability clusters for many operating systems. The [Linux-HA](#) project is one commonly used [free software](#) HA package for the [Linux](#) OSs.

1.2.2. Load-balancing clusters

- [Load-balancing](#) when multiple computers are linked together to share computational workload or function as a single virtual computer. Logically, from the user side, they are multiple machines, but function as a single virtual machine. Requests initiated from the user are managed by, and distributed among, all the standalone computers to form a cluster. This results in balanced computational work among different machines, improving the performance of the cluster system.

1.2.3. Compute clusters

- Often clusters are used for primarily computational purposes, rather than handling IO-oriented operations such as web service or databases. For instance, a cluster might support computational simulations of weather or vehicle crashes. The primary distinction within compute clusters is how tightly-coupled the individual nodes are. For instance, a single compute job may require frequent communication among nodes - this implies that the cluster shares a dedicated network, is densely located, and probably has homogenous nodes. This cluster design is usually referred to as [Beowulf Cluster](#). The other extreme is where a compute job uses one or few nodes, and needs little or no inter-node communication. This latter category is sometimes called "Grid" computing. Tightly-coupled compute clusters are designed for work that might traditionally have been called "supercomputing". Middleware such as [MPI \(Message Passing Interface\)](#) or [PVM \(Parallel Virtual Machine\)](#) permits compute clustering programs to be portable to a wide variety of clusters.

1.2.4. Grid computing

- Grids are usually computer clusters, but more focused on throughput like a [computing utility](#) rather than running fewer, tightly-coupled jobs. Often, grids will incorporate heterogeneous collections of computers, possibly distributed geographically, sometimes administered by unrelated organizations.
 - [Grid computing](#) is optimized for workloads which consist of many independent jobs or packets of work, which do not have to share data between the jobs during the computation process. Grids serve to manage the allocation of jobs to computers which will perform the work independently of the rest of the grid cluster. Resources such as storage may
-
-

be shared by all the nodes, but intermediate results of one job do not affect other jobs in progress on other nodes of the grid.

- An example of a very large grid is the [Folding@home](#) project. It is analyzing data that is used by researchers to find cures for diseases such as Alzheimer's and cancer. Another large project is the [SETI@home](#) project, which may be the largest distributed grid in existence. It uses approximately three million home computers all over the world to analyze data from the [Arecibo Observatory radiotelescope](#), searching for evidence of extraterrestrial intelligence. In both of these cases, there is no inter-node communication or shared storage. Individual nodes connect to a main, central location to retrieve a small processing job. They then perform the computation and return the result to the central server. In the case of the @home projects, the software is generally run when the computer is otherwise idle. The grid setup means that the nodes can take however many jobs they are able in one session.

1.3. Technologies

- [MPI](#) is a widely-available communications library that enables parallel programs to be written in [C](#), [Fortran](#), [Python](#), [OCaml](#), and many other programming languages.
- The GNU/Linux world supports various cluster software; for application clustering, there is [Beowulf](#), [distcc](#), and [MPICH](#). [Linux Virtual Server](#), [Linux-HA](#) - director-based clusters that allow incoming requests for services to be distributed across multiple cluster nodes. [MOSIX](#), [openMosix](#), [Kerrighed](#), [OpenSSI](#) are full-blown clusters integrated into the [kernel](#) that provide for automatic process migration among homogeneous nodes. OpenSSI, openMosix and Kerrighed are [single-system image](#) implementations.
- [Microsoft Windows](#) Compute Cluster Server 2003 based on the [Windows Server](#) platform provides pieces for High Performance Computing like the Job Scheduler, MSMPI library and management tools. [NCSA](#)'s recently installed Lincoln is a cluster of 450 Dell PowerEdge 1855 blade servers running Windows Compute Cluster Server 2003. This cluster debuted at #130 on the [Top500](#) list in June 2006.
- [gridMathematica](#) provides distributed computations over clusters including data analysis, computer algebra and 3D visualization. It can make use of other technologies such as Altair PBS Professional, Microsoft Windows Compute Cluster Server, Platform LSF and Sun Grid Engine.
- [gLite](#) is a set of middleware technologies created by the [Enabling Grids for E-science](#) (EGEE) project.

1.4. Example of clustered web server system

System Name	BlueGene/L
-------------	-------------------

Site	DOE/NNSA/LLNL
System Family	IBM BlueGene
System Model	BlueGene/L
Computer	eServer Blue Gene Solution
Vendor	IBM
URL	http://www.llnl.gov/asc/comput...
Application area	Not Specified
Main Memory	73728 GB
Installation Year	2007
Operating System	CNK/SLES 9
Memory	73728 GB
Interconnect	Proprietary
Processor	PowerPC 440 700 MHz (2.8 GFlops)

- Housed in Lawrence Livermore National Laboratory's Terascale Simulation Facility, BlueGene/L (BGL) clocked 478.2 trillion floating operations per second (teraFLOPS) on LINPACK, the industry standard of measure for high-performance computing. Built by IBM, BGL is a workhorse supercomputer used to make possible science simulation of unprecedented detail for NNSA's tri-lab Advanced Simulation and Computing (ASC) Program, which leverages the computing expertise and resources of Sandia, Los Alamos and Lawrence Livermore national laboratories.
 - Computer simulations are a cornerstone of NNSA's program to ensure the safety, security and reliability of the nation's nuclear deterrent without underground testing – stockpile stewardship.
 - Recently expanded to accommodate growing demand for high-performance systems able to run the most complex nuclear weapons science calculations, BGL now has a peak speed of 596 teraFLOPS. In partnership with IBM, the machine was scaled up from 65,536 to 106,496 nodes in five rows of racks; the 40,960 new nodes have double the memory of those installed in the original machine.
 - The upgrading of BGL, notably through the addition of nodes with twice the memory, allows scientists from the three nuclear weapons labs to develop and explore a broader set of applications than the single package weapons science oriented work that has been the mainstay of the machine in the past. For example, BGL had been used widely for materials science calculations such as assessing materials at extreme temperatures and pressures. Now it will be much easier to run more complex applications
-
-

related to modeling integrated systems as opposed to focused exploration of one area of physics or chemistry.

Site Name	ECMWF
URL	http://www.ecmwf.int/
Segment	Research
City	Reading
State	N/A
Country	United Kingdom

- The European Centre for Medium-Range Weather Forecasts (ECMWF, the Centre) is an international organisation supported by 28 European States, based in Reading, west of London, in the United Kingdom.
 - ECMWF provides state-of-the-art weather forecast data and products to its Member States as well as managing a super-computer facility which provides resources for weather forecasting research and computer modelling of the global weather atmosphere and ocean.
 - ECMWF Member States are allocated a proportion of ECMWF's supercomputing resources and have access to its data archives.
-
-

2. Load balancing

2.1. What is Load balancing

- In [computer networking](#), **load balancing** is a technique to spread work between two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, and minimize response time. Using multiple components with load balancing, instead of a single component, may increase reliability through [redundancy](#). The balancing service is usually provided by a dedicated program or hardware device (such as a [multilayer switch](#)).
- It is commonly used to mediate internal communications in [computer clusters](#), especially [high-availability clusters](#).

2.2. For Internet Service

- One of the most common applications of load balancing is to provide a single [Internet](#) service from multiple [servers](#), sometimes known as a [server farm](#). Commonly load-balanced systems include popular [web sites](#), large [Internet Relay Chat](#) networks, high-bandwidth [File Transfer Protocol](#) sites, [NNTP](#) servers and [DNS](#) servers.
 - For Internet services, the load balancer is usually a software program which is listening on the [port](#) where external clients connect to access services. The load balancer forwards requests to one of the "backend" servers, which usually replies to the load balancer. This allows the load balancer to reply to the client without the client ever knowing about the internal separation of functions. It also prevents clients from contacting backend servers directly, which may have security benefits by hiding the structure of the internal network and preventing attacks on the kernel's network stack or unrelated services running on other ports.
 - Some load balancers provide a mechanism for doing something special in the event that all backend servers are unavailable. This might include forwarding to a backup load balancer, or displaying a message regarding the outage.
 - An alternate method of load balancing which does not necessarily require a dedicated software or hardware node, is called [round robin DNS](#). In this technique, multiple [IP addresses](#) are associated with a single [domain name](#) (i.e. [www.example.org](#)); clients themselves are expected to choose which server to connect. Unlike the use of a dedicated load balancer, this technique is not "opaque" to clients, because it exposes the existence of multiple backend servers. The technique has other advantages and disadvantages, depending on the degree of control over the DNS server and the granularity of load balancing which is desired.
-
-

-
-
- A variety of [scheduling algorithms](#) are used by load balancers to determine which backend server to send a request to. Simple algorithms include random choice or [round robin](#). More sophisticated load balancers may take into account additional factors, such as a server's reported load, recent response times, up/down status (determined by a monitoring poll of some kind), number of active connections, geographic location, capabilities, or how much traffic it has recently been assigned. High-performance systems may use multiple layers of load balancing.
 - In addition to using dedicated hardware load balancers, software-only solutions are available, including open source options. Examples of the latter include the [Apache](#) web server's `mod_proxy_balancer` extension and the [Pound](#) reverse proxy and load balancer.

2.3.Persistence

- An important issue when operating a load-balanced service is **how to handle information that must be kept across the multiple requests in a user's session**. If this information is stored locally on one back end server, then subsequent requests going to different back end servers would not be able to find it. This might be cached information that can be recomputed, in which case load-balancing a request to a different back end server just introduces a performance issue.
 - One solution to the session data issue is to send all requests in a user session consistently **to the same back end server**. This is known as "persistence" or "stickiness". A large downside to this technique is its lack of automatic [failover](#): if a backend server goes down, its per-session information becomes inaccessible, and sessions depending on it are lost. Interestingly enough, the very same problem is usually relevant to central database servers, even if web servers are "stateless" and not "sticky", central database is (see below).
 - Assignment to a particular server might be based on a username, client [IP address](#), or random assignment. Due to [DHCP](#), [Network Address Translation](#), and [web proxies](#), the client's IP address may change across requests, and so this method can be somewhat unreliable. Random assignments must be remembered by the load balancer, which creates a storage burden. If the load balancer is replaced or fails, this information can be lost, and assignments may need to be deleted after a timeout period or during periods of high load, to avoid exceeding the space available for the assignment table. The random assignment method also requires that clients maintain some state, which can be a problem, for example when a web browser has disabled storage of cookies. Sophisticated load balancers use multiple persistence techniques to avoid some of the shortcomings of any one method.
-
-

-
-
- Another solution is to keep the per-session data in a [database](#). Generally this is bad for performance since it increases the load on the database: the database is best used to store information less transient than per-session data. (Interestingly, to prevent a database from becoming a [single point of failure](#), and to improve [scalability](#), the database is often replicated across multiple machines, and [load balancing](#) is used to spread the query load across those replicas.)
 - Fortunately there are more efficient approaches. In the very common case where the client is a web browser, per-session data can be stored in the browser itself. One technique is to use a browser cookie, suitably time-stamped and encrypted. Another is [URL rewriting](#). Storing session data on the client is generally the preferred solution: then the load balancer is free to pick any backend server to handle a request. However, this method of state-data handling is not really suitable for some complex business logic scenarios, where session state payload is very big or recomputing it with every request on a server is not feasible.

2.4. Load balancer features

Hardware and software load balancers can come with a variety of special features.

1. **Asymmetric load:** A ratio can be manually assigned to cause some backend servers to get a greater share of the workload than others. This is sometimes used as a crude way to account for some servers being faster than others.
 2. **Priority activation:** When the number of available servers drops below a certain number, or load gets too high, standby servers can be brought online
 3. **SSL Offload and Acceleration:** [SSL](#) applications can be a heavy burden on the resources of a Web Server, especially on the CPU and the end users may see a slow response (or at the very least the servers are spending a lot of cycles doing things they weren't designed to do). To resolve these kinds of issues, a Load Balancer capable of handling SSL Offloading in specialized hardware may be used. When Load Balancers are taking the SSL connections, the burden on the Web Servers is reduced and performance will not degrade for the end users.
 4. **Distributed Denial of Service (DDoS) attack protection:** load balancers can provide features such as [SYN cookies](#) and delayed-binding (the back-end servers don't see the client until it finishes its TCP handshake) to mitigate [SYN flood](#) attacks and generally offload work from the servers to a more efficient platform.
 5. **HTTP compression:** reduces amount of data to be transferred for HTTP objects by utilizing gzip compression available in all modern web browsers
 6. **TCP offload:** different vendors use different terms for this, but the idea is that normally each HTTP request from each client is a different TCP
-
-

-
-
- connection. This feature utilizes HTTP/1.1 to consolidate multiple HTTP requests from multiple clients into a single TCP socket to the back-end servers.
7. **TCP buffering:** the load balancer can buffer responses from the server and spoon-feed the data out to slow clients, allowing the server to move on to other tasks.
 8. **Direct Server Return:** an option for asymmetrical load distribution, where request and reply have different network paths.
 9. **Health checking:** the balancer will poll servers for application layer health and remove failed servers from the pool.
 10. **HTTP caching:** the load balancer can store static content so that some requests can be handled without contacting the web servers.
 11. **Content Filtering:** some load balancers can arbitrarily modify traffic on the way through.
 12. **HTTP security:** some load balancers can hide HTTP error pages, remove server identification headers from HTTP responses, and encrypt cookies so end users can't manipulate them.
 13. **Priority queuing:** also known as [rate shaping](#), the ability to give different priority to different traffic.
 14. **Content aware switching:** most load balancers can send requests to different servers based on the URL being requested.
 15. **Client authentication:** authenticate users against a variety of authentication sources before allowing them access to a website.
 16. **Programmatic traffic manipulation:** at least one load balancer allows the use of a scripting language to allow custom load balancing methods, arbitrary traffic manipulations, and more.
 17. **Firewall:** Direct connections to backend servers are prevented, for network security reasons.

2.5. Implement Load balancing

2.5.1. Local DNS Caching

Once the local DNS receives the reply, it will cache that information for a specified time, known as time to live (TTL). TTL is specified by the authoritative DNS as part of its reply. That means, the local DNS will simply reply to all subsequent requests with the information it has from the earlier DNS reply until the TTL expires. Once the TTL expires, the next request to the local DNS will trigger a request to the authoritative DNS again. Caching helps ensure faster response time for the same name to address resolution queries from subsequent clients. At the same time, TTL helps ensure that the local DNS captures any updates or changes from the authoritative DNS. Changing the TTL to a lower value causes the local DNS to query the authoritative DNS more often. Changing the TTL to a higher value puts the local DNS at the risk of having stale information for increased durations.

If the local DNS receives multiple IP addresses as part of the DNS reply, it may give one IP address to each of its clients in a round-robin manner. In addition to the local DNS caching the DNS responses, the client browser also caches the DNS response.

Unfortunately, popular client browsers currently ignore the TTL set by the authoritative DNS. Versions 3.x of Microsoft Internet Explorer, for example, cache the DNS response for 24 hours. Unless the browser application is terminated and restarted, it does not query the DNS again for 24 hours for a given domain.

Versions 4.x and later cache the DNS response for 30 minutes. Microsoft provides a note on the support section of its Web site on how to change the cache time-out value for Internet Explorer by modifying certain entries in the registry. (Search for keywords *ie cache dns timeout* in the support section of Microsoft's Website).

2.5.2.Using Standard DNS for load balancing

DNS can be used for load balancing across multiple servers using the standard round-robin mechanism available in the DNS servers. Each IP address configured for the domain name may actually be a VIP on a load balancer that's bound to several servers connected to the load balancer. DNS can be used for some rudimentary load balancing across the various individual servers or multiple load balancers at different sites where each load balancer performs server load balancing.

But the DNS has no knowledge of which of the different IP addresses is actually working or how much load is on each one of those sites. A site may be completely inaccessible, but the DNS may continue to provide that IP address as part of its reply. We can't view this as a shortcoming of the DNS architecture because DNS was never designed for GSLB. It was devised as a way to provide the name-to-address translation.

2.5.3.HTTP Redirect

One approach that can be used with no changes to the existing DNS system or configuration is a method called *HTTP redirect*. The protocol definition for HTTP includes a way for a Web server to reply with an HTTP response that contains a redirect error code and the redirected URL. This informs the browser that it must go to the new URL in order to get the information it's looking for. Figure 5.5 shows how HTTP redirect works. When a user types *http://www.foo.com/* the local DNS resolves the name *http://www.foo.com/* to the IP address of a Web server in New York. When the browser makes the HTTP request, the Web server in New York redirects the browser to *http://www1.foo.com/*. The browser goes to the local DNS again to resolve the name *www1.foo.com* to an IP address that is in San Jose. Finally, the browser makes the HTTP request to the server in San Jose and retrieves the Web page content. The server in New York can decide whether and where to redirect the user, based on different parameters or policies.

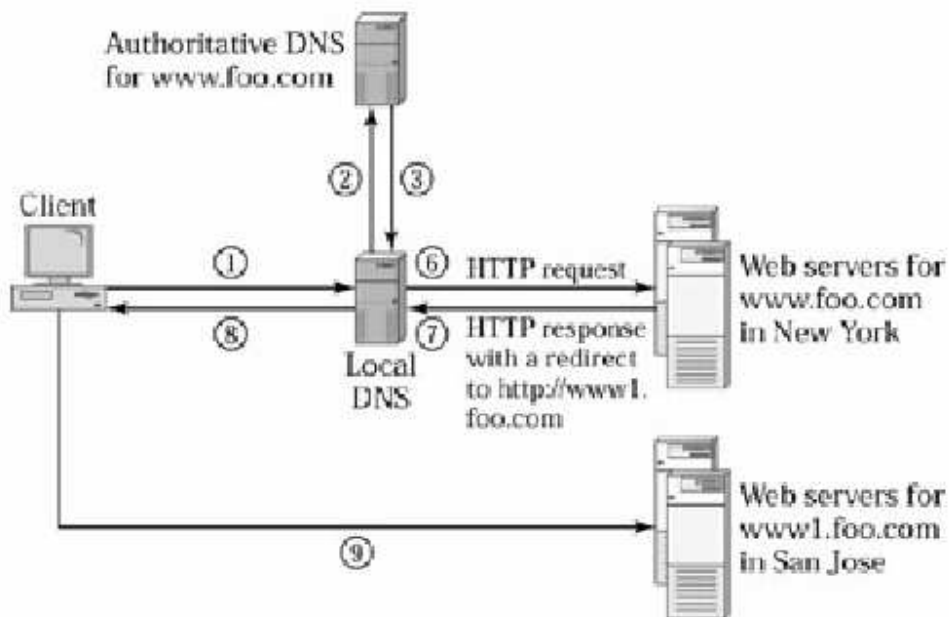


Figure 5.5: HTTP redirect.

The advantages of the HTTP redirect method.

- There is no change to any of the existing DNS setup or configuration.
- When the server in New York gets the HTTP request, it knows the client's IP address, which can be helpful.

The disadvantages of the HTTP redirect method.

As the name indicates, this method works only for HTTP applications and not for any others.

The initial response time now increases as it includes an additional DNS lookup for `www1.foo.com`, establishing a TCP connection with `www1.foo.com` and sending the HTTP request again.

Since all users must first go to `http://www.foo.com/` this may become a performance or reliability bottleneck, although this can be alleviated a bit by using standard DNS-based round-robin load balancing.

2.6. Some Algorithm for Load balancing

2.6.1. Random

- This load balancing method randomly distributes load across the servers available, picking one via random number generation and sending the current connection to it. While it is available on many load balancing products, its usefulness is questionable except where uptime is concerned – and then only if you detect down machines.
- *Plain Programmer Description:* The system builds an array of Servers being load balanced, and uses the random number generator to determine who gets the next connection... Far from an elegant solution, and most often found in large software packages that have thrown load balancing in as a feature.

2.6.2. Round Robin

- Round Robin passes each new connection request to the next server in line, eventually distributing connections evenly across the array of machines being load balanced. Round Robin works well in most configurations, but could be better if the equipment that you are load balancing is not roughly equal in processing speed, connection speed, and/or memory.
- *Plain Programmer Description:* The system builds a standard circular queue and walks through it, sending one request to each machine before getting to the start of the queue and doing it again. While I've never seen the code (or actual load balancer code for any of these for that matter), we've all written this queue with the modulus function before. In school if nowhere else.

2.6.3. Weighted Round Robin (called Ratio on the BIG-IP)

- With this method, the number of connections that each machine receives over time is proportionate to a ratio weight you define for each machine. This is an improvement over Round Robin because you can say “Machine 3 can handle 2x the load of machines 1 and 2”, and the load balancer will send two requests to machine #3 for each request to the others.
 - *Plain Programmer Description:* The simplest way to explain for this one is that the system makes multiple entries in the Round Robin circular queue for servers with larger ratios. So if you set ratios at 3:2:1:1 for your four servers, that's what the queue would look like – 3 entries for the first server, two for the second, one each for the third and fourth. In this version, the weights are set when the load balancing is configured for your application and never change, so the system will just keep looping through that circular queue. Different vendors use different weighting systems – whole numbers, decimals that must total 1.0 (100%), etc. but this is an
-
-

implementation detail, they all end up in a circular queue style layout with more entries for larger ratings.

2.6.4. Dynamic Round Robin (Called Dynamic Ratio on the BIG-IP)

- This is similar to Weighted Round Robin, however, weights are based on continuous monitoring of the servers and are therefore continually changing. This is a dynamic load balancing method, distributing connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time. This *Application Delivery Controller* method is *rarely* available in a simple load balancer.
- *Plain Programmer Description:* If you think of Weighted Round Robin where the circular queue is rebuilt with new (dynamic) weights whenever it has been fully traversed, you'll be dead-on.

2.6.5. Fastest

- The Fastest method passes a new connection based on the fastest response time of all servers. This method may be particularly useful in environments where servers are distributed across different logical networks. On the BIG-IP, only servers that are active will be selected.
- *Plain Programmer Description:* The load balancer looks at the response time of each attached server and chooses the one with the best response time. This is pretty straight-forward, but can lead to congestion because response time *right now* won't necessarily be response time in 1 second or two seconds. Since connections are generally going through the load balancer, this algorithm is a lot easier to implement than you might think, as long as the numbers are kept up to date whenever a response comes through.

2.6.6. Least Connections

- With this method, the system passes a new connection to the server that has the least number of current connections. Least Connections methods work best in environments where the servers or other equipment you are load balancing have similar capabilities. This is a dynamic load balancing method, distributing connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time. This *Application Delivery Controller* method is *rarely* available in a simple load balancer.
 - *Plain Programmer Description:* This algorithm just keeps track of the number of connections attached to each server, and selects the one with the smallest number to receive the connection. Like fastest, this can cause
-
-

congestion when the connections are all of different durations – like if one is loading a plain HTML page and another is running a JSP with a ton of database lookups. Connection counting just doesn't account for that scenario very well.

2.6.7.Observed

- The Observed method uses a combination of the logic used in the Least Connections and Fastest algorithms to load balance connections to servers being load-balanced. With this method, servers are ranked based on a combination of the number of current connections and the response time. Servers that have a better balance of fewest connections and fastest response time receive a greater proportion of the connections. This *Application Delivery Controller* method is *rarely* available in a simple load balancer.
- *Plain Programmer Description:* This algorithm tries to merge Fastest and Least Connections, which does make it more appealing than either one of the above than alone. In this case, an array is built with the information indicated (how weighting is done will vary, and I don't know even for F5, let alone our competitors), and the element with the highest value is chosen to receive the connection. This somewhat counters the weaknesses of both of the original algorithms, but does not account for when a server is about to be overloaded – like when three requests to that query-heavy JSP have just been submitted, but not yet hit the heavy work.

2.6.8.Predictive

- The Predictive method uses the ranking method used by the Observed method, however, with the Predictive method, the system analyzes the trend of the ranking over time, determining whether a servers performance is currently improving or declining. The servers in the specified pool with better performance rankings that are currently improving, rather than declining, receive a higher proportion of the connections. The Predictive methods work well in any environment. This *Application Delivery Controller* method is *rarely* available in a simple load balancer.
- *Plain Programmer Description:* This method attempts to fix the one problem with Observed by watching what is happening with the server. If its response time has started going down, it is less likely to receive the packet. Again, no idea what the weightings are, but an array is built and the most desirable is chosen.

2.6.9.Locality Aware Request Distribution

- With content based request distribution the front end takes into account both the service content requested and the current load on the backend nodes when deciding which backend node should serve a given request. □
- The potential advantages of content based request distribution are increased performance due to improved hit rates in the backend's main memory caches increased secondary storage scalability due to the ability to partition the server's database over the different backend nodes and the ability to employ backend nodes that are specialized for certain types of requests eg audio and video.
- The locality aware request distribution □LARD strategy presented in this paper is a form of content based request distribution focusing on obtaining the first of the advantages cited above namely improved cache hit rates in the backends.

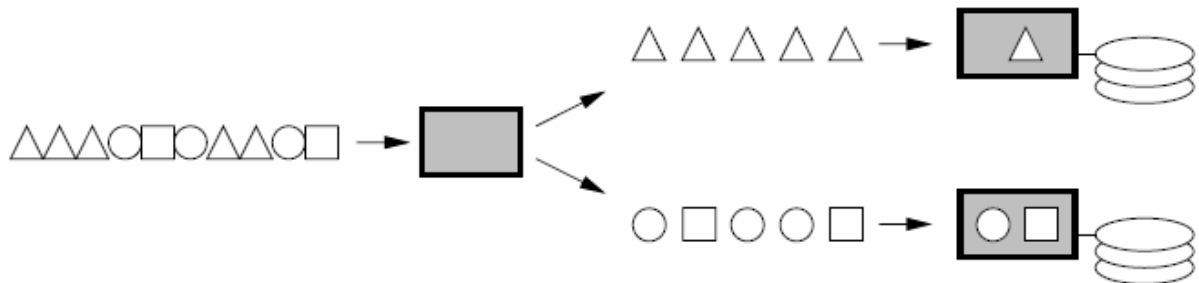


Figure 1: Locality-Aware Request Distribution

- Figure illustrates the principle of LARD in a simple server with two back ends and three targets (A.B.C) in the incoming request stream. The front end directs all requests for A to back end 1 and all requests for B and C to back end 2. By doing so there is an increased like lihood that the request □ finds the requested target in the cache at the back end. In contrast with a round robin distribution of incoming requests requests of all three targets will arrive at both backends.This increases the likelihood of a cache miss if the sum of the sizes of the three targets or more generally if the size of the working set exceeds the size of the main memory cache at an individual backend node.

2.6.10.Genetic Based GDE Approach in Load Balancing

The aim of this method is to compare and equalize each neighbor server workload in an arbitrary web cluster topology until the workload distribution reaches the balance stage. The first step of GDE approach is using edge-coloring method to determine the dimension indices. The dimension is defined as edges with the same color while the iterative process for all dimensions in the corresponding system topology is defined as a sweep. Based on the predefined dimension, neighbor server workload exchanged is equalized along the order dimension. The exchange parameter of the GDE method will govern the workload amount for each server. Optimal sets of exchange parameters have been selected using genetic

algorithm to accelerate the system to achieve the stable stage. The structure of the process flow is illustrated in.

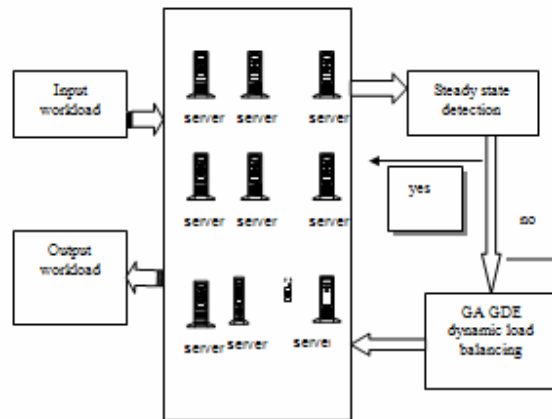


Figure 1: Genetic Based GDE Approach Process Flow

3.Round robin DNS

3.1.What 's round robin DNS

- **Round robin DNS** is a technique of [load distribution](#), load balancing, or [fault-tolerance](#) provisioning multiple, redundant [Internet Protocol](#) service hosts, e.g., [Web servers](#), [FTP servers](#), by managing the [Domain Name System](#)'s (DNS) responses to address requests from client computers according to an appropriate statistical model.
- In its simplest implementation Round-robin DNS works by responding to DNS requests not only with a single [IP address](#), but a list of IP addresses of several servers that host identical services. The order in which IP addresses from the list are returned is the basis for the term *round robin*. With each DNS response, the IP address sequence in the list is [permuted](#). Usually, basic IP clients attempt connections with the first address returned from a DNS query so that on different connection attempts clients would receive service from different providers, thus distributing the overall load among servers.
- There is no standard procedure for deciding which address will be used by the requesting application - a few resolvers attempt to re-order the list to give priority to numerically "closer" networks. Some desktop clients do try alternate addresses after a connection timeout of 30-45 seconds.
- Round robin DNS is often used for balancing the load of geographically-distributed [Web servers](#). For example, a company has one domain name and three identical web sites residing on three servers with three different IP addresses. When one user accesses the home page it will be sent to the first IP address. The second user who accesses the home page will be sent to the next IP address, and the third user will be sent to the third IP address. In each case, once the IP address is given out, it goes to the end of the list. The fourth user, therefore, will be sent to the first IP address, and so forth.
- Many [IRC networks](#) use round robin DNS to distribute users among the servers on their networks. Indeed, virtually all large and established networks have round robin DNS implementations for each continent or country in which they have servers - so users can use a 'random' server but not necessarily one local to them.

3.2.Implement Round Robin DNS

3.2.1.DNS load balancing implementation (Multiple CNAMEs)

- This approach works for BIND 4 name servers, where multiple CNAMEs are not considered as a configuration error. Assuming there are 4 web servers in the cluster configured with IP addresses 123.45.67.[1-4], add all of them to the DNS with Address records (A Names) as below. The srv[1-
-
-

4] can be set to any name you want, such as foo[1-4], but should match the next step.

```
srv1 IN A 123.45.67.1
srv2 IN A 123.45.67.2
srv3 IN A 123.45.67.3
srv4 IN A 123.45.67.4
```

- Add the following canonical names to resolve www.domain.com to one of these servers.

```
www IN CNAME srv1.domain.tld.
IN CNAME srv2.domain.tld.
IN CNAME srv3.domain.tld.
IN CNAME srv4.domain.tld.
```

- The DNS server will resolve the www.domain.com to one of the listed servers in a rotated manner. That will spread the requests over the group of servers.

Note: The requests sent to http://domain.com (without 'www') should be forwarded to http://www.domain.com in this case to work. For BIND 8 name servers, the above approach will throw an error for multiple CNAMEs. This can be avoided by an explicit multiple CNAME configuration option as shown below.

```
options {
multiple-cnames yes;
};
```

3.2.2.DNS load balancing implementation (Multiple A Records)

This above approach with multiple CNAMEs for one domain name is not a valid DNS server configuration for BIND 9 and above. In this case, multiple A records are used.

```
www.domain.tld. 60 IN A 123.45.67.1
www.domain.tld. 60 IN A 123.45.67.2
www.domain.tld. 60 IN A 123.45.67.3
www.domain.tld. 60 IN A 123.45.67.4
```

The TTL value should be kept to a low value, so that the DNS cache is refreshed faster.

3.3.Performance

- While the performance of fast zones isn't exactly stellar, it is not much more than the normal CPU loads induced by BIND. Testing was performed on a Sun Sparc-2 being used as a normal workstation, but no
-
-

resolvers were using the name server - essentially the nameserver was idle. For a configuration with no fast subzones, BIND accrued 11 CPU seconds in 24 hours. For a configuration with one fast zone, six address records, and being refreshed every 300 seconds (5 minutes), BIND accrued 1 minute 4 seconds CPU time. For the same previous configuration, but being refreshed every sixty seconds, BIND accrued 5 minutes and 38 seconds of CPU time.

- As is no great surprise, the CPU load on the serving machine was linear to the frequency of the refresh time. The sixty second refresh configuration used approximately five times as much CPU time as did the 300 second refresh configuration. One can easily extrapolate that the overall CPU utilization would be linear to the number of zones and the frequency of the refresh period. All of this is based on a shell script that always indicated that a zone update was necessary, a more intelligent program should realize when the reordering of the RRs was unnecessary and avoid such periodic zone reloads.

3.4.Drawbacks

- Although easy to implement, round robin DNS has problematic drawbacks, such as those arising from record caching in the DNS hierarchy itself, as well as client-side address caching and reuse, the combination of which can be difficult to manage. Round robin DNS should not solely be relied upon for service availability. If a service at one of the addresses in the list fails, the DNS will continue to hand out that address and clients will still attempt to reach the inoperable service.
 - Also, it may not be the best choice for [load balancing](#) on its own since it merely alternates the order of the address records each time a name server is queried. There is no consideration for matching the user IP address and its geographical location, transaction time, server load, network congestion, etc. Round robin DNS load balancing works best for services with a large number of uniformly distributed connections to servers of equivalent capacity. Otherwise it just does [load distribution](#).
 - Methods exist to overcome such limitations. For example, modified DNS servers (such as [Ibnamed^{\[1\]}](#)) can routinely poll mirrored servers for availability and load factor. If a server does not reply as required, the server can be temporarily removed from the DNS pool, until it reports that it is once again operating within specifications.
-
-

Chapter 11.

1. Web Services

1.1. Concept of WebServices

- A **Web Service** (also **Web Service**, **Webservice**) is defined by the [W3C](#) as "a software system designed to support [interoperable machine-to-machine](#) interaction over a [network](#)".^[1] Web services are frequently just [Internet application programming interfaces](#) (API) that can be accessed over a network, such as the [Internet](#), and executed on a remote system hosting the requested services. Other approaches with nearly the same functionality as web services are [Object Management Group's](#) (OMG) [Common Object Request Broker Architecture](#) ([CORBA](#)), [Microsoft's](#) [Distributed Component Object Model](#) ([DCOM](#)) or [SUN's](#) [Java/Remote Method Invocation](#) ([RMI](#)).
- The [W3C](#) Web service definition encompasses many different systems, but in common usage the term refers to [clients](#) and [servers](#) that communicate over the [HTTP](#) protocol used on the Web. Such services tend to fall into one of two camps: Big Web Services and RESTful Web Services.

1.2. Generic Architecture of WebServices

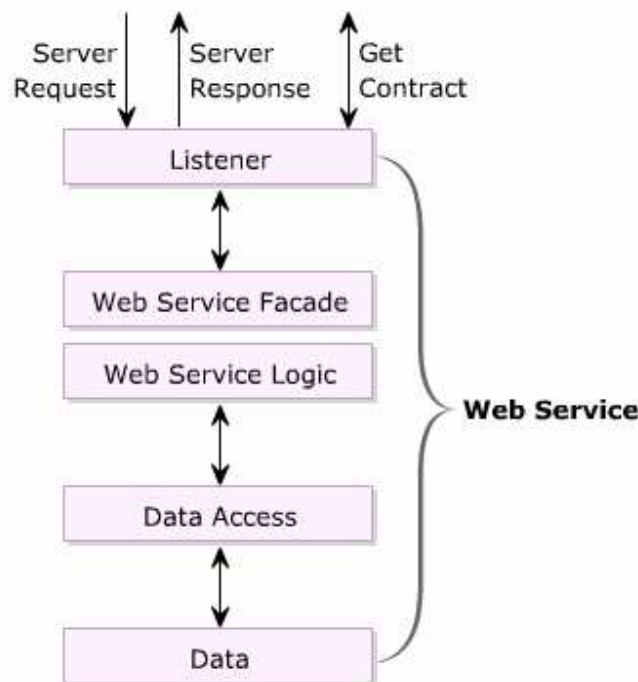


Figure 11.1 Architecture of WebServices

-
-
- Web Services is independent modular components, which do not have a user interface and provide services on LAN, WAN, MAN, and internet. They designed to provide 100 percent interoperability over dissimilar networks as they use TCP/IP, HTTP, and XML for communication.
 - Web Services use Simple Object Access Protocol (SOAP) to communicate with heterogeneous system by exchanging message, SOAP is a platform-independent protocol that use XML to exchange information. A Web Service is the ideal solution to all the problems traditional distributed computing.
 - The architecture of a Web Service helps in understanding the working of the Web Services. There is five layers in generic architecture of Web Services.
 - Data layer stores the information needed by Web Service.
 - Data Access layer maintains data integrity by separating the data sources from the manipulation done to the business logic.
 - Web Service Logic layer consist web server logic.
 - Web Service Façade layer provide interface corresponding to functions provided by the Web Service.
 - Listen Layer interacts with the client applications, accepts the request, and parses it. It then passes the message parameters to the appropriate methods in the business façade. This is the only layer, which is aware that it is a part of Web Service.

1.3.Life Cycle of a Web Service

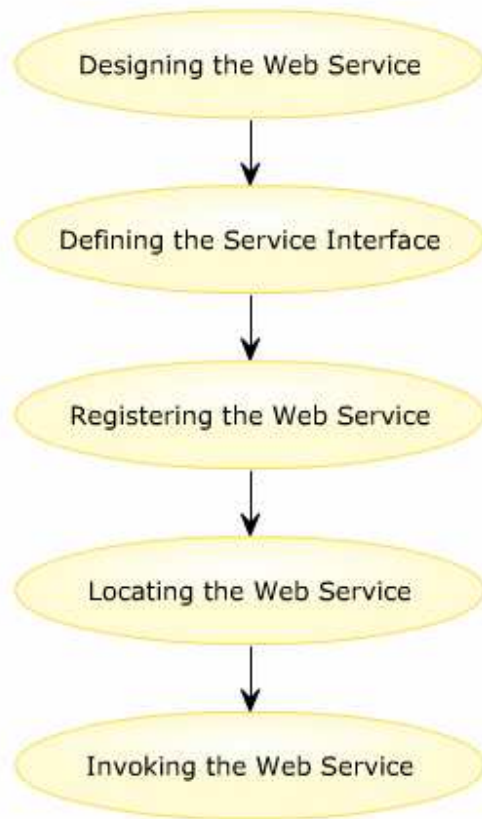


Figure 11.2 Life Cycle of a Web Service

Web Services provide the service interface instead of a user interface. The service interface is an XML document that is used to call the Web Service. Therefore, this interface plays a major role in the lifecycle of a Web Service. The life cycle of a Web Service defines the scope of the Web Service and the activities involved to use it. The tasks involved in the life cycle of a Web Service are

- Design a Web Service.
- Clearly define the service interface and Web Service methods.
- Once the Web Service is created, it should be registered on some central network node to help the Web clients in finding and using the service.
- Now, the Web clients should locate the Web Services on the Web to use them.
- The Web Service should be called.

1.4.XML Web Services

1.4.1.Concept of XML Web Services

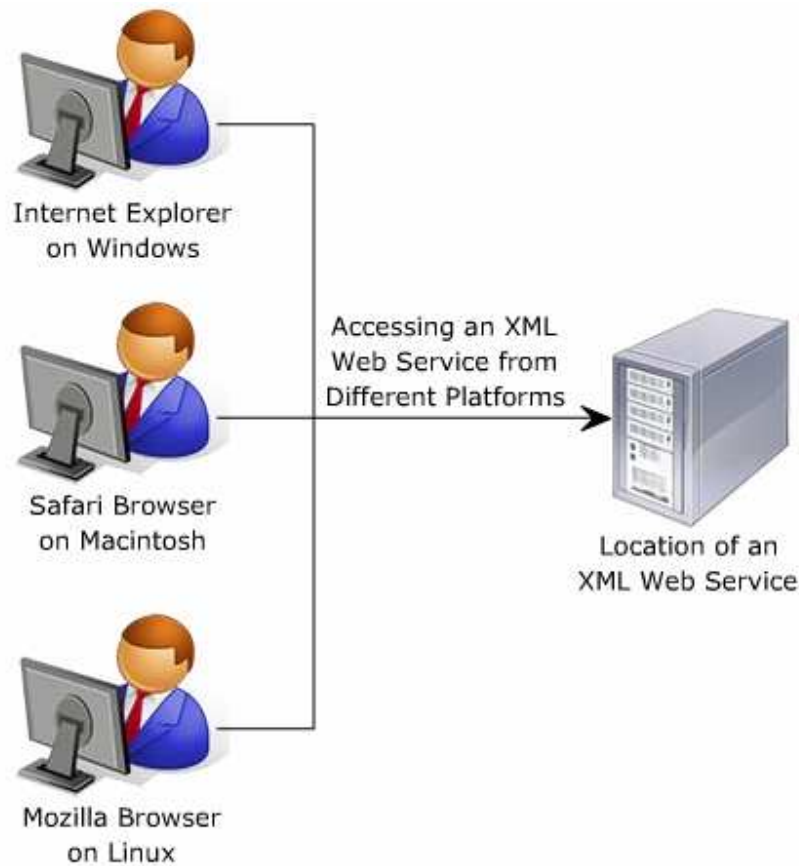


Figure 11.3 XML Web Services

- An XML Web Service is an independent component that communicates with other applications using standard Web protocols such as HTTP, XML, SOAP, and TCP/IP. This allows XML Web Services to communicate with applications written in different languages, operating on different platforms, and communicating with different protocols. Web Services created using ASP.Net are called XML Web Services as they communicate on the network by exchanging messages in XML format.
- Apart from the feature of interoperability, Web Services provides some more features. These features include
 - Stateless architecture
 - Asynchronous architecture
 - Platform and language-independent communication.

1.4.2.XML Web Services Infrastructure

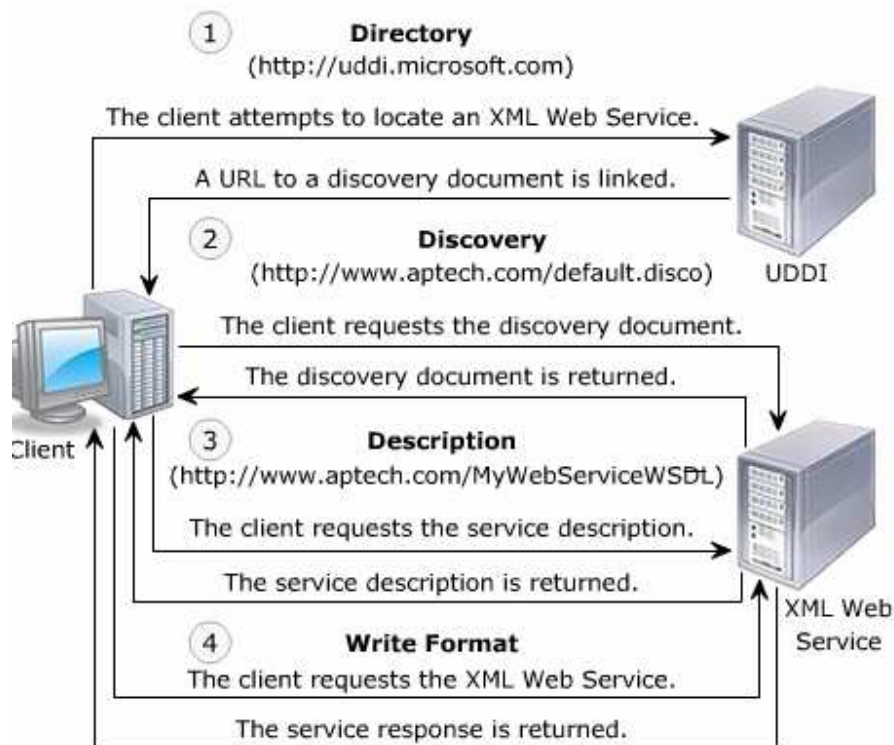


Figure 11.4 XML Web Services Infrastructure

- XML Web Services register themselves on a central network node to make the services available to users. The node is also known as Universal Description, Discovery, and Intergration (UDDI) registry. After registration, Web clients locate XML Web Services by using an XML documents called the Web Services Description Language(WSDL), and the Discovery (DISCO) specification. The WSDL document provides the description about the Web Services, whereas the DISCO specification provides algorithms to locate the Web Service. All these operations are conducted using the XML Web Services infrastructure.
- The infrastructure consists of four different componets to fullfill the operations over the Web. These componets are
 - XML Web Services Directories allow registering Web Services and searching for the registered Web Services using UDDI. Web clients are UDDI to search for the required Web Services.
 - XML Web Service Discovery is a process that allows Web clients to discover documents describing an XML Web Service. It is performed using the WSDL document and the DISCO specification. When the Web client searches uses UDDI to locate a Web Service, the WSDL document is returned as response.

- XML Web Service Description provides the description related to a particular XML Web Service. The description helps in providing the information on how to interact with a particular XML Web Service.
- XML Web Service Wire Formats. XML Web Services communicate using open wire formats. These are platform-independent protocols, which can be interpreted by systems supporting the common Web standards such as HTTP and XML. SOAP is main protocol used extensively used for communicating with XML Web Services.

1.4.3. Client and XML Web Service Communication

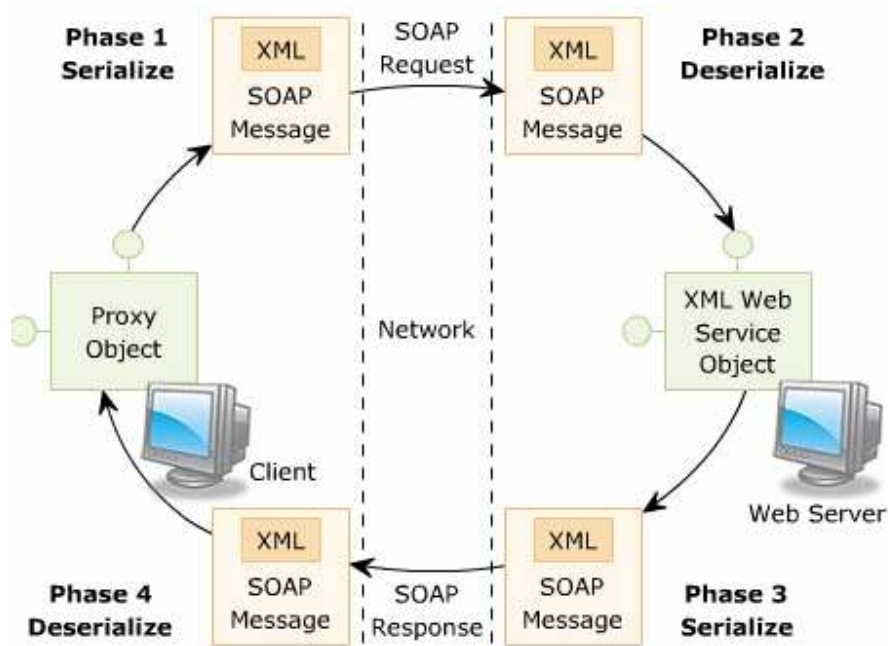


Figure 11.5. Client and XML Web Service Communication

The Web client must perform a few steps to call a Web Service. These steps include

- The client creates an object of the XML Web Service proxy class. The proxy class is an object on the client computer acting as a representative of the service.
- The client calls the method on the proxy class.
- The arguments of the XML Web Service method are serialized into a SOAP message by the XML Web Service infrastructure on the client computer.
- The arguments are sent to the XML Web Service over the network.
- The Web server receives the SOAP message and deserializes it.
- The infrastructure on the server invokes the method by creating an object of class implementing the service and passes the arguments.

-
-
- The method is executed and the value returned to the client using the same process of sending the request message.

1.5 Advantages and Disadvantages compared to the CORBA & RMI

1.5.1. Advantages

- Allowing cross business integration by providing access to third-party softwares and legacy applications irrespective of their platform and programming language.
- Providing increased efficiency, scalability, and maintainability as applications are split into independent smaller components.
- Reducing complexity by providing encapsulation, which frees the clients from knowing the Web Service architecture.
- Ensuring interoperability by using common Web standards.
- Providing on-demand integration of distributed components.

1.5.2. Disadvantages

- No track of activities as HTTP is a stateless protocol.
- Fear of security risks as even critical services such as credit card validations are explored on Web.
- Mandatory XML support without which the Web Services cannot function.
- High cost for deploying Web Services.

1.6. SDL (Service Description Language)

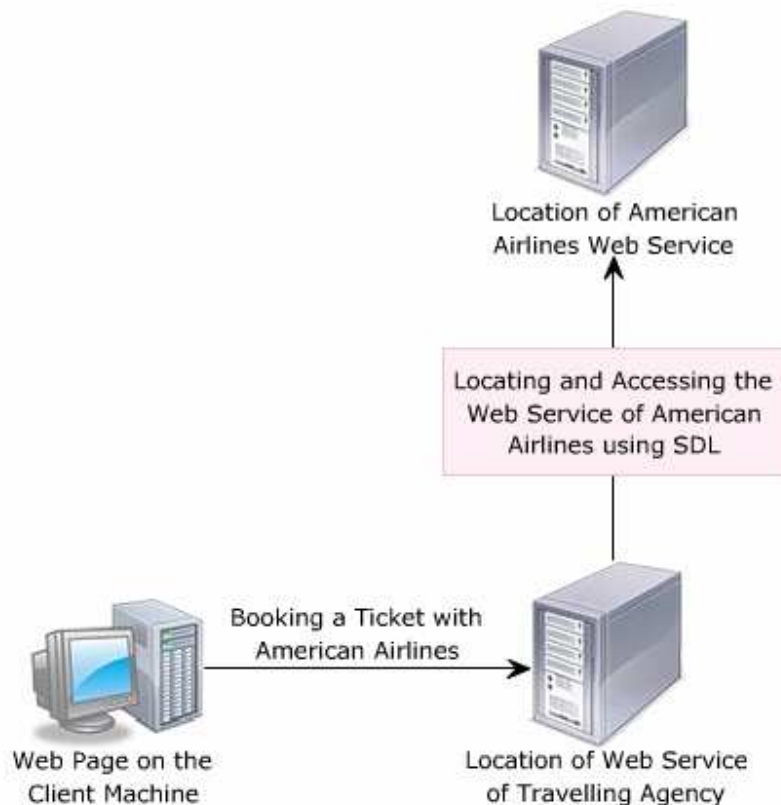


Figure 11.6 SDL

1.6.1 The need of SDL

- Consider a scenario where a traveling agency is planning to create a Web Service for their customers for registering the air tickets online. The customers can book tickets with any airlines across the globe. Therefore, the Web Service must interact with the Web Services of the airline companies to identify the availability of flights and seats on the required date and time.
- It is possible that these Web Services might operate on different operating systems and use different communication protocols. To ensure communication amongst Web Services, each Web Service must define interoperable interfaces and must know about the functionalities provided by each other.
- This can be accomplished by using Service Description Language. SDL is an interoperable language used to describe the Web methods along with the input and output parameters.

1.6.2. Definition of SDL

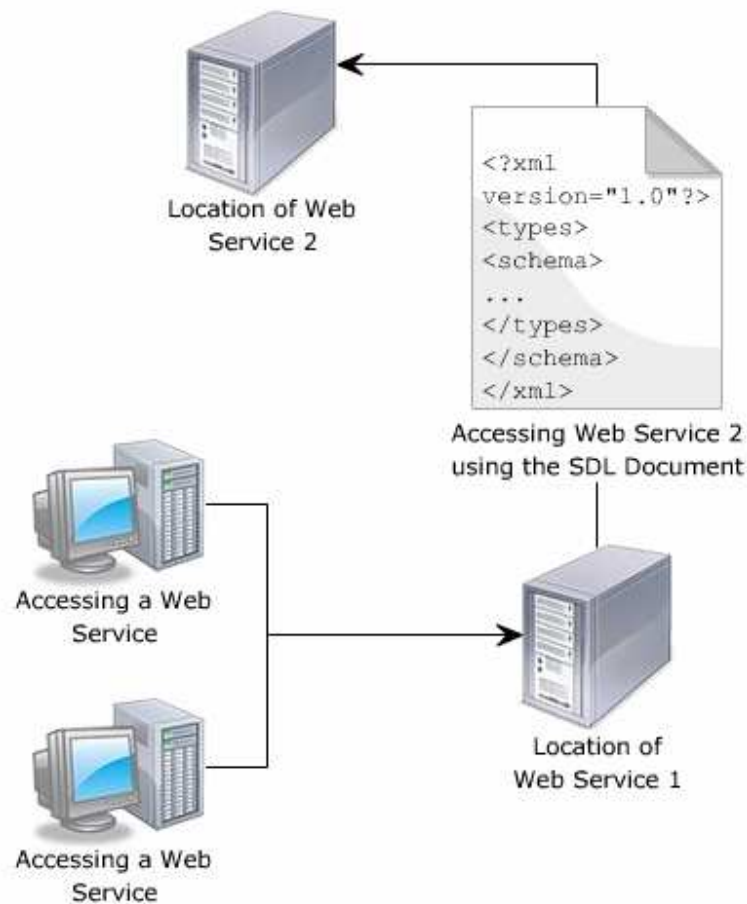


Figure 11.7 SDL in working

- SDL is an XML document used to describe Web Services. It allows other Web Services or applications to acquire information includes method name, data types of parameters , return values, and so on. For example, using SDL in a distributed environment, an application can identify that specific Web Service accepts two interger, multiplies them, and returns the product as an interger.
 - It is similar to Interface Definition Language (IDL) used in CORBA. IDL is a language used to define an application's interface and the common behavior for interacting with the interface. The behavior is defined by specifying methods of the interface, parameters to these methods, their data types, and return type. The only difference between IDL and SDL is that SDL is platform independent as it uses XML.
-
-

1.7.WSDL (Web Service Description Language)

1.7.1.Concept of WSDL

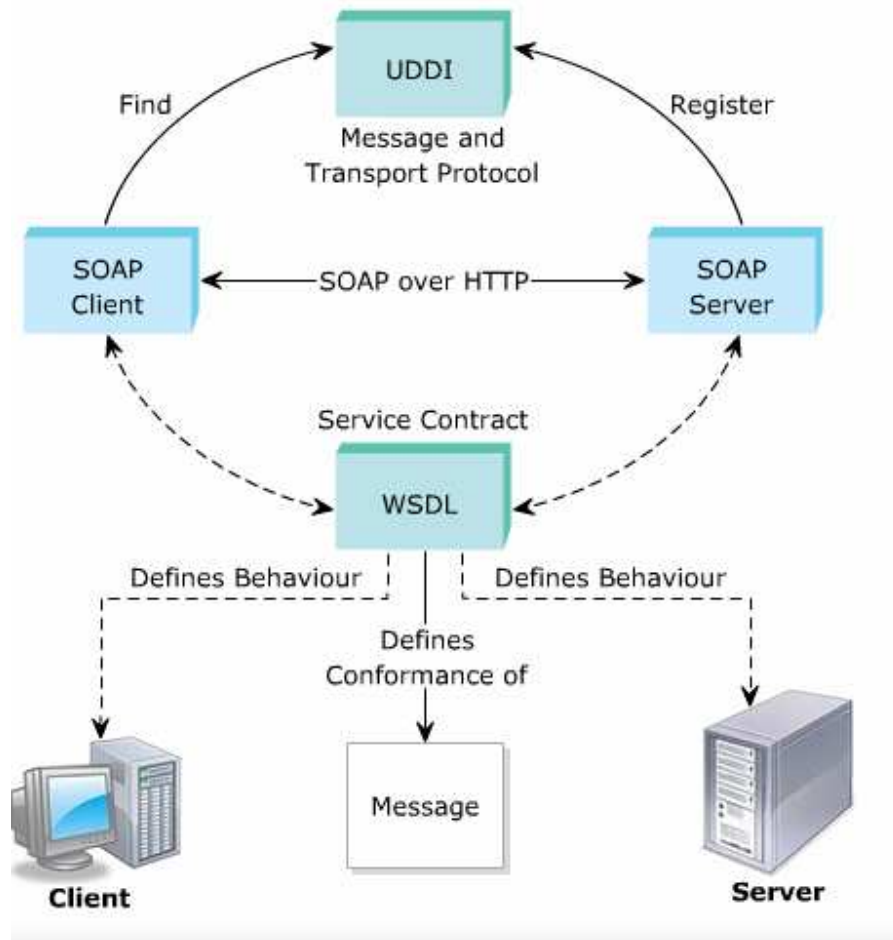


Figure 11.8 WSDL

- WSDL is an IDL used to define the Web Service and Web Service contract. It is an XML document, which contain custom tags used to define the elements and attributes of the service. WSDL also defines a standard manner in which a Web Service can be accessed.
 - When the Web Service consumer queries the Web Service broker to discover a Web Service, the consumer retrieves the WSDL document of the requested Web Service. The document behaves as a contract between the consumer and the Web Service.
 - WSDL is a subset of two files, namely Network Accessible Service Specification Language (NASSL) and Well Defined Service document (WDS). NASSL is an IDL, which uses XML to describe operational data of network-based services. This data includes the service interface and implementation details. WDS document defines non-operational data of a service such as the expiry date and company name.
-
-

1.7.2.Element of a WSDL File

- WSDL defines services as a set of network endpoints. Therefore, a WSDL file consists of service definitions. It uses XML tags to define services, message, methods, protocols, and endpoints. An endpoint is a port, which is a set of network addresses used to enable network communication.
- A WSDL file refers to a method as an operation. The supported operations of a Web Service together are called port types. Each port type specifies a network protocol and a data format for exchanging information. This binds the messages and operations to a protocol and data format, which is referred to as binding. This is how a WSDL file defines a public interface used to access a Web Service.

There are six basic elements

- **definitions** is the root element and contains the default namespaces for the WSDL document. The default namespace of the WSDL document is specified as <http://tempuri.org>. You must change the default namespace before the making of Web Service public.
- **types** defines the data types to be used in the Web Service.
- **message** describes the data to be exchanged on the network.
- **portType** specifies operations for a port. The port address are assigned dynamically, which makes them reusable.
- **binding** describes the communication protocol and data format for portType. Binding refers to a way for the operations to access the service using a certain protocol.
- **service** refers to a collection of related endpoints or ports. A web service is a set of reusable ports.

1.8.SOAP (Simple Object Access Protocol)

1.8.1.Overview SOAP

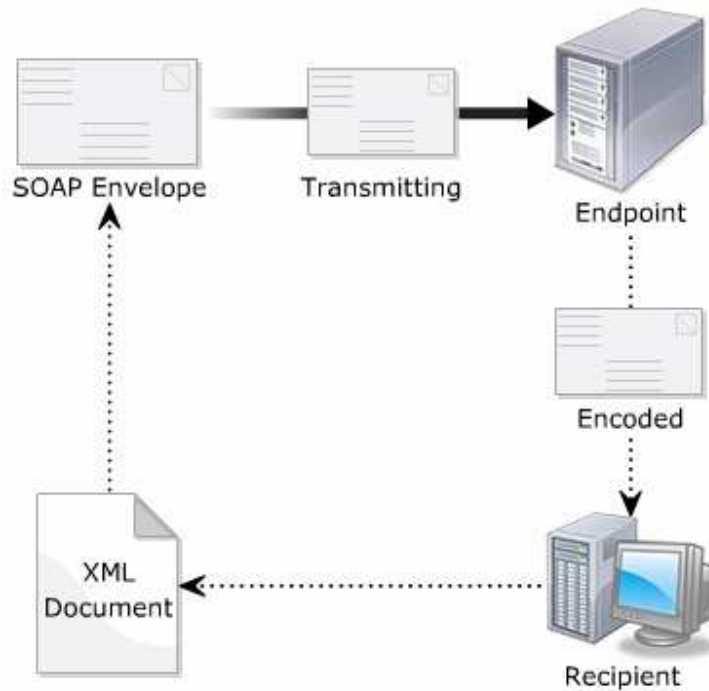


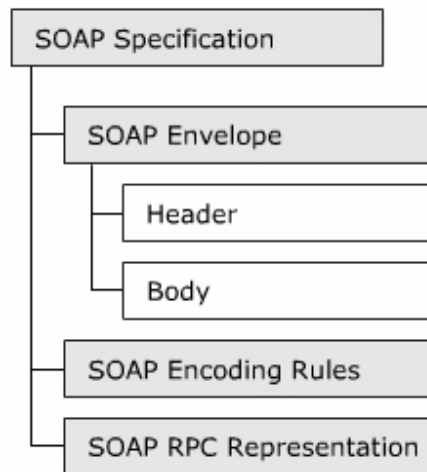
Figure 11.9 SOAP

- SOAP stands for Simple Object Access Protocol. It is a lightweight and XML-base protocol for exchanging information in a distributed enviroment. The goal of SOAP is to provide a well-defined XML packet that can be safely transmitted over any network through through standard protocols such as HTTP, SMTP. SOAP specifies how to invoke remote objects in a sercured manner and is widely used in invoking Web Services.

The SOAP specification is divided into three parts, which are

- SOAP Envelope: Defines the message and identifies who should handle it.
- SOAP Encoding Rules: Defines a mechanism for converting plain-text messages in a secret code that is not reable.
- SOAP RPC Representation: Defines priciples used to handle RPC.

1.8.2.SOAP Specifications



- SOAP Envelope is an important element of XML Document that represents a SOAP message. The SOAP Envelope comprises of an optional header and a mandatory body. The body part of the SOAP Envelope is persistently designed to contain the actual SOAP message intended for the final message recipient. On other hand, the header entries contain the header information that is specific to the application. In addition, the SOAP Envelope tag contains many namespace definitions.
- SOAP Encoding Rules define a set of rules used for exchanging the instances of the datatypes defined by the XML application. SOAP is encoded in very basic style, which illustrates a data type that is independent of the programming language that is used.
- SOAP RPC Representation represents the remote procedure calls and response.

1.8.3.Working of SOAP

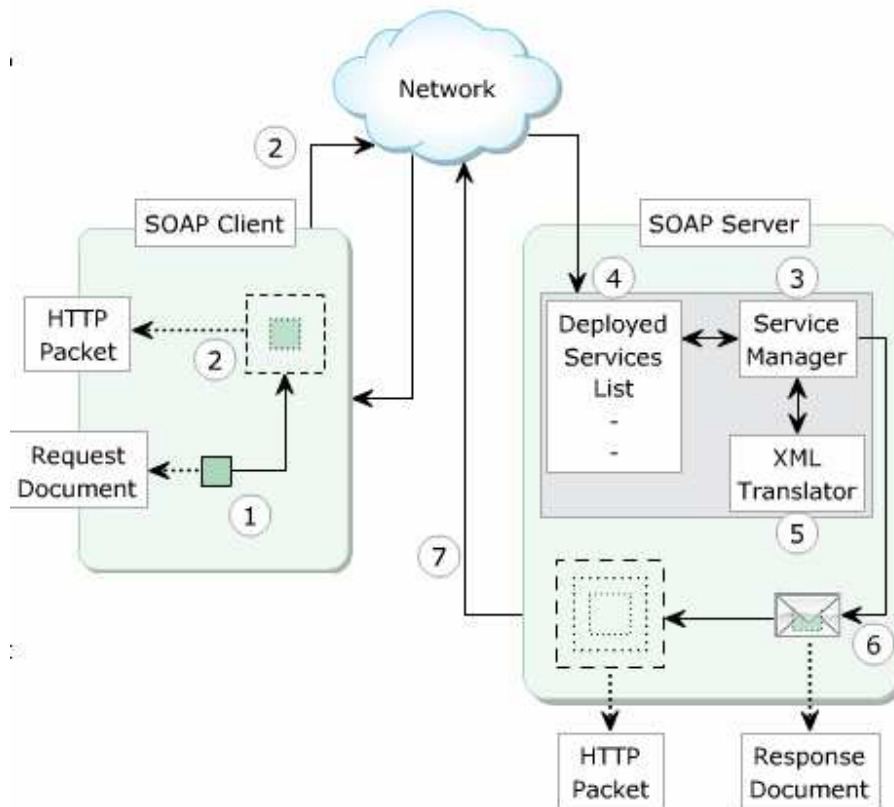
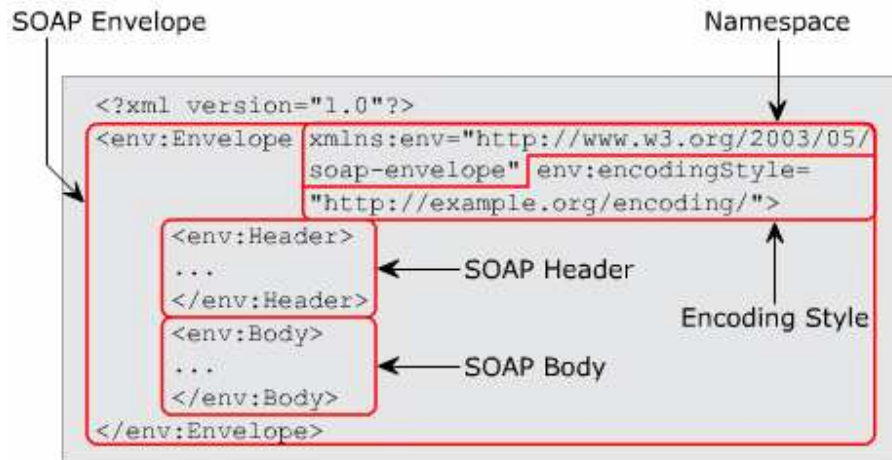


Figure 11.10 Working of SOAP

There are three components involved in communication are the SOAP client, SOAP Server and services. Invoking a service using SOAP and HTTP consists a few steps.

1. The client procedures the SOAP request document that define the parameters of the Web Method.
2. The document is sent as HTTP packet via the network.
3. The Service manager recieves the SOAP packet.
4. The SM searches for the requested service in the developed services list, validates the parameters, and passes the request to the XML Transator.
5. The XML translator passes the parameters to the method by converting them in a programming language implemented for the method. The method is excuted and the return value is sent to the translator.
6. The translator passes the result to the manager, who wraps it as the SOAP response document.
7. The document is sent to the clients as HTTP packet.

1.8.4.SOAP Message Architecture



- A SOAP Message contains an Envelope, an optional Header, and the SOAP Body. The SOAP Envelope is wrapped in any transport protocol such as HTTP. A SOAP Envelope is the mandatory root element of the XML messages. Within the SOAP envelope, there are two elements namely, the SOAP Header and the SOAP Body.
- The SOAP Header is the optional element in a SOAP message. It exists as the immediate child element of the SOAP Envelope. It might contain header entries, which can include information such as authentication details. These entries must be the immediate child elements of the SOAP Header, and must be namespace qualified.
- The SOAP Body consists of the request parameters if it is a SOAP request message. In the SOAP response message, the SOAP body contains the return values of the executed method.

1.8.5.SOAP request message and SOAP response message

```
POST /MiniCalculator/MiniCalculator.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=http://www.w3.org/2003/05/soap-envelope>
  <soap12:Body>
    <SquareRoot xmlns="http://tempuri.org/">
      <number>int</number>
    </SquareRoot>
  </soap12:Body>
</soap12:Envelope>
```

SOAP Request Message

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=http://www.w3.org/2003/05/soap-envelope>
  <soap12:Body>
    <SquareRootResponse xmlns="http://tempuri.org/">
      <SquareRootResult>double</SquareRootResult>
    </SquareRootResponse>
  </soap12:Body>
</soap12:Envelope>
```

SOAP Response Message

- The SOAP envelope element appears in both request and response message. It must be compulsorily encoded using XML. Its attributes and elements are uniquely identified using the mandatory namespace. <http://www.w3c.org/2003/05/soap-envelope>. The SOAP payload must be enclosed in a SOAP Envelope to invoke a remote method.
- The encodingStyle attribute of the SOAP Envelope is used to identify the data types to be used in the document. It can appear with any of SOAP elements and it is applied to the element's data along with its child elements.
- In the SOAP request message, the SOAP body sets the data type of the method parameters using placeholders. Placeholders are variables whose values will be given by the client. The values will be assigned to these variables. Similarly, in the SOAP response message, placeholders in the SOAP Body represent the data type of the return value of the Web method, if any.

11.8.6.SOAP Data Types

- The availability of particular data type relies upon the protocol which is currently being used by the Web Service. For instance, HTTP-GET and HTTP-POST protocols used only name/value pairs. SOAP utilizes XML in a better manner to encode complex data. SOAP supports transmitting parameter using the pass-by-value and pass-by-reference techniques. However, the HTTP GET and HTTP POST protocols support only pass-by-value technique.

Type	Description
Classes and structs, Arrays of classes and structs	Include classes and structs defining public fields and properties, which can be marshalled. It is mandatory for these data structures to define a default constructor without any parameters.
DataSet and Arrays of DataSet	Includes the ADO.NET DataSet types. These DataSet types can also be utilized as fields within classes or structs.
XmlNode, Arrays of XmlNode	XmlNode type is used to represent a XML fragment in the memory. This type can be utilized as fields within classes or structs and can also be transmitted in the form of parameters or return values.

1.9.UDDI

1.9.1.Overview of UDDI

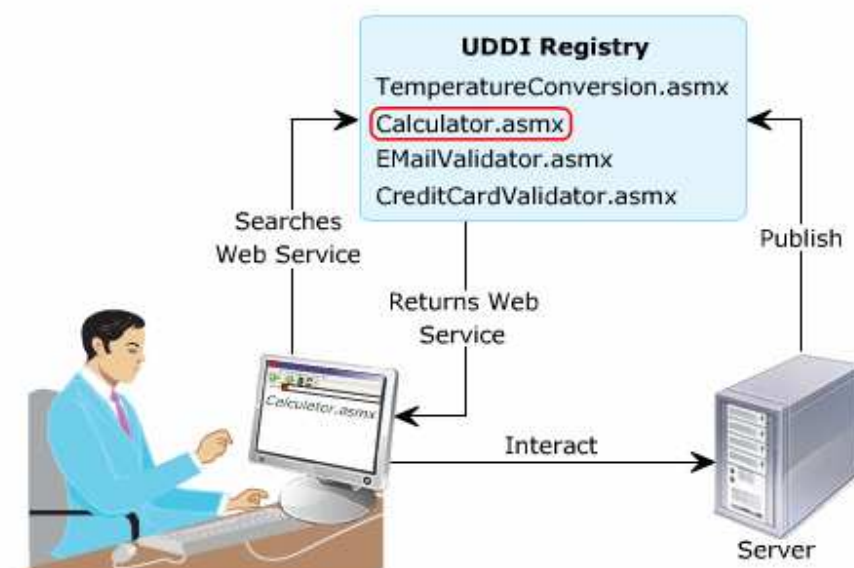


Figure 11.11 UDDI

- UDDI is Web Service Discovery mechanism, It acts as a centralized directory service that allows you to publish and discover Web Services. UDDI consists of a list of published Web Services. It allows the providers to register Web Services, and consumers to locate Web Services.

1.9.2.Purpose

- It provides an easy and efficient way for registering services. This allows the other uses to discover the services. UDDI is platform-independent as it uses XML to store the Web Service details. This means, it provides a standard way for registering. In addition, UDDI determines the security and transport protocols that are supported by the Web Services.
- UDDI contains only information about the Web Service. It does not contain the actual Web Service. The main purpose of UDDI is to enable the service providers to publish information about their Web Service. This information helps the consumer to determine the location of the Web Service and invoke it.

1.9.3.Registry Type

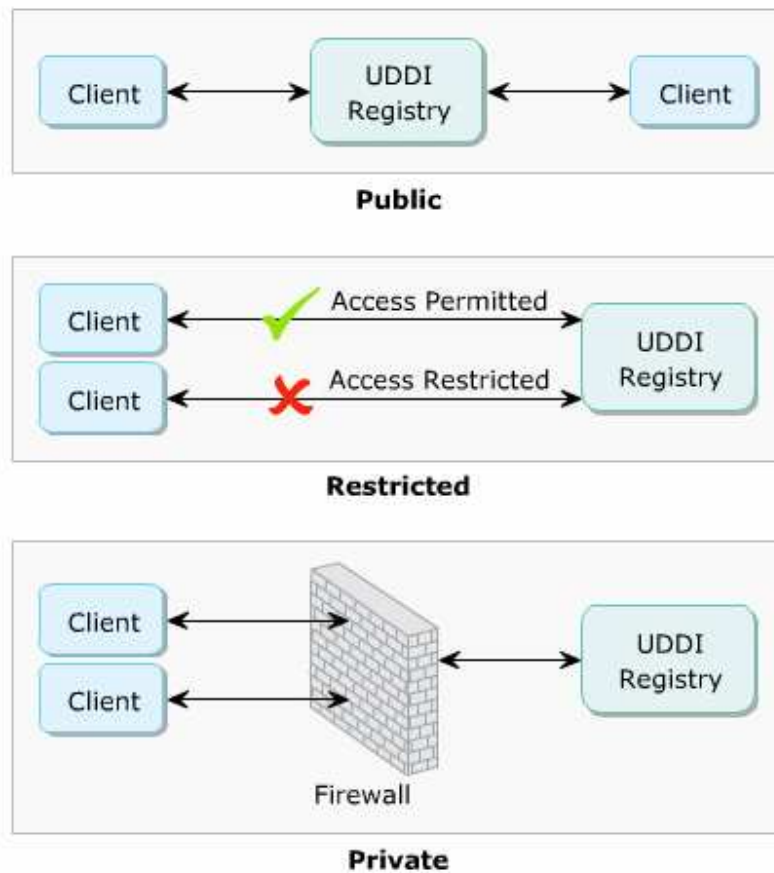
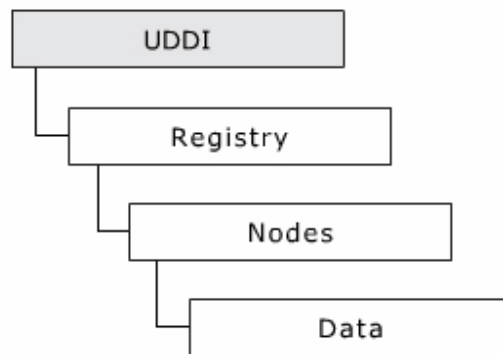


Figure 11.12 Registry Type

- UDDI consists of registries, which contains information about Web Services. The registries are of different types namely, public, private, and restricted. These registry types determine the way the registries are accessible to various clients.
- The public registry is accessible to all clients through Internet. The private registry exists behind the firewall, which exist between the client and the UDDI. The private registry allows you to search Web Services within the Intranet. The restricted registry provides access to only clients, who have permissions to access the request Web Service.

1.9.4.UDDI Architecture



- UDDI is composed of a set data,nodes and registries. The data in the UDDI server represents information, which assists users in obtaining all the required information for using a Web Service. The data exists in XML format to make it accessible by any system on the network and is managed by UDDI nodes. A UDDI node is a server, which performs several operations on UDDI data by accessing and manipulating it. Its functionalities are defined in the UDDI specification.
- To manage data about different Web Services, multiple UDDI nodes are implemented. This collection of the UDDI nodes exists in the UDDI registry. Each node in the registry manages a related set of UDDI data.

1.9.5.Working of UDDI

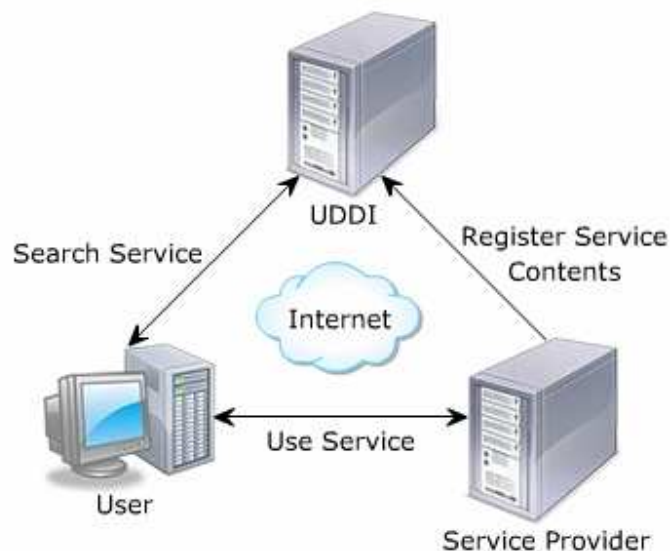


Figure 11.13 Working of UDDI

- UDDI enables the Web Service providers to publish their Web Services. This allow other software clients to locate the Web Service and use it.
-
-

-
- When the client sends a request for accessing the Web Service by providing a URL, the specified Web Service URL is sent to the UDDI server. The UDDI server checks the existence of the Web Service and returns the information about the Web Service. The client then interacts with the Web Service provider for invoking the Web Service.

1.9.6.Data Structures

The data in UDDI is managed using various data structures. These data structures are also called entities, which are represented in an XML format. The table lists the data structures in UDDI.

Data Structure	Description
<code>businessEntity</code>	Provides information about the Web providers that provides one or more Web Services.
<code>businessService</code>	Provides information about a set of related Web Services offered by the Web Service provider, which is described by a <code>businessEntity</code> .
<code>bindingTemplate</code>	Provides technical information that is required to invoke a particular Web Service.
<code>tModel</code>	Provides a technical model, which contains information about a Web Service type or a protocol used by Web Services.
<code>publisherAssertion</code>	Provides information about the relationship that one <code>businessEntity</code> has with another <code>businessEntity</code> .

1.10.XML (See more in next chapter).

- XML is also a text-based markup language that allows the developer to define custom tags. Therefore, other applications can easily interpret the data encoded using XML. It follows the rules and specifications of the Universal Character Set (UCS), which is recognized by the ISO/IEC 10646 specifications.
 - For UCS, the encoding schemes supported by XML are Universal character set Transformation Format UTF-8 and UTF-16. These encoding schemes are acceptable by most software and hardware platforms. This enables the
-

XML data encoded in UTF-8 or UTF-16 to be decoded and interpreted on any platform. Thus XML provides a global platform for network communication amongst heterogeneous applications.

- The fundamental difference between HTML and XML is that XML is used to describe data, and HTML is used to display data. The other difference is that unlike HTML, XML allows you to define your own tags and validates them.



2.XML

2.1.Evolution of XML

- In order to address the issues raised by earlier markup languages, the Extensible Markup Language was created. XML is a W3C recommendation.
- XML is a set of rules for defining semantic tags that break a document into parts and identify the different parts of the document.
- XML was developed over HTML because of the basic differences between them given in the table below.

HTML	XML
HTML was designed to display data.	XML was designed to carry data.
HTML displays data and focuses on how data looks.	XML describes data and focuses on what data is.
HTML displays information.	XML describes information.

2.2.Features of XML

1. XML stands for Extensible Markup Language
2. XML is a markup language much like HTML
3. XML was designed to describe data
4. XML tags are not predefined. You must define your own tags
5. XML uses a Document Type Definition (DTD) or an XML Schema to describe the data
6. XML with a DTD or XML Schema is designed to be self-descriptive.

2.3.XML Markup

- XML markup defines the physical and logical layout of the document.
 - Markup is important because a program will find it difficult to distinguish a piece of text from any other piece, if the document has no labels or boundaries.
 - XML's markup divides a document into separate information containers called elements. A document consists of one outermost element, called root element that contains all the other elements, plus some optional administrative information at the top, known as XML declaration.
 - XML can be used to create other languages like WAP and WML.
-
-

2.4. Benefits of XML

- **Data Independence** is essential characteristic of XML. It separates the content from its presentation. Since an XML document describes data, it can be processed by any application.
- **Easier to parse.** The absence of formatting instructions make it easy to parse. This makes XML an ideal framework for data exchange.
- **Reducing Server Load.** Because XML's semantic and structural information enables it to be manipulated by any application, much of the processing that was once earlier limited to servers can now be performed by clients. This reduces server load and networkd traffic, resulting in a faster, more efficient Web.
- **Easier to create.** It is text-based, so it is easy to create an XML document, with even the most primitive text processing tools. However, XML also can describe images, vector graphics, animation or any other data type to which it is extended.
- **Web Site Content.** The W3C uses XML to write its specifications and tranforms it to a number of other presentation formats. Some Web Site also use XML for their content and get it tranformed to HTML using XSLT and CSS and display the content directly in browsers.
- **Remote Procedure Calls.** RPC is a protocol that allows objects on one computer to call objects on another computer to do work, allowing distributed computing.
- **e-Commerce.** XML can be used as an exchange format in order to send data form one company to another.

2.5.XML Document Structure

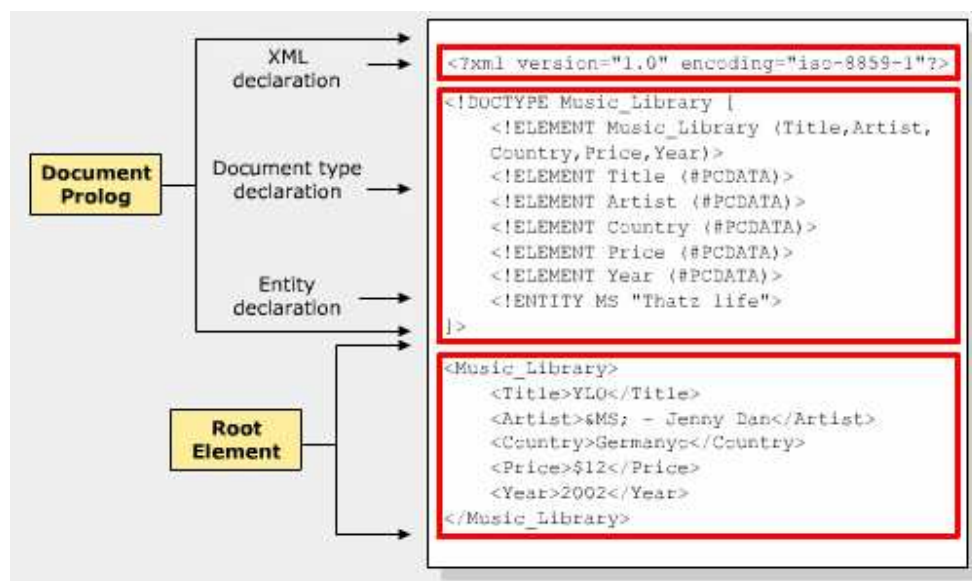


Figure 12.1 XML Document

- XML documents are commonly stored in text files with extension .xml
- The two sections of an XML document are Document Prolog and Root Element.
- Document Prolog. XML parser get information about the content in the document with the help of document prolog. Document prolog contains metadata and consists of two parts- XML Declaration and Document Type Declaration. XML Declaration specifies the version of XML being used. Document Type Declaration defines entities or attributes values and check grammar and vocabulary of markup.
- Root Element also Document Element. It must contain all the other elements and content in the document. An XML element has a start Tag and end Tag.

2.6.XML Document Life Cycle

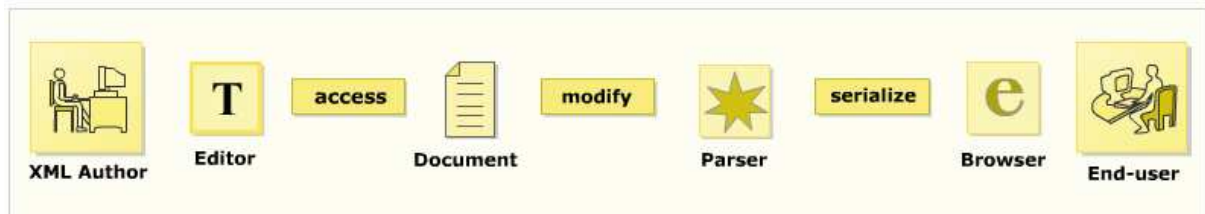


Figure 12.2 XML Document Life Cycle

- An XML editor refers to the DTD and creates an XML document. After the document is created the document is scanned for elements and attributes in it. This stage is known as **Scanning**. The XML parser builds a complete data structure after parsing.
- The data is then extracted from elements and attributes of the document. This stage is known as **Access**.
- It is then converted into the application program. The document structure can also be modified during the process by inserting or deleting elements or by changing textual content of element or attribute. This stage is known as **Modification**.
- The data is then serialized to a textual form and is passed to a browser or any other application that can display it. This stage is known as **Serialization**.

2.7.Well-formed XML document

Well-formedness refers to the standards that are to be followed by the XML documents. Well-formedness makes XML processors and browsers read XML documents. A document is well formed, it fulfills the following rules.

- Minimum of one element is required.
- XML tags are case sensitive.
- Every start tag should end with end tag.
- XML tags should be nested properly
- XML tags should be valid.
- Length of markup names.
- XML attributes should be valid
- XML documents should be verified.

2.8. Classification of character data

An XML document is divided into markup and character data

- Character data describes the document's actual content with the white space. The text in character data is not processed by the parser and thus not treated as a regular text. The character data can be classified into CDATA and PCDATA
 - PCDATA the data that is parsed by the parser is called as parsed character data (PCDATA). The PCDATA specifies that the element has parsed character data. It is used in the element declaration. Escape character like "<" when used in the XML document will make the parser interpret it as a new element.
 - CDATA The text inside a CDATA section is not parsed by XML parser. A text is considered in a CDATA section if it contains '<' or '<&>' characters. The CDATA part begins with a "<![CDATA[" and ends with "]]>". The CDATA sections cannot be nested. It also does not accept line breaks or spaces inside the "]]>" string. Comments are also not recognized.
-
-

3.XHTML

3.1.What is XHTML

- The **Extensible Hypertext Markup Language**, or **XHTML**, is a [markup language](#) that has the same depth of expression as [HTML](#), but also conforms to [XML](#) syntax.
- While HTML prior to [HTML 5](#) was defined as an application of [Standard Generalized Markup Language](#) (SGML), a very flexible markup language, XHTML is an application of [XML](#), a more restrictive subset of SGML. Because they need to be [well-formed](#), true XHTML documents allow for automated processing to be performed using standard XML tools—unlike HTML, which requires a relatively complex, lenient, and generally custom [parser](#).
- XHTML can be thought of as the intersection of HTML and XML in many respects, since it is a reformulation of HTML in XML. XHTML 1.0 became a [World Wide Web Consortium](#) (W3C) [Recommendation](#) on January 26, 2000. XHTML 1.1 became a W3C Recommendation on May 31, 2001.

3.2. Why the need for XHTML?

- Document developers and user agent designers are constantly discovering new ways to express their ideas through new markup. In XML, it is relatively easy to introduce new elements or additional element attributes. The XHTML family is designed to accommodate these extensions through XHTML modules and techniques for developing new XHTML-conforming modules (described in the XHTML Modularization specification). These modules will permit the combination of existing and new feature sets when developing content and when designing new user agents.
- Alternate ways of accessing the Internet are constantly being introduced. The XHTML family is designed with general user agent interoperability in mind. Through a new user agent and document profiling mechanism, servers, proxies, and user agents will be able to perform best effort content transformation. Ultimately, it will be possible to develop XHTML-conforming content that is usable by any XHTML-conforming user agent.

3.3. Valid XHTML documents

An XHTML document that conforms to an XHTML specification is said to be *valid*. Validity assures consistency in document code, which in turn eases processing, but does not necessarily ensure consistent rendering by browsers. A

document can be checked for validity with the [W3C Markup Validation Service](#). In practice, many web development programs provide code validation based on the [W3C](#) standards.

3.3.1. Root element

- The root element of an XHTML document must be `html`, and must contain an `xmlns` attribute to associate it with the XHTML [namespace](#). The namespace URI for XHTML is `http://www.w3.org/1999/xhtml`. For XHTML 1.1 and later there should also ideally be a `version` attribute to clearly identify the version of XHTML being used. The example tag below additionally features an `xml:lang` attribute to identify the document with a [natural language](#).

```
<html xmlns="http://www.w3.org/1999/xhtml" version="XHTML 1.2"
xml:lang="en">
```

- For XHTML 1.1 and 2.0 an optional `schemaLocation` attribute can be added, to associate the namespace with an [XML Schema](#). The example below is for XHTML 2.0.

```
<html xmlns="http://www.w3.org/2002/06/xhtml2/" version="XHTML 2.0"
xml:lang="en" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/1999/xhtml
http://www.w3.org/MarkUp/SCHEMA/xhtml2.xsd">
```

- This example also demonstrates the use of multiple namespaces within a document. The `xmlns:xsi` declaration indicates that any elements or attributes prefixed with `xsi:` fall within the XML Schema instance namespace rather than the XHTML namespace. This mechanism of prefixes allows elements and attributes from different XML vocabularies to be mixed together in the same document, without the potential for naming clashes.

3.3.2. DOCTYPEs

- In order to validate an XHTML document, a [Document Type Declaration](#), or DOCTYPE, may be used. A DOCTYPE declares to the browser that [Document Type Definition](#) (DTD) the document conforms to. A Document Type Declaration should be placed before the [root element](#).
 - The [system identifier](#) part of the DOCTYPE, which in these examples is the [URL](#) that begins with `http://`, need only point to a copy of the DTD to use, if the validator cannot locate one based on the [public identifier](#) (the other
-
-

quoted string). It does not need to be the specific URL that is in these examples.

- In fact, authors are encouraged to use local copies of the DTD files when possible. The public identifier, however, must be character-for-character the same as in the examples.

3.3.3. XML declaration

- A [character encoding](#) may be specified at the beginning of an XHTML document in the XML declaration when the document is served using the `application/xhtml+xml` MIME type. (If an XML document lacks encoding specification, an XML parser assumes that the encoding is [UTF-8](#) or [UTF-16](#), unless the encoding has already been determined by a higher protocol.)

```
<?xml version="1.0" encoding="UTF-8" ?>
```

- The declaration may be optionally omitted because it declares as its encoding the default encoding. However, if the document instead makes use of XML 1.1 or another character encoding, a declaration is necessary. [Internet Explorer](#) prior to version 7 enters [quirks mode](#), if it encounters an XML declaration in a document served as `text/html`.

3.3.4. Using XHTML with other namespaces

- The XHTML namespace may be used with other XML namespaces as per [\[XMLNS\]](#), although such documents are not strictly conforming XHTML 1.0 documents as defined above. Work by W3C is addressing ways to specify conformance for documents involving multiple namespaces. For an example, see [\[XHTML+MathML\]](#).
- The following example shows the way in which XHTML 1.0 could be used in conjunction with the MathML Recommendation:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>A Math Example</title>
  </head>
  <body>
    <p>The following is MathML markup:</p>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply> <log/>
        <logbase>
          <cn> 3 </cn>
        </logbase>
      </apply>
    </math>
  </body>
</html>
```

```
<ci> x </ci>
</apply>
</math>
</body>
</html>
```

- The following example shows the way in which XHTML 1.0 markup could be incorporated into another XML namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- initially, the default namespace is "books" -->
<book xmlns='urn:loc.gov:books'
  xmlns:isbn='urn:ISBN:0-395-36341-6' xml:lang="en" lang="en">
  <title>Cheaper by the Dozen</title>
  <isbn:number>1568491379</isbn:number>
  <notes>
  <!-- make HTML the default namespace for a hypertext commentary -->
  <p xmlns='http://www.w3.org/1999/xhtml'>
    This is also available <a href="http://www.w3.org/">online</a>.
  </p>
  </notes>
</book>
```

3.4. Differences with HTML.

3.4.1. Documents must be well-formed

- [Well-formedness](#) is a new concept introduced by [\[XML\]](#). Essentially this means that all elements must either have closing tags or be written in a special form (as described below), and that all the elements must nest properly.
- Although overlapping is illegal in SGML, it is widely tolerated in existing browsers.

CORRECT: nested elements.

```
<p>here is an emphasized <em>paragraph</em>.</p>
```

INCORRECT: overlapping elements

```
<p>here is an emphasized <em>paragraph.</p></em>
```

3.4.2. Element and attribute names must be in lower case

-
-
- XHTML documents must use lower case for all HTML element and attribute names. This difference is necessary because XML is case-sensitive e.g. and are different tags.

3.4.3. For non-empty elements, end tags are required

- In SGML-based HTML 4 certain elements were permitted to omit the end tag; with the elements that followed implying closure. XML does not allow end tags to be omitted. All elements other than those declared in the DTD as EMPTY must have an end tag. Elements that are declared in the DTD as EMPTY can have an end tag *or* can use empty element shorthand.

CORRECT: terminated elements

```
<p>here is a paragraph.</p><p>here is another paragraph.</p>
```

INCORRECT: unterminated elements

```
<p>here is a paragraph.<p>here is another paragraph.
```

3.4.4. Attribute values must always be quoted

- All attribute values must be quoted, even those which appear to be numeric.

CORRECT: quoted attribute values

```
<td rowspan="3">
```

INCORRECT: unquoted attribute values

```
<td rowspan=3>
```

3.4.5. Attribute Minimization

- XML does not support attribute minimization. Attribute-value pairs must be written in full. Attribute names such as compact and checked cannot occur in elements without their value being specified.

CORRECT: unminimized attributes

```
<dl compact="compact">
```

INCORRECT: minimized attributes

```
<dl compact>
```

3.4.6. Empty Elements

- Empty elements must either have an end tag or the start tag must end with `/>`. For instance, `
` or `<hr></hr>`.

CORRECT: terminated empty elements

```
<br/><hr/>
```

INCORRECT: unterminated empty elements

```
<br><hr>
```

3.4.7. White Space handling in attribute values

- Strip leading and trailing white space.
- Map sequences of one or more white space characters (including line breaks) to a single inter-word space.

3.4.8. Script and Style elements

- In XHTML, the script and style elements are declared as having #PCDATA content. As a result, `<` and `&` will be treated as the start of markup, and entities such as `<` and `&` will be recognized as entity references by the XML processor to `<` and `&` respectively. Wrapping the content of the script or style element within a CDATA marked section avoids the expansion of these entities.

```
<script type="text/javascript">  
<![CDATA[  
... unescaped script content ...  
]]>  
</script>
```

- CDATA sections are recognized by the XML processor and appear as nodes in the Document Object Model.

An alternative is to use external script and style documents.

Chapter 12

1 Web 2.0

1.1 definition



"Web 2.0" refers to a second generation of web development and design, that facilitates communication, secure information sharing, interoperability, and collaboration on the World Wide Web. Web 2.0 concepts have led to the development and evolution of web-based communities, hosted services, and applications such as social-networking sites, video-sharing sites, wikis, blogs, mashup and folksonomies.

"Web 2.0" is a revolution in computer industry. It occurred when people no longer use computers but Internet as a foundation and make effort to succeed on that new foundation. The main rule is: to build up applications that can take advantages of "network effects" to make better value and thus, there are more users. (In another word, it means to take advantages of "collective wisdom")

1.2 Characteristics

First, it is necessary to confirm that: web 2.0 is affirmed by many people and media as a performance tendency/method of websites rather than technology and programming techniques. However, web 2.0 also has page layout, display, technology that are different from Web 1.0. Let us try comparing some points of web 1.0 with web 2.0.

For Web 1.0

With Web 1.0, a company of groups of individuals create a web page, and its development focus on the parents who birth father is this, they will update the information, decision sites have something on that site and how development depends on the efforts of the few individuals involved in the operation and management business from the site it

Web 1.0 users in the very passive, meaning that they can visit, can send little information on to contact, comment on the compatibility between the user and the user is most likely be omitted or very limited

Web 1.0 information resources are concentrated on one and limited. Website seems to be only used for organization units specified available as a company, write a report, state agencies, governments, ...

For Web 2.0

Web 2.0 is a trend of industry World Wide Web (WWW), web design, in the second community-based web services and hosted (stored & server on the network) as a social network, blog, wiki, the mark/ open web directory purpose is to create the easy way to create and share information, collaborate with each other among the users. This is how many people see as a new era of WWW. The term Web 2.0 applications, note that the first workshop on Web 2.0 by O'Reilly Media Web 2.0 in 2004. Although the term is mentioned as a new version of the WWW, but it does not imply on the updated technical particular, it focuses on how the software development and web user interaction.



We can distinguish by the following signals:

web 1.0 – collective

web 2.0 – dispersed in many places

web 1.0 – for individuals

web 2.0 – for society, collective wisdom

web 1.0 – provide content

web 2.0 – provide services and APIs

web 1.0 – readable

web 2.0 – writable

web 1.0 – communication between systems

web 2.0 – synchronization between systems

web 1.0 – the system includes structure, the content generated is pre-calculated

web 2.0 – auto-generate and auto-suggest

web1.0 – static

web2.0 – connected, portable

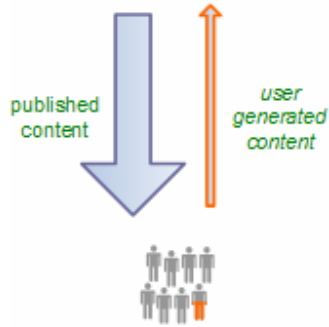
web1.0 – solid, inflexible

web2.0 – loose and flexible relation

Web 1.0

"the mostly read-only Web"

250,000 sites



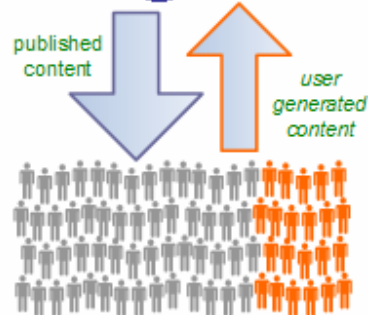
45 million global users

1996

Web 2.0

"the wildly read-write Web"

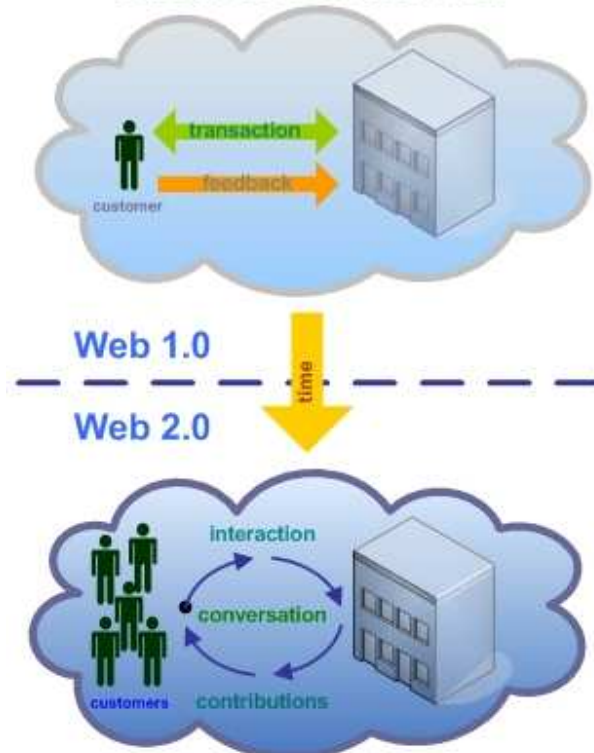
80,000,000 sites




1 billion+ global users

2006

Differences in the Way Organizations Have Interacted with Customers on the Web



Source: <http://web2.wsj2.com> 

Example describing the difference between 1.0 & 2.0

Web 1.0 --> Web 2.0

DoubleClick --> Google AdSense

Ofoto --> Flickr

Akamai --> BitTorrent

mp3.com --> Napster

Britannica Online --> Wikipedia

personal websites --> blogging

evite --> upcoming.org and EVDB

domain name speculation --> search engine optimization

page views --> cost per click

screen scraping --> web services

publishing --> participation

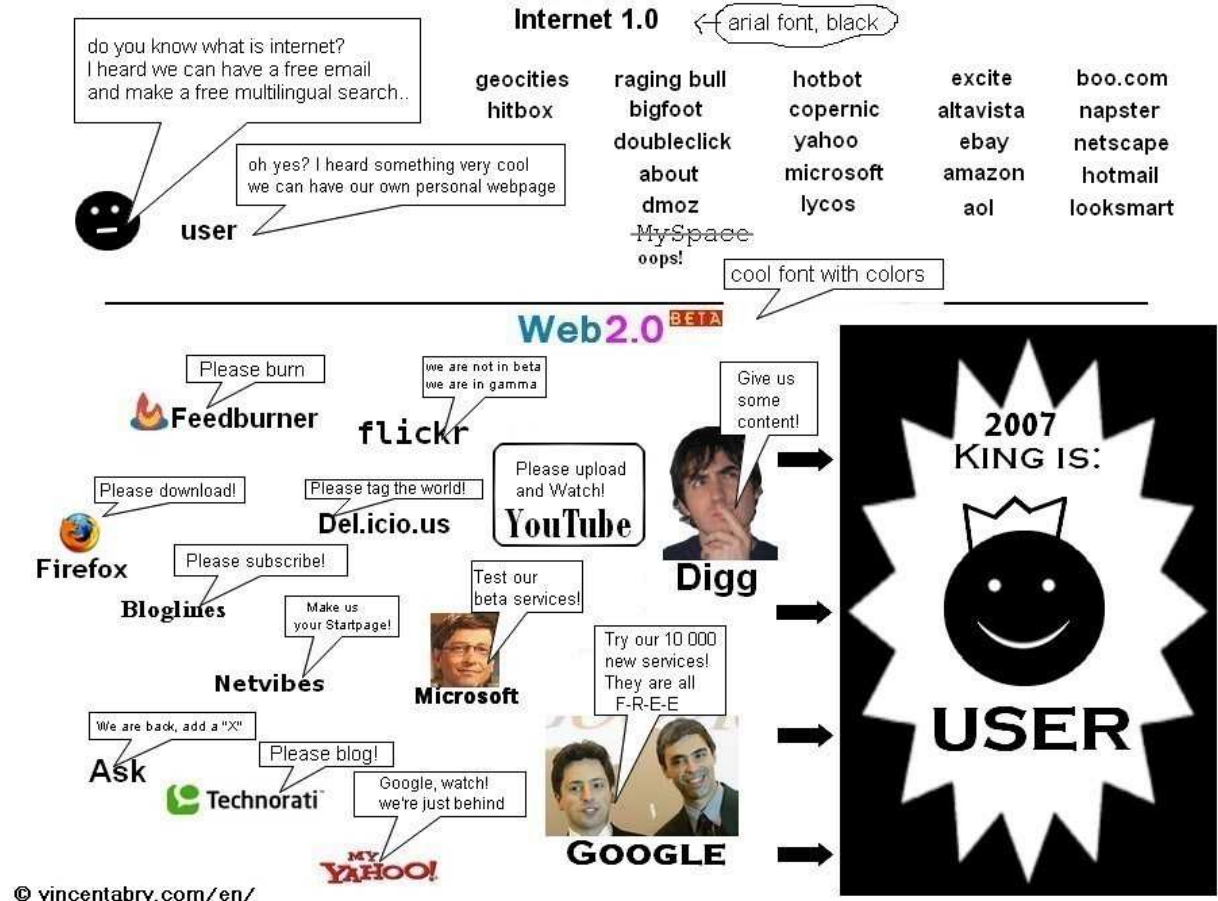
content management systems --> wikis

directories (taxonomy) --> tagging ("folksonomy")

stickiness --> syndication

Web 1.0: for an organization to pre-specified. Example Electronic Press Vietnamnet, Dantri.

the Web 2.0 : to anyone, the number of participants is unlimited, is social. Value of the Web site was created by the emerging contributions of participants in the network society. Example: Y! 360, Cyvee, Cyworld, ...



Border between Web 2.0 and Web 1.0 in some cases is quite fragile. For example, Bangkok is the web 1.0, report new (old version) is a Web 1.0, new version is assigned to Web 2.0

1.3 Examples

-Google Maps

Company search services giant makes up a large part in the "cake pieces" when the Web 2.0 software launch map activities relatively excellent. Based on AJAX, Google Maps support people see more models of the city or a region is on the world. They can zoom in, zoom out to observe each child in the street photos taken from satellites . Programming interface applications (Application Program Interface - API) Google Maps also help experts develop promote maximum creativity when building software map.



All that users need to enter the address position and they want to keyword search, Google will do the end .For example, if they are in Kansas and want to have an area to sell Thai food, Google Maps immediately provide a list of shops to the left of the page, while a Google Map maps will appear at the right, mark the restaurants near address their best. Google Local is a "companion" great for all of their city.

-Flickr

Flickr is an online photo management and sharing application. Its primary goals are to help people make photos available to those who matter to them, and to enable new ways of organizing pictures. You can join Flickr for free and begin sharing images immediately. Pro accounts are available for those who want to add and display high volumes of photos.



Flickr

2.DOM

2.1 Introduction

The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term "document" is used in the broad sense - increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model, with a few exceptions - in particular, the DOM interfaces for the XML internal and external subsets have not yet been specified.

As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide

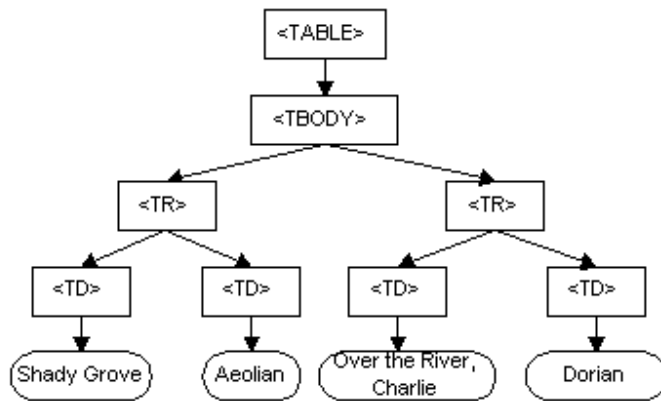
variety of environments and applications. The DOM is designed to be used with any programming language. In order to provide a precise, language-independent specification of the DOM interfaces, we have chosen to define the specifications in OMG IDL, as defined in the [CORBA 2.2 specification](#). In addition to the OMG IDL specification, we provide language bindings for Java and ECMAScript (an industry-standard scripting language based on JavaScript and JScript). ***Note:** OMG IDL is used only as a language-independent and implementation-neutral way to specify interfaces. Various other IDLs could have been used. In general, IDLs are designed for specific computing environments. The Document Object Model can be implemented in any computing environment, and does not require the object binding runtimes generally associated with such IDLs.*

What the Document Object Model is

The DOM is a programming API for documents. It closely resembles the structure of the documents it models. For instance, consider this table, taken from an HTML document:

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

The DOM represents this table like this:



DOM representation of the example table

In the DOM, documents have a logical structure which is very much like a tree; to be more precise, it is like a "forest" or "grove", which can contain more than one tree. However, the DOM does not specify that documents must be *implemented* as a tree or a grove, nor does it specify how the relationships among objects be implemented. The DOM is a logical model that may be implemented in any convenient manner. In this specification, we use the term *structure model* to describe the tree-like representation of a document; we specifically avoid terms like "tree" or "grove" in order to avoid implying a particular implementation. One important property of DOM structure models is *structural isomorphism*: if any two Document Object Model implementations are used to create a representation of the same document, they will create the same structure model, with precisely the same objects and relationships.

The name "Document Object Model" was chosen because it is an "object model" in the traditional object oriented design sense: documents are modeled using objects, and the model encompasses not only the structure of a document, but also the behavior of a document and the objects of which it is composed. In other words, the nodes in the above diagram do not represent a data structure, they represent objects, which have functions and identity. As an object model, the DOM identifies:

- the interfaces and objects used to represent and manipulate a document
- the semantics of these interfaces and objects - including both behavior and attributes
- the relationships and collaborations among these interfaces and objects

The structure of SGML documents has traditionally been represented by an abstract data model, not by an object model. In an abstract data model, the model is centered around the data. In object oriented programming languages, the data

itself is encapsulated in objects that hide the data, protecting it from direct external manipulation. The functions associated with these objects determine how the objects may be manipulated, and they are part of the object model.

The Document Object Model currently consists of two parts, DOM Core and DOM HTML. The DOM Core represents the functionality used for XML documents, and also serves as the basis for DOM HTML. A compliant implementation of the DOM must implement all of the fundamental interfaces in the Core chapter with the semantics as defined. Further, it must implement at least one of the HTML DOM and the extended (XML) interfaces with the semantics as defined.

What the Document Object Model is not

This section is designed to give a more precise understanding of the DOM by distinguishing it from other systems that may seem to be like it.

- Although the Document Object Model was strongly influenced by "Dynamic HTML", in Level 1, it does not implement all of "Dynamic HTML". In particular, events have not yet been defined. Level 1 is designed to lay a firm foundation for this kind of functionality by providing a robust, flexible model of the document itself.
 - The Document Object Model is not a binary specification. DOM programs written in the same language will be source code compatible across platforms, but the DOM does not define any form of binary interoperability.
 - The Document Object Model is not a way of persisting objects to XML or HTML. Instead of specifying how objects may be represented in XML, the DOM specifies how XML and HTML documents are represented as objects, so that they may be used in object oriented programs.
 - The Document Object Model is not a set of data structures, it is an object model that specifies interfaces. Although this document contains diagrams showing parent/child relationships, these are logical relationships defined by the programming interfaces, not representations of any particular internal data structures.
 - The Document Object Model does not define "the true inner semantics" of XML or HTML. The semantics of those languages are defined by W3C Recommendations for these languages. The DOM is a programming model designed to respect these semantics. The DOM does not have any ramifications for the way you write XML and HTML documents; any document that can be written in these languages can be represented in the DOM.
-
-

-
-
- The Document Object Model, despite its name, is not a competitor to the Component Object Model (COM). COM, like CORBA, is a language independent way to specify interfaces and objects; the DOM is a set of interfaces and objects designed for managing HTML and XML documents. The DOM may be implemented using language-independent systems like COM or CORBA; it may also be implemented using language-specific bindings like the Java or ECMAScript bindings specified in this document.

Where the Document Object Model came from

The DOM originated as a specification to allow JavaScript scripts and Java programs to be portable among Web browsers. "Dynamic HTML" was the immediate ancestor of the Document Object Model, and it was originally thought of largely in terms of browsers. However, when the DOM Working Group was formed at W3C, it was also joined by vendors in other domains, including HTML or XML editors and document repositories. Several of these vendors had worked with SGML before XML was developed; as a result, the DOM has been influenced by SGML Groves and the HyTime standard. Some of these vendors had also developed their own object models for documents in order to provide an API for SGML/XML editors or document repositories, and these object models have also influenced the DOM.

DOM Interfaces and DOM Implementations

The DOM specifies interfaces which may be used to manage XML or HTML documents. It is important to realize that these interfaces are an abstraction - much like "abstract base classes" in C++, they are a means of specifying a way to access and manipulate an application's internal representation of a document. Interfaces do not imply a particular concrete implementation. Each DOM application is free to maintain documents in any convenient representation, as long as the interfaces shown in this specification are supported. Some DOM implementations will be existing programs that use the DOM interfaces to access software written long before the DOM specification existed. Therefore, the DOM is designed to avoid implementation dependencies; in particular,

1. Attributes defined in the IDL do not imply concrete objects which must have specific data members - in the language bindings, they are translated to a pair of get()/set() functions, not to a data member. (Read-only functions have only a get() function in the language bindings).
 2. DOM applications may provide additional interfaces and objects not found in this specification and still be considered DOM compliant.
 3. Because we specify interfaces and not the actual objects that are to be created, the DOM can not know what constructors to call for an implementation. In general, DOM users call the createXXX() methods on the Document class to create document structures, and DOM
-
-

implementations create their own internal representations of these structures in their implementations of the createXXX() functions.

HTML DOM Window Object

Window Object

The Window object is the top level object in the JavaScript hierarchy.

The Window object represents a browser window.

A Window object is created automatically with every instance of a <body> or <frameset> tag.

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Window Object Collections

Collection	Description	IE	F	O
frames[]	Returns all named frames in the window	4	1	9

Window Object Properties

Property	Description	IE	F	O
closed	Returns whether or not a window has been closed	4	1	9
defaultStatus	Sets or returns the default text in the statusbar of the window	4	No	9
document	See Document object	4	1	9
history	See History object	4	1	9
length	Sets or returns the number of frames in the window	4	1	9
location	See Location object	4	1	9
name	Sets or returns the name of the window	4	1	9
opener	Returns a reference to the window that created the window	4	1	9
outerHeight	Sets or returns the outer height of a window	No	1	No
outerWidth	Sets or returns the outer width of a window	No	1	No
pageXOffset	Sets or returns the X position of the current page in relation to the upper left corner of a window's display area	No	No	No
pageYOffset	Sets or returns the Y position of the current page in relation to the upper left corner of a window's display area	No	No	No
parent	Returns the parent window	4	1	9
personalbar	Sets whether or not the browser's personal bar (or directories bar) should be visible			
scrollbars	Sets whether or not the scrollbars should be visible			

self	Returns a reference to the current window	4	1	9
status	Sets the text in the statusbar of a window	4	No	9
statusbar	Sets whether or not the browser's statusbar should be visible			
toolbar	Sets whether or not the browser's tool bar is visible or not (can only be set before the window is opened and you must have UniversalBrowserWrite privilege)			
top	Returns the topmost ancestor window	4	1	9

Window Object Methods

Method	Description	IE	F	O
alert()	Displays an alert box with a message and an OK button	4	1	9
blur()	Removes focus from the current window	4	1	9
clearInterval()	Cancels a timeout set with setInterval()	4	1	9
clearTimeout()	Cancels a timeout set with setTimeout()	4	1	9
close()	Closes the current window	4	1	9
confirm()	Displays a dialog box with a message and an OK and a Cancel button	4	1	9
createPopup()	Creates a pop-up window	4	No	No
focus()	Sets focus to the current window	4	1	9
moveBy()	Moves a window relative to its current position	4	1	9
moveTo()	Moves a window to the specified position	4	1	9
open()	Opens a new browser window	4	1	9
print()	Prints the contents of the current window	5	1	9
prompt()	Displays a dialog box that prompts the user for input	4	1	9
resizeBy()	Resizes a window by the specified pixels	4	1	9
resizeTo()	Resizes a window to the specified width and height	4	1.5	9
scrollBy()	Scrolls the content by the specified number of pixels	4	1	9
scrollTo()	Scrolls the content to the specified coordinates	4	1	9
setInterval()	Evaluates an expression at specified intervals	4	1	9
setTimeout()	Evaluates an expression after a specified number of milliseconds	4	1	9

HTML DOM Button Object

Button Object

The Button object represents a push button.

For each instance of a <button> tag in an HTML document, a Button object is created.

IE: Internet Explorer, F: Firefox, O: Opera, W3C: World Wide Web Consortium (Internet Standard).

Button Object Properties

Property	Description	IE	F	O	W3C
accessKey	Sets or returns the keyboard key to access a button	6	1	9	Yes
disabled	Sets or returns whether a button should be disabled	6	1	9	Yes
form	Returns a reference to the form that contains the button	6	1	9	Yes
id	Sets or returns the id of a button	6	1	9	Yes
name	Sets or returns the name of a button	6	1	9	Yes
tabIndex	Sets or returns the tab order for a button	6	1	9	Yes
type	Returns the type of form element a button is	6	1	9	Yes
value	Sets or returns the text that is displayed on a button	6	1	9	Yes

Frame Object

The Frame object represents an HTML frame.

For each instance of a <frame> tag in an HTML document, a Frame object is created.

IE: Internet Explorer, F: Firefox, O: Opera, W3C: World Wide Web Consortium (Internet Standard).

Frame Object Properties

Property	Description	IE	F	O	W3C
contentDocument	Returns the frame's document as an HTML object	No	1	9	Yes
frameBorder	Sets or returns whether or not to display borders around a frame	5	1	9	Yes
id	Sets or returns the id of a frame	4	1	9	Yes
longDesc	Sets or returns a URL to a document containing a description of the frame contents	6	1	9	Yes
marginHeight	Sets or returns the top and bottom margins of a frame	5	1	9	Yes
marginWidth	Sets or returns the left and right margins of a frame	5	1	9	Yes
name	Sets or returns the name of a frame	5	1	9	Yes
noResize	Sets or returns whether or not a frame can be resized	5	1	9	Yes
scrolling	Sets or returns whether or not a frame should have scrollbars	No	1	No	Yes

src	Sets or returns the URL of the document that should be loaded into a frame	5	1	9	Yes
-----	--	---	---	---	-----

A simple DOM example

The following Java program uses DOM to read the recipe collection and cut it down to the first recipe:

```
import java.io.*;
import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.*;

public class FirstRecipeDOM {

    public static void main(String[] args) {
        try {
            DOMParser p = new DOMParser();
            p.parse(args[0]);
            Document doc = p.getDocument();
            Node n = doc.getDocumentElement().getFirstChild();
            while (n!=null && !n.getNodeName().equals("recipe"))
                n = n.getNextSibling();
            PrintStream out = System.out;
            out.println("<?xml version=\"1.0\"?>");
            out.println("<collection>");
            if (n!=null)
                print(n, out);
            out.println("</collection>");
        } catch (Exception e) {e.printStackTrace();}
    }

    static void print(Node node, PrintStream out) {
        int type = node.getNodeType();
        switch (type) {
            case Node.ELEMENT_NODE:
                out.print("<" + node.getNodeName());
                NamedNodeMap attrs = node.getAttributes();
                int len = attrs.getLength();
                for (int i=0; i<len; i++) {
```

```

    Attr attr = (Attr)attrs.item(i);
    out.print(" " + attr.getNodeName() + "=\"" +
              escapeXML(attr.getNodeValue()) + "\"");
    }
    out.print('>');
    NodeList children = node.getChildNodes();
    len = children.getLength();
    for (int i=0; i<len; i++)
        print(children.item(i), out);
    out.print("</" + node.getNodeName() + ">");
    break;
case Node.ENTITY_REFERENCE_NODE:
    out.print("&" + node.getNodeName() + ";");
    break;
case Node.CDATA_SECTION_NODE:
    out.print("<![CDATA[" + node.getNodeValue() + "]]>");
    break;
case Node.TEXT_NODE:
    out.print(escapeXML(node.getNodeValue()));
    break;
case Node.PROCESSING_INSTRUCTION_NODE:
    out.print("<?" + node.getNodeName());
    String data = node.getNodeValue();
    if (data!=null && data.length(>0)
        out.print(" " + data);
    out.println(">");
    break;
    }
}

static String escapeXML(String s) {
    StringBuffer str = new StringBuffer();
    int len = (s != null) ? s.length() : 0;
    for (int i=0; i<len; i++) {
        char ch = s.charAt(i);
        switch (ch) {
            case '<': str.append("&lt;"); break;
            case '>': str.append("&gt;"); break;
            case '&': str.append("&amp;"); break;
            case '\"': str.append("&quot;"); break;
            case '\\': str.append("&apos;"); break;
            default: str.append(ch);
        }
    }
}

```

```
return str.toString();
}
}
```

Change text, URL, and target attribute of a link

```
<html>
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('myAnchor').innerHTML="Visit W3Schools";
document.getElementById('myAnchor').href="http://www.w3schools.com";
document.getElementById('myAnchor').target="_blank";
}
</script>
</head>

<body>
<a id="myAnchor" href="http://www.microsoft.com">Visit Microsoft</a>
<input type="button" onclick="changeLink()" value="Change link">
<p>In this example we change the text and the URL of a hyperlink. We also
change the target attribute.
The target attribute is by default set to "_self", which means that the link will open
in the same window.
By setting the target attribute to "_blank", the link will open in a new
window.</p>
</body>

</html>
```

3 AJAX

3.1 What is AJAX?

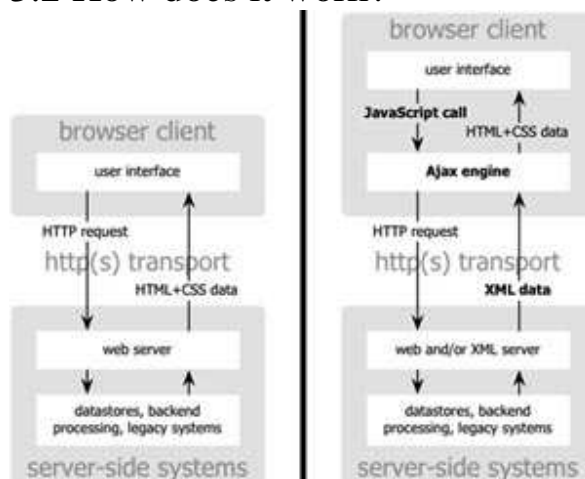
AJAX, short for Asynchronous JavaScript and from XML (JavaScript and XML is not sync), is a set of tools to allow speed up web applications by cutting a small data and display only what you need, instead of loading the load the entire site. AJAX is not a single technology but a combination of a group of technologies together. The HTML and CSS as display data, DOM model presented the

information, XMLHttpRequest object to exchange data is not synchronized with the web server, and XML format is primarily for data transmission.

Most stories about the origin of AJAX is beginning to develop the Microsoft Remote Scripting technology in 1998. However, the method is not of the content on a web page appear in the IFRAME's Internet Explorer 3 (1996) and the layer of Netscape 4.0 in 1997. When introduced Internet Explorer 4.0, Microsoft has used the object model document DOM differences. To 2000, Netscape completely lose the browser market into the hands of manufacturers software Bill Gates and the layer is no longer the specialist web development attention.

Must be a few years later, the new AJAX engaging the interest of the technology and tools to become innovative user interfaces for web applications. This term appears only 1 year ago (2 / 2005) in the famous article by Jesse James Garrett on adaptive Path. Since then, AJAX has become the center of all stories related to the Web 2.0.

3.2 How does it work?



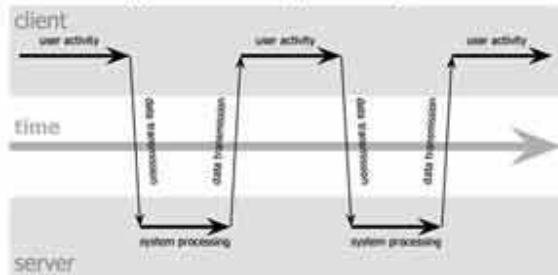
Web Applications traditional (left) and AJAX applications. (Adaptive Path)

people imagine a computer application will then be stored and run completely on the web instead of chain located in hard drive. Whatever the scenario that has not happen due to web applications is limited by the principle that all actions must be done through HTTP (HyperText Transfer Protocol - Protocol transmitted through hyperlink) The activities of users on the site will generate a request to the HTTP server. Servers perform a business process such as retrieve the data, calculation and check the valid information, modify memory, then send a complete HTML page to the client. Of techniques, this approach seems reasonable, but quite inconvenient and time consuming, because the server is implementing its role, the user will do? Of course waiting.

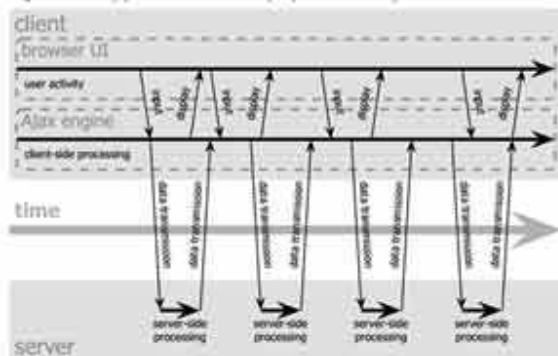
To overcome the limitations, experts developed referral forms intermediaries- the process AJAX - between client and server. This is like raising a middle class for the application to reduce the "travel" of information and reduce response time. Instead reload (refresh) a whole page, it only load the information

changes, and keep other parts. Instead reload (refresh) a whole page, it only load the information changes, and keep other parts. Therefore, when a browser supports AJAX, the user will never see a white window (blank) and the icon of sand - signs that the server is performed tasks. For example, in a photo website, with traditional applications, the entire page containing the photo will be open from the beginning if a change is on page. When applied AJAX, DHTML replace only the title and just edit, so make the transaction smooth, quick.

classic web application model (synchronous)



Ajax web application model (asynchronous)



Interaction of web applications in traditional (on) and walk in AJAX applications. (Adaptive Path)

"Any action by the user will send commands to JavaScript AJAX processor, instead of creating an HTTP request (HTTP request) and to query the server," Jesse James Garrett stated in the article first defines of this term. "If you need anything from the server, such as download additional code or get the new data, will transmit AJAX request to a server is not synchronized, usually using XML, but not to interrupt the interaction of users with Web applications " .

3.3 AJAX - applications popular

Google Suggest displays the term suggests almost immediately when the user has not finished typing your keywords. And with Google Maps, people can track changes, and serial, drag-drop on the map as desktop environment. Google Suggest and Google Maps are two prominent examples of methods applied the new web. the search services leading world have invested heavily on the development of AJAX. Almost every program they introduced last year, from Orkut, Gmail version to test Google Groups, are all AJAX applications.

Many other companies also got connected this trend sharing site like Flickr photos (Yahoo does belong) or search Amazon's A9.com. Yahoo plans a few weeks I will launch the Yahoo Mail Beta 1 using the AJAX world (in limited testing). Email will be equipped with more features as belonging to Web 2.0 RSS, see the previous message (preview) ... Microsoft is also implementing the program Windows Live Mail and Windows Live Messenger support AJAX.

The project found that the AJAX is not a technology too far away that are present in the real world, since the model is very simple like Google Suggest to sophisticated and complex as Google Maps.

Biggest challenges when creating AJAX applications is not in phases by the technical part of it has been long, stable operation and is known. Problem here is only the "experts design should forget the limits of the web, start thinking wider, more in-depth about the capabilities of technology and creativity it's own way each person, Alexei White, Director of production company's eBusiness (U.S.), comment. "Ajax will gradually lose influence by Microsoft in the market. While not completely at times, it will feature the alternative for most products as part of Office."

3.4 THE weakness of AJAX

AJAX can contribute to creating a new generation for web applications (or as colr.org backpackit.com). However, it is also a technology "dangerous" when caused many problems on the user interface. For example, the "Back" (return to the previous page) are valued in the standard website. Unfortunate, this feature does not work with Javascript gearing and people can not find the content before the press Back. Therefore, only a small profile of the data on the changes has been difficult and can be restored. This is one of the main reasons people do not make many applications support Javascript.

Besides, people can not save the web address to the Favorite (Bookmark) to review later. By class applied to mediate the transaction, AJAX applications do not have a fixed address for each content. Mistakes as AJAX for easy "points lost" in the human eye does.

The browser supports AJAX is Microsoft Internet Explorer 5.0 and above; Gecko-based browser such as Mozilla, Firefox, SeaMonkey, Epiphany, Galeon and Netscape 7.1; browser contains KHTML API 3.2 and above as Konqueror, Apple Safari ...

- CSS - file type to the floor (Cascading Style Sheets) - is used to describe how the presentation of documents written in HTML, XHTML, XML, SVG, XUL ... The specifications of the CSS organization World Wide Web Consortium (W3C) management.

 - DOM - the object model document (Document Object Model) - is an interface application programming (API). Usually form a DOM tree structure data and is used to access the HTML documents and XML. DOM
-
-

model activities independent of the operating system and based on the technical programming targeted to describe documents.

- DHTML, or dynamic HTML, a site created by a combination of components: markup language static HTML, the language the client commands (eg JavaScript) and language format CSS and DOM. Due to the rich possibilities, DHTML is used as a tool to build simple games on the browser.

4 RIA

4.1 Concept of RIA

- **Rich Internet applications (RIAs)** are web applications that have some of the characteristics of desktop applications, typically delivered by way of a proprietary web browser plug-ins or independently via sandboxes or virtual machines^[1]. Examples of RIA frameworks include Curl (programming language), Adobe Flash/Adobe Flex/AIR, Java/JavaFX, uniPaaS and Microsoft Silverlight.

The term was introduced in the 1990s by vendors like Macromedia who were addressing limitations at the time in the "richness of the application interfaces, media and content, and the overall sophistication of the solutions" by introducing proprietary extensions.

As web standards (such as Ajax and HTML 5) have developed and web browsers' compliance has improved there is less need for such extensions, and Javascript compilers with their associated desktop-like widget sets reduce the need for browser extensions even further. HTML 5 delivers a full-fledged application platform; "a level playing field where video, sound, images, animations, and full interactivity with your computer are all standardized".

It is now possible to build RIA-like Web applications that run in all modern browsers without the need of special run-times or plug-ins. This means that if one could run a modern Ajax-based Web application *outside* of a web browser (e.g. using Mozilla Prism or Fluid) it would essentially be an RIA, though there is some contention as to whether this is actually the case.

4.2 Examples of RIA frameworks

- **Adobe Flex** is a software development kit released by Adobe Systems for the development and deployment of cross-platform rich Internet applications based on
-
-

the Adobe Flash platform. Flex applications can be written using Adobe Flex Builder or by using the freely available Flex compiler from Adobe.

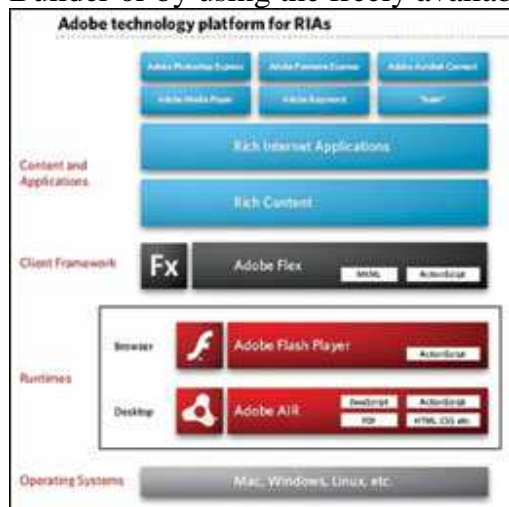


Figure 1. Platform technology by Adobe RIA

Until now, Adobe Systems is the "hand game" the biggest and most mature in the field of RIA. Macromedia bought over 3 years (in 4 / 2005), Adobe is the inheritance to go from production to Shockwave Flash, Flex and AIR (original name is Apollo)

Flash is a technology show is famous for the ability to handle vector graphics and video, allowing content multimedia interact with users through actionscript. Flex-based Flash using language report based on XML (MXML) associated with actionscript, provides isolation between the user interface and logistics applications clearly than Flash. Flex applications are compiled into the file. Swf and run the Flash player - runs on any system which supports Flash, can run on both mobile devices (with Flash Lite) (Figure 1).

- **Adobe Integrated Runtime (AIR)** is a cross-platform runtime environment for building rich Internet applications using Adobe Flash, Adobe Flex, HTML, or Ajax, that can be deployed as a desktop application.

AIR is intended to be a versatile runtime environment, as it allows existing Flash, Actionscript or HTML and JavaScript code to be used to construct a more traditional desktop-like program. Adobe positions it as a browser-less runtime for rich Internet applications that can be deployed onto the desktop, rather than a fully-fledged application framework. The differences between each deployment paradigm provides both advantages and disadvantages. For example, a rich internet application deployed in a browser does not require installation, while one deployed with AIR requires the application be packaged, digitally signed, and installed to the user's local file system. However, this provides access to local storage and file systems, while browser-deployed applications are more limited in where and how data are accessed and stored. In most cases, rich internet applications store users' data on their own servers, but the ability to consume and work with data on a user's local file system allows for greater flexibility.

Adobe AIR 1.1 was released on June 16, 2008, and provides support for building internationalized applications. Runtime installation dialogs were localized to Brazilian Portuguese, Chinese (Traditional and Simplified), French, German, Italian, Japanese, Korean, Russian and Spanish. In addition, version 1.1 includes support for Microsoft Windows XP Tablet PC Edition and 64-bit editions of Windows Vista Home Premium, Business, Ultimate, or Enterprise.

Applications Flash / Flex can run on your desktop with the library implementation AIR. AIR (Adobe Integrated Runtime) technology is deployed to run applications Flash / Flex and Ajax without the browser, similar to Microsoft's ClickOnce - technology developed applications based on Windows, but the AIR run on Windows and Mac (version running on Linux is in the process of testing).

Java is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun made available most of their Java technologies as free software under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Classpath.

Examples

Hello world

The traditional [Hello world program](#) can be written in Java as:

```
/*
 * Outputs "Hello, world!" and then exits
 */

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

A more comprehensive example

```
// OddEven.java
import javax.swing.JOptionPane;

public class OddEven {
    // "input" is the number that the user gives to the computer
```

```

private int input; // a whole number("int" means integer)

/*
 * This is the constructor method. It gets called when an object of the OddEven
type
 * is being created.
 */
public OddEven() {
//Code not shown
}

// This is the main method. It gets called when this class is run through a Java
interpreter.
public static void main(String[] args) {
/*
 * This line of code creates a new instance of this class called "number" (also
known as an
 * Object) and initializes it by calling the constructor. The next line of code
calls
 * the "showDialog()" method, which brings up a prompt to ask you for a
number
 */
    OddEven number = new OddEven();
    number.showDialog();
}

public void showDialog() {
/*
 * "try" makes sure nothing goes wrong. If something does,
 * the interpreter skips to "catch" to see what it should do.
 */
    try {
/*
 * The code below brings up a JOptionPane, which is a dialog box
 * The String returned by the "showInputDialog()" method is converted
into
 * an integer, making the program treat it as a number instead of a word.
 * After that, this method calls a second method, calculate() that will
 * display either "Even" or "Odd."
 */
        input = new Integer(JOptionPane.showInputDialog("Please Enter A
Number"));
        calculate();
    } catch (NumberFormatException e) {

```

```

    /*
    * Getting in the catch block means that there was a problem with the
format of
    * the number. Probably some letters were typed in instead of a number.
    */
    System.err.println("ERROR: Invalid input. Please type in a numerical
value.");
    }
}

/*
* When this gets called, it sends a message to the interpreter.
* The interpreter usually shows it on the command prompt (For Windows
users)
* or the terminal (For Linux users).(Assuming it's open)
*/
private void calculate() {
    if (input % 2 == 0) {
        System.out.println("Even");
    } else {
        System.out.println("Odd");
    }
}
}
}

```

JavaFX is a software platform for creating and delivering rich Internet applications that can run across wide variety of connected devices. The current release (JavaFX 1.2, June 2009) enables building applications for desktop, browser and mobile phones. TV set-top boxes, gaming consoles, Blu-ray players and other platforms are planned.



Figure 2. Overview of JavaFX.

JavaFX (F3 code name) was built based Java technology, consists of 2 parts: JavaFX Script and Java Mobile. JavaFX Script is the language declared not based on XML to help store the burden to build the user interface of capital previously required a lot code command Swing. JavaFX Script must be compiled

into Java byte code to run on Java virtual machine (JVM). JavaFX Mobile is the operating system for mobile devices are designed to deploy RIA applications on the devices (Figure 2)

Microsoft Silverlight is a programmable web browser plugin that enables features such as animation, vector graphics and audio-video playback which characterize rich Internet applications. Version 2.0, released in October 2008, brought additional interactivity features and support for .NET languages and development tools. Microsoft made the beta of Silverlight 3.0 available on March 18, 2009. The final version is expected to arrive on July 10, 2009

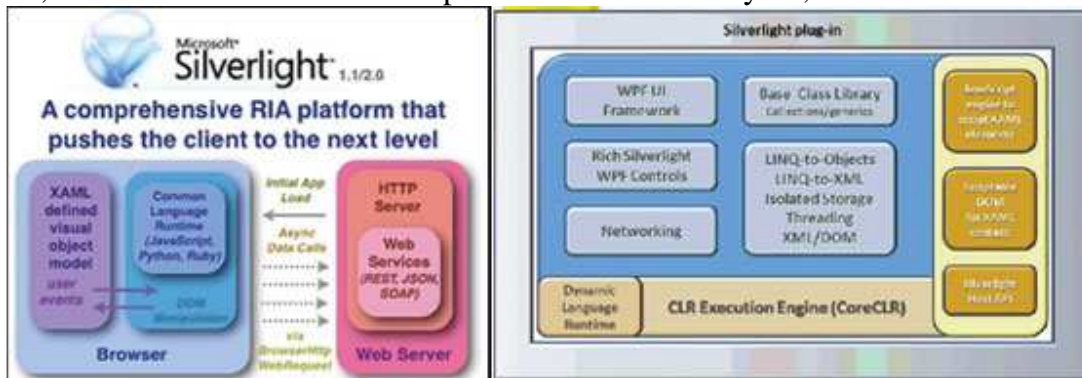


Figure 3

Silverlight (2.0) may be the child of Windows Presentation Foundation (WPF), also using XAML (eXtensible Application Markup Language) - language report based on XML to define the user interface (including vector graphics, effects and interactive data interface). The task processing complex is separated from the defined user interface and can write code with the "managed" (managed code) use the language. NET or JavaScript, and Python / Ruby (Figure 3).

Silverlight install required additional library browser to implement the client. Existing Silverlight applications can run with Internet Explorer and Firefox on Windows, Firefox and Safari on Mac OS X; not support any browsers on Linux (but has the project open source Moonlight enabled applications Silverlight on Linux). Support for mobile devices, Microsoft has Silverlight for Mobile to run on the operating system Nokia S60 and Windows Mobile 6.

Chapter 13

1 Web 3.0



If you think the Web 2.0 is really not perfect, and World Wide Web is still a cloud filled with information of the image, then that is what the Web 3.0 corrects the - delete layer clouds that in certain order.

1.1 The breakthrough of new media channels

This is also the hot topic that most science computers now tracking the day, which is typical Google or Yahoo and the new company set up the depth is Garlik , Metaweb Technologies, Powerset, ZoomInfo, Radar Networks ... with many large corporations in the world such as Citigroup, Eli Lilly, Kodak and Oracle

1.2 definition

Web 3.0 is defined as the creation of high-quality content and services produced by gifted individuals using Web 2.0 technology as an enabling platform.

1.3 Characteristics

Web 3.0 also known to the name of Semantic Web (Web sense) and will be equipped with features high argued. Said another way, sources of information in the directory under the new web will be converted into sources of data that computers can read and estimates. Only in this way, the web will become smarter in the reception of the work that today we still operate with manual, chosen such that the nearest restaurant, booking the best flights or buy a CD with the lowest prices.

The term Semantic Web was born in 2001 by Berners-Lee and two co-author of the book is Scientific American James Hendler and Ora Lassila. For Semantic Web can be operated, sources of information online must be read by devices digital.

Services on the network like Google can do good work to filter information from many sites, but we still want more when viewing search results. Bersyon new Berners-Lee's conduct as a string of serial data together with information in a concentrated form was built before

Inside the structure, they added the system to sense the computer can understand. For example, if you use Google tools to search for someone in the Web 3.0, people that will be described with sufficient information plentiful, from the date of birth, title of work, address home, hobbies to in order must

Users Semantic Web will have the clear connection with the work, relatives, work history and preferences of each other. Suppose want to arrange a banquet following a meeting, you will first browse window address book email and see who can join the conference, then sent invitations to more parties. Then all the guests home with party together talk about time and place via email regale

In Semantic Web, software assistant will know what will be involved in arranging the banquet instead of you to send dozens of emails, it will filter chat object of the conference and on the guest list parties. Even it can also book through your address book to see who lives where and when party organization is the appropriate

Once the guest list has been approved, the software assistant will review the location of the parties, final instructions will be sent to everyone. Even with GPS, it also helps you know how long it is a guests will have to face delays. Ability to connect to sources with individual ways tangible and invisible is what makes the reputation of Web 3.0.

The main differences between Web 1.0, Web 2.0 and Web 3.0.

Web 1.0	Web 2.0	Web 3.0
"the mostly read only web"	"the wildly read-write web"	"the portable personal web"
45 million global users (1996)	1 billion+ global users (2006)	focused on the individual
focused on companies	focused on communities	lifestream
home pages	blogs	consolidating dynamic content
owning content	sharing content	the semantic web
Britannica Online	Wikipedia	widgets, drag & drop mashups
HTML, portals	XML, RSS	user behavior ("me-onomy")
web forms	web applications	iGoogle, NetVibes
directories (taxonomy)	tagging ("folksonomy")	user engagement
Netscape	Google	advertainment
pages views	cost per click	
advertising	word of mouth	

Web 1.0 - That Geocities & Hotmail era was all about read-only content and static HTML

websites. People preferred navigating the web through link directories of Yahoo! and dmoz.

Web 2.0 - This is about user-generated content and the read-write web. People are consuming as well as contributing information through blogs or sites like Flickr, YouTube, Digg, etc. The line dividing a consumer and content publisher is increasingly getting blurred in the Web 2.0 era.

Web 3.0 - This will be about semantic web (or the meaning of data), personalization (e.g. iGoogle), intelligent search and behavioral advertising among other things.

2 Metadata

2.1 Definitions

Metadata (meta data, or sometimes metainformation) is "data about other data", of any sort in any media. An item of metadata may describe an individual datum, or content item, or a collection of data including multiple content items and hierarchical levels, for example a database schema. In data processing, metadata provides information about, or documentation of, other data managed within an

application or environment. This commonly defines the structure or schema of the primary data. The term should be used with caution as all data is about something, and is therefore "metadata" in a sense, and vice versa.

For example, metadata would document data about data elements or attributes, (name, size, data type, etc) and data about records or data structures (length, fields, columns, etc) and data about data (where it is located, how it is associated, ownership, etc.). Metadata may include descriptive information about the context, quality and condition, or characteristics of the data. It may be recorded with high or low granularity.

The term was introduced intuitively, without a formal definition. Because of that, today there are various definitions. The most common one is the literal translation:

Meta is a classical Greek preposition (μετ' ἄλλων εἰρηῶν) and prefix (μεταβασις) conveying the following senses in English, depending upon the case of the associated noun: among; along with; with; by means of; in the midst of; after; behind. In epistemology, the word means "about (its own category)"; thus metadata is "data about the data".

"Data about data are referred to as metadata."

As for most people the difference between data and information is merely a philosophical one of no relevance in practical use, other definitions are:

- Metadata is information about data.
- Metadata is information about information.
- Metadata contains information about that data or other data

There are more sophisticated definitions, such as:

Metadata is structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment, and management of the described entities."

[Metadata is a set of] optional structured descriptions that are publicly available to explicitly assist in locating objects."

Fundamentally, then, metadata is "the data that describe the structure and workings of an organization's use of information, and which describe the systems it uses to manage that information". To do a model of metadata is to do an "Enterprise model" of the information technology industry itself.

2.2 Purpose

Metadata provides context for data.

Metadata is used to facilitate the understanding, characteristics, and management usage of data. The metadata required for effective data management varies with the type of data and context of use. In a library, where the data is the content of the titles stocked, metadata about a title would typically include a description of the content, the author, the publication date and the physical location.

2.3 Examples of metadata

Book

Examples of metadata regarding a book would be the title, author, date of publication, subject, a unique identifier (such an International Standard Book Number), its dimensions, number of pages, and the language of the text.

Photograph

Metadata for a photograph would typically include the date and time at which it was taken and details of the camera settings (such as focal length, aperture, exposure). Many digital cameras record metadata in exchangeable image file format (EXIF).

Audio

Audio recordings may also be labelled with metadata. When audio formats moved from analogue to digital, it became possible to embed this metadata within the digital content itself.

Metadata can be used to name, describe, catalogue and indicate ownership or copyright for a digital audio file, and its presence makes it much easier to locate a specific audio file within a group - through use of a search engine that accesses the metadata. As different digital audio formats were developed, it was agreed that a standardized and specific location would be set aside within the digital files where this information could be stored.

As a result, almost all digital audio formats, including mp3, broadcast wav and AIFF files, have similar standardized locations that can be populated with metadata. This "information about information" has become one of the great advantages of working with digital audio files - since the catalogue and descriptive information that makes up the metadata is built right into the audio file itself, ready for easy access and use.

Web page

The HTML used to mark-up web pages allows for the inclusion of a variety of types of meta data, from simple descriptive text, dates and keywords to highly-granular information such as the Dublin Core and e-GMS standards. Pages can be geotagged with coordinates. Metadata may be included in the page's header or in a separate file. Microformats allow on-page data to be marked up as meta data. The Hypertext Transfer Protocol used to link web pages also includes metadata.

2.4 Use

Metadata has many different applications; this section lists some of the most common.

Metadata is used to speed up and enrich searching for resources. In general, search queries using metadata can save users from performing more complex filter operations manually. It is now common for web browsers (with the notable exception of Mozilla Firefox), P2P applications and media management software to automatically download and locally cache metadata, to improve the speed at which files can be accessed and searched

Metadata may also be associated to files manually. This is often the case with documents which are scanned into a document storage repository such as FileNet or Documentum. Once the documents have been converted into an electronic format a user brings the image up in a viewer application, manually reads the document and keys values into an online application to be stored in a metadata repository. Metadata provide additional information to users of the data it describes. This information may be descriptive or algorithmic

Metadata helps to bridge the semantic gap. By telling a computer how data items are related and how these relations can be evaluated automatically, it becomes possible to process even more complex filter and search operations

Certain metadata is designed to optimize lossy compression.

Some metadata is intended to enable variable content presentation..

Other descriptive metadata can be used to automate workflows

Metadata is becoming an increasingly important part of electronic discovery. [2] Application and file system metadata derived from electronic documents and files can be important evidence

Metadata has become important on the World Wide Web because of the need to find useful information from the mass of information available. Manually-created metadata adds value because it ensures consistency. If a web page about a certain topic contains a word or phrase, then all web pages about that topic should contain that same word or phrase. Metadata also ensures variety, so that if a topic goes by two names each will be used

2.5 Types of metadata

Metadata can be classified by:

- Content. Metadata can either describe the resource itself (for example, name and size of a file) or the content of the resource (for example, "This video shows a boy playing football").
- Mutability. With respect to the whole resource, metadata can be either immutable (for example, the "Title" of a video does not change as the video itself is being played) or mutable (the "Scene description" does change).
- Logical function. There are three layers of logical function: at the bottom the subsymbolic layer that contains the raw data itself, then the symbolic layer with metadata describing the raw data, and on the top the logical layer containing metadata that allows logical reasoning using the symbolic layer

types of metadata are;

1. descriptive metadata.
2. administrative metadata.
3. structural metadata.
4. technical metadata.
5. use metadata

To successfully develop and use metadata, several important issues should be treated with care:

Metadata risks

Microsoft Office files include metadata beyond their printable content, such as the original author's name, the creation date of the document, and the amount of time spent editing it. Unintentional disclosure can be awkward or even, in professional practices requiring confidentiality, raise malpractice concerns. Some of Microsoft Office document's metadata can be seen by clicking File then Properties from the program's menu. Other metadata is not visible except through external analysis of a file, such as is done in forensics

Metadata lifecycle

Even in the early phases of planning and designing it is necessary to keep track of all metadata created. It is not economical to start attaching metadata only after the production process has been completed. For example, if metadata created by a digital camera at recording time is not stored immediately, it may have to be restored afterwards manually with great effort. Therefore, it is necessary for different groups of resource producers to cooperate using compatible methods and standards.

- **Manipulation.** Metadata must adapt if the resource it describes changes. It should be merged when two resources are merged. These operations are seldom performed by today's software; for example, image editing programs usually do not keep track of the Exif metadata created by digital cameras.
- **Destruction.** It can be useful to keep metadata even after the resource it describes has been destroyed, for example in change histories within a text document or to archive file deletions due to digital rights management. None of today's metadata standards consider this phase.

Storage

Metadata can be stored either internally, in the same file as the data, or externally, in a separate file. Metadata that are embedded with content is called embedded metadata. A data repository typically stores the metadata detached from the data

3 RDF

3.1 Introduction to RDF

The Resource Description Framework (RDF) is a W3C standard for describing Web resources, such as the title, author, modification date, content, and copyright information of a Web page.

3.1.1 Definitions

- RDF stands for **R**esource **D**escription **F**ramework
 - RDF is a framework for describing resources on the web
 - RDF is designed to be read and understood by computers
 - RDF is not designed for being displayed to people
 - RDF is written in XML
 - RDF is a part of the W3C's Semantic Web Activity
 - RDF is a W3C Recommendation
-
-

3.1.2 RDF - Examples of Use

- Describing properties for shopping items, such as price and availability
- Describing time schedules for web events
- Describing information about web pages (content, author, created and modified date)
- Describing content and rating for web pictures
- Describing content for search engines
- Describing electronic libraries

RDF is Designed to be Read by Computers

RDF was designed to provide a common way to describe information so it can be read and understood by computer applications.

RDF descriptions are not designed to be displayed on the web.

RDF and "The Semantic Web"

The RDF language is a part of the W3C's Semantic Web Activity. W3C's "Semantic Web Vision" is a future where:

- Web information has exact meaning
- Web information can be understood and processed by computers
- Computers can integrate information from the web

RDF Rules

RDF uses Web identifiers (URIs) to identify resources.

RDF describes resources with properties and property values.

Explanation of Resource, Property, and Property value:

- A **Resource** is anything that can have a URI, such as "http://www.w3schools.com/rdf"
- A **Property** is a Resource that has a name, such as "author" or "homepage"
- A **Property value** is the value of a Property, such as "Jan Egil Refsnes" or "http://www.w3schools.com" (note that a property value can be another resource)

The following RDF document could describe the resource "http://www.w3schools.com/rdf":

```
<?xml version="1.0"?>
```

```
<RDF>  
<Description about="http://www.w3schools.com/rdf">  
  <author>Jan Egil Refsnes</author>  
  <homepage>http://www.w3schools.com</homepage>  
</Description>  
</RDF>
```

RDF Statements

The combination of a Resource, a Property, and a Property value forms a **Statement** (known as the **subject, predicate and object** of a Statement).

Let's look at some example statements to get a better understanding:

Statement: "The author of <http://www.w3schools.com/rdf> is Jan Egil Refsnes".

- The subject of the statement above is: <http://www.w3schools.com/rdf>
- The predicate is: author
- The object is: Jan Egil Refsnes

Statement: "The homepage of <http://www.w3schools.com/rdf> is <http://www.w3schools.com>".

- The subject of the statement above is: <http://www.w3schools.com/rdf>
- The predicate is: homepage
- The object is: <http://www.w3schools.com>

RDF Example

Here are two records from a CD-list:

Title	Artist	Country	Company	Price	Year
Empire Burlesque	Bob Dylan	USA	Columbia	10.90	1985
Hide your heart	Bonnie Tyler	UK	CBS Records	9.90	1988

Below is a few lines from an RDF document:

```
<?xml version="1.0"?>
```

```
<rdf:RDF
```

```
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cd="http://www.recshop.fake/cd#">
```

```
  <rdf:Description
```

```
    rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
```

```
      <cd:artist>Bob Dylan</cd:artist>
```

```
      <cd:country>USA</cd:country>
```

```
      <cd:company>Columbia</cd:company>
```

```
      <cd:price>10.90</cd:price>
```

```
      <cd:year>1985</cd:year>
```

```
    </rdf:Description>
```

```
  <rdf:Description
```

```
    rdf:about="http://www.recshop.fake/cd/Hide your heart">
```

```
      <cd:artist>Bonnie Tyler</cd:artist>
```

```
      <cd:country>UK</cd:country>
```

```
      <cd:company>CBS Records</cd:company>
```

```
      <cd:price>9.90</cd:price>
```

```
      <cd:year>1988</cd:year>
```

```
    </rdf:Description>
```

```
  .
```

```
  .
```

```
  .
```

```
</rdf:RDF>
```

The first line of the RDF document is the XML declaration. The XML declaration is followed by the root element of RDF documents: **<rdf:RDF>**.

The **xmlns:rdf** namespace, specifies that elements with the rdf prefix are from the namespace "http://www.w3.org/1999/02/22-rdf-syntax-ns#".

The **xmlns:cd** namespace, specifies that elements with the cd prefix are from the namespace "http://www.recshop.fake/cd#".

The **<rdf:Description>** element contains the description of the resource identified by the **rdf:about** attribute.

The elements: **<cd:artist>**, **<cd:country>**, **<cd:company>**, etc. are properties of the resource.

RDF Main Elements

The <rdf:RDF> Element

<rdf:RDF> is the root element of an RDF document. It defines the XML document to be an RDF document. It also contains a reference to the RDF namespace:

```
<?xml version="1.0"?>
```

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
...Description goes here...
</rdf:RDF>
```

The <rdf:Description> Element

The <rdf:Description> element identifies a resource with the about attribute.

The <rdf:Description> element contains elements that describe the resource:

```
<?xml version="1.0"?>
```

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>

</rdf:RDF>
```

The elements, artist, country, company, price, and year, are defined in the <http://www.recshop.fake/cd/> namespace. This namespace is outside RDF (and not a part of RDF). RDF defines only the framework. The elements, artist, country, company, price, and year, must be defined by someone else (company, organization, person, etc).

Properties as Attributes

The property elements can also be defined as attributes (instead of elements):

```
<?xml version="1.0"?>
```

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">
```

```
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque"
cd:artist="Bob Dylan" cd:country="USA"
cd:company="Columbia" cd:price="10.90"
cd:year="1985" />
```

```
</rdf:RDF>
```

Properties as Resources

The property elements can also be defined as resources:

```
<?xml version="1.0"?>
```

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">
```

```
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist rdf:resource="http://www.recshop.fake/cd/dylan" />
  ...
  ...
</rdf:Description>
```

```
</rdf:RDF>
```

In the example above, the property artist does not have a value, but a reference to a resource containing information about the artist.

RDF Container Elements

RDF containers are used to describe group of things.

The following RDF elements are used to describe groups: <Bag>, <Seq>, and <Alt>.

The <rdf:Bag> Element

The <rdf:Bag> element is used to describe a list of values that does not have to be in a special order.

The <rdf:Bag> element may contain duplicate values.

Example

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cd="http://www.recshop.fake/cd#">

  <rdf:Description
    rdf:about="http://www.recshop.fake/cd/Beatles">
    <cd:artist>
      <rdf:Bag>
        <rdf:li>John</rdf:li>
        <rdf:li>Paul</rdf:li>
        <rdf:li>George</rdf:li>
        <rdf:li>Ringo</rdf:li>
      </rdf:Bag>
    </cd:artist>
  </rdf:Description>

</rdf:RDF>
```

The <rdf:Seq> Element

The <rdf:Seq> element is used to describe an ordered list of values (For example, in alphabetical order).

The <rdf:Seq> element may contain duplicate values.

Example

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cd="http://www.recshop.fake/cd#">
```

```
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:artist>
    <rdf:Seq>
      <rdf:li>George</rdf:li>
      <rdf:li>John</rdf:li>
      <rdf:li>Paul</rdf:li>
      <rdf:li>Ringo</rdf:li>
    </rdf:Seq>
  </cd:artist>
</rdf:Description>

</rdf:RDF>
```

The <rdf:Alt> Element

The <rdf:Alt> element is used to describe a list of alternative values (the user can select only one of the values).

Example

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:format>
    <rdf:Alt>
      <rdf:li>CD</rdf:li>
      <rdf:li>Record</rdf:li>
      <rdf:li>Tape</rdf:li>
    </rdf:Alt>
  </cd:format>
</rdf:Description>

</rdf:RDF>
```

RDF Terms

In the examples above we have talked about "list of values" when describing the container elements. In RDF these "list of values" are called members.

So, we have the following:

- A container is a resource that contains things
- The contained things are called members (not list of values)

RDF Collections

RDF collections describe groups that can contain ONLY the specified members.

The `rdf:parseType="Collection"` Attribute

As seen in the previous chapter, a container says that the containing resources are members - it does not say that other members are not allowed.

RDF collections are used to describe groups that can contains ONLY the specified members.

A collection is described by the attribute `rdf:parseType="Collection"`.

Example

```
<?xml version="1.0"?>
```

```
<rdf:RDF
```

```
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cd="http://recshop.fake/cd#">
```

```
<rdf:Description
```

```
  rdf:about="http://recshop.fake/cd/Beatles">
```

```
  <cd:artist rdf:parseType="Collection">
```

```
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/George"/>
```

```
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/John"/>
```

```
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Paul"/>
```

```
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Ringo"/>
```

```
  </cd:artist>
```

```
</rdf:Description>
```

```
</rdf:RDF>
```

RDF Schema (RDFS)

RDF Schema (RDFS) is an extension to RDF.

RDF Schema and Application Classes

RDF describes resources with classes, properties, and values.

In addition, RDF also need a way to define application-specific classes and properties. Application-specific classes and properties must be defined using extensions to RDF.

One such extension is RDF Schema.

RDF Schema (RDFS)

RDF Schema does not provide actual application-specific classes and properties.

Instead RDF Schema provides the framework to describe application-specific classes and properties.

Classes in RDF Schema is much like classes in object oriented programming languages. This allows resources to be defined as instances of classes, and subclasses of classes.

RDFS Example

The following example demonstrates some of the RDFS facilities:

```
<?xml version="1.0"?>
```

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#">
```

```
<rdf:Description rdf:ID="animal">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
```

```
</rdf:Description>
```

```
<rdf:Description rdf:ID="horse">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>
```

```
</rdf:RDF>
```

In the example above, the resource "horse" is a subclass of the class "animal".

Example Abbreviated

Since an RDFS class is an RDF resource we can abbreviate the example above by using `rdfs:Class` instead of `rdf:Description`, and drop the `rdf:type` information:

```
<?xml version="1.0"?>
```

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.animals.fake/animals#">
```

```
<rdfs:Class rdf:ID="animal" />
```

```
<rdfs:Class rdf:ID="horse">
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdfs:Class>
```

```
</rdf:RDF>
```

RDF Dublin Core Metadata Initiative

The Dublin Core Metadata Initiative (DCMI) has created some predefined properties for describing documents.

The first Dublin Core properties were defined at the **Metadata Workshop in Dublin, Ohio** in 1995 and is currently maintained by the Dublin Core Metadata Initiative

Property	Definition
Contributor	An entity responsible for making contributions to the content of the resource
Coverage	The extent or scope of the content of the resource
Creator	An entity primarily responsible for making the content of the resource
Format	The physical or digital manifestation of the resource
Date	A date of an event in the lifecycle of the resource
Description	An account of the content of the resource
Identifier	An unambiguous reference to the resource within a given context
Language	A language of the intellectual content of the resource
Publisher	An entity responsible for making the resource available
Relation	A reference to a related resource
Rights	Information about rights held in and over the resource
Source	A Reference to a resource from which the present resource is derived
Subject	A topic of the content of the resource
Title	A name given to the resource
Type	The nature or genre of the content of the resource

A quick look at the table above indicates that RDF is ideal for representing Dublin Core information.

RDF Example

The following example demonstrates the use of some of the Dublin Core properties in an RDF document:

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">

<rdf:Description rdf:about="http://www.w3schools.com">
  <dc:description>W3Schools - Free tutorials</dc:description>
  <dc:publisher>Refsnes Data as</dc:publisher>
  <dc:date>2008-09-01</dc:date>
  <dc:type>Web Development</dc:type>
  <dc:format>text/html</dc:format>
  <dc:language>en</dc:language>
</rdf:Description>

</rdf:RDF>
```

4. SPARQL Query Language for RDF

4.1 Introduction

RDF is a directed, labeled graph data format for representing information in the Web.

RDF is often used to represent, among other things, personal information, social networks, metadata about digital artifacts, as well as to provide a means of integration over disparate sources of information. This specification defines the syntax and semantics of the SPARQL query language for RDF.

The SPARQL query language for RDF is designed to meet the use cases and requirements identified by the RDF Data Access Working Group in *RDF Data Access Use Cases and Requirements*[UCNR].

The SPARQL query language is closely related to the following specifications:

- The SPARQL Protocol for RDF [SPROT] specification defines the remote protocol for issuing SPARQL queries and receiving the results.
- The SPARQL Query Results XML Format [RESULTS] specification defines an XML document format for representing the results of SPARQL SELECT and ASK queries.

4.2 SPARQL Syntax

4.2.1 RDF Term Syntax

4.2.1.1 Syntax for IRIs

The IRIref production designates the set of IRIs [RFC3987]; IRIs are a generalization of URIs [RFC3986] and are fully compatible with URIs and URLs. The PrefixedName production designates a prefixed name. The mapping from a prefixed name to an IRI is described below. IRI references (relative or absolute IRIs) are designated by the IRI_REF production, where the '<' and '>' delimiters do not form part of the IRI reference. Relative IRIs match the irelative-ref reference in section 2.2 ABNF for IRI References and IRIs in [RFC3987] and are resolved to IRIs as described below.

Grammar rules:

```
[67]  IRIref           ::= IRI_REF | PrefixedName
[68]  PrefixedName   ::= PNAME_LN | PNAME_NS
[69]  BlankNode      ::= BLANK_NODE_LABEL | ANON
```

[70]	IRI_REF	::=	'<' ([^<>"{} ^\\]-[#x00-#x20])* '>'
[71]	PNAME_NS	::=	PN_PREFIX? ':'
[72]	PNAME_LN	::=	PNAME_NS PN_LOCAL

The set of RDF terms defined in RDF Concepts and Abstract Syntax includes RDF URI references while SPARQL terms include IRIs. RDF URI references containing "<", ">", '"' (double quote), space, "{", "}", "|", "\", "^", and "`" are not IRIs. The behavior of a SPARQL query against RDF statements composed of such RDF URI references is not defined.

Prefixed names

The PREFIX keyword associates a prefix label with an IRI. A prefixed name is a prefix label and a local part, separated by a colon ":". A prefixed name is mapped to an IRI by concatenating the IRI associated with the prefix and the local part. The prefix label or the local part may be empty. Note that SPARQL local names allow leading digits while XML local names do not.

Relative IRIs

Relative IRIs are combined with base IRIs as per Uniform Resource Identifier (URI): Generic Syntax [RFC3986] using only the basic algorithm in Section 5.2. Neither Syntax-Based Normalization nor Scheme-Based Normalization (described in sections 6.2.2 and 6.2.3 of RFC3986) are performed. Characters additionally allowed in IRI references are treated in the same way that unreserved characters are treated in URI references, per section 6.5 of Internationalized Resource Identifiers (IRIs) [RFC3987].

The BASE keyword defines the Base IRI used to resolve relative IRIs per RFC3986 section 5.1.1, "Base URI Embedded in Content". Section 5.1.2, "Base URI from the Encapsulating Entity" defines how the Base IRI may come from an encapsulating document, such as a SOAP envelope with an xml:base directive or a mime multipart document with a Content-Location header. The "Retrieval URI" identified in 5.1.3, Base "URI from the Retrieval URI", is the URL from which a particular SPARQL query was retrieved. If none of the above specifies the Base URI, the default Base URI (section 5.1.4, "Default Base URI") is used.

The following fragments are some of the different ways to write the same IRI:

```
<http://example.org/book/book1>  
BASE <http://example.org/book/>  
<book1>  
PREFIX book: <http://example.org/book/>  
book:book1
```

4.2..2 Syntax for Literals

The general syntax for literals is a string (enclosed in either double quotes, "...", or single quotes, '...'), with either an optional language tag (introduced by @) or an optional datatype IRI or prefixed name (introduced by ^^).

As a convenience, integers can be written directly (without quotation marks and an explicit datatype IRI) and are interpreted as typed literals of datatype `xsd:integer`; decimal numbers for which there is '.' in the number but no exponent are interpreted as `xsd:decimal`; and numbers with exponents are interpreted as `xsd:double`. Values of type `xsd:boolean` can also be written as `true` or `false`.

To facilitate writing literal values which themselves contain quotation marks or which are long and contain newline characters, SPARQL provides an additional quoting construct in which literals are enclosed in three single- or double-quotation marks.

Examples of literal syntax in SPARQL include:

- "chat"
- 'chat'@fr with language tag "fr"
- "xyz"^^<http://example.org/ns/userDatatype>
- "abc"^^appNS:appDataType
- '''The librarian said, "Perhaps you would enjoy 'War and Peace'.'''
- 1, which is the same as "1"^^xsd:integer
- 1.3, which is the same as "1.3"^^xsd:decimal
- 1.300, which is the same as "1.300"^^xsd:decimal
- 1.0e6, which is the same as "1.0e6"^^xsd:double
- true, which is the same as "true"^^xsd:boolean
- false, which is the same as "false"^^xsd:boolean

Grammar rules:

[60]	RDFLiteral	::=	String (LANGTAG ('^' IRIref))?
[61]	NumericLiteral	::=	NumericLiteralUnsigned NumericLiteralPositive NumericLiteralNegative
[62]	NumericLiteralUnsigned	::=	INTEGER DECIMAL DOUBLE

[63]	NumericLiteralPositive	::=	INTEGER_POSITIVE DECIMAL_POSITIVE DOUBLE_POSITIVE
[64]	NumericLiteralNegative	::=	INTEGER_NEGATIVE DECIMAL_NEGATIVE DOUBLE_NEGATIVE
[65]	BooleanLiteral	::=	'true' 'false'
[66]	String	::=	STRING_LITERAL1 STRING_LITERAL2 STRING_LITERAL_LONG1 STRING_LITERAL_LONG2
[76]	LANGTAG	::=	'@' [a-zA-Z]+ ('-' [a-zA-Z0-9]+)*
[77]	INTEGER	::=	[0-9]+
[78]	DECIMAL	::=	[0-9]+ '.' [0-9]* '.' [0-9]+
[79]	DOUBLE	::=	[0-9]+ '.' [0-9]* EXPONENT '.' ([0-9]) ⁺ EXPONENT ([0-9]) ⁺ EXPONENT
[80]	INTEGER_POSITIVE	::=	'+' INTEGER
[81]	DECIMAL_POSITIVE	::=	'+' DECIMAL
[82]	DOUBLE_POSITIVE	::=	'+' DOUBLE
[83]	INTEGER_NEGATIVE	::=	'-' INTEGER
[84]	DECIMAL_NEGATIVE	::=	'-' DECIMAL
[85]	DOUBLE_NEGATIVE	::=	'-' DOUBLE
[86]	EXPONENT	::=	[eE] [+]? [0-9]+
[87]	STRING_LITERAL1	::=	" " (([^#x27#x5C#xA#xD]) ECHAR) [*] " "
[88]	STRING_LITERAL2	::=	' ' (([^#x22#x5C#xA#xD]) ECHAR) [*] ' '

Tokens matching the productions INTEGER, DECIMAL, DOUBLE and

BooleanLiteral are equivalent to a typed literal with the lexical value of the token and the corresponding datatype (xsd:integer, xsd:decimal, xsd:double, xsd:boolean).

4.2.3 Syntax for Query Variables

Query variables in SPARQL queries have global scope; use of a given variable name anywhere in a query identifies the same variable. Variables are prefixed by either "?" or "\$"; the "?" or "\$" is not part of the variable name. In a query, \$abc and ?abc identify the same variable. The possible names for variables are given in the SPARQL grammar.

Grammar rules:

```
[ 44 ] Var      ::= VAR1 | VAR2
[ 74 ] VAR1     ::= '?' VARNAME
[ 75 ] VAR2     ::= '$' VARNAME
[ 97 ] VARNAME  ::= ( PN_CHARS_U | [0-9] ) ( PN_CHARS_U | [0-9] | #x0
                    [#x0300-#x036F] | [#x203F-#x2040] )*
```

4.2.4 Syntax for Blank Nodes

Blank nodes in graph patterns act as non-distinguished variables, not as references to specific blank nodes in the data being queried.

Blank nodes are indicated by either the label form, such as "_:abc", or the abbreviated form "[]". A blank node that is used in only one place in the query syntax can be indicated with []. A unique blank node will be used to form the triple pattern. Blank node labels are written as "_:abc" for a blank node with label "abc". The same blank node label cannot be used in two different basic graph patterns in the same query.

The [:p :v] construct can be used in triple patterns. It creates a blank node label which is used as the subject of all contained predicate-object pairs. The created blank node can also be used in further triple patterns in the subject and object positions.

The following two forms

```
[ :p "v" ] .
[] :p "v" .
```

allocate a unique blank node label (here "b57") and are equivalent to writing:

```
_:b57 :p "v" .
```

This allocated blank node label can be used as the subject or object of further triple patterns. For example, as a subject:

```
[ :p "v" ] :q "w" .
```

which is equivalent to the two triples:

```
_:b57 :p "v" .  
_:b57 :q "w" .
```

and as an object:

```
:x :q [ :p "v" ] .
```

which is equivalent to the two triples:

```
:x :q _:b57 .  
_:b57 :p "v" .
```

Abbreviated blank node syntax can be combined with other abbreviations for common subjects and common predicates.

```
[ foaf:name ?name ;  
  foaf:mbox <mailto:alice@example.org> ]
```

This is the same as writing the following basic graph pattern for some uniquely allocated blank node label, "b18":

```
_:b18 foaf:name ?name .  
_:b18 foaf:mbox <mailto:alice@example.org> .
```

Grammar rules:

```
[ 39] BlankNodePropertyList ::= ['PropertyListNotEmpty']  
[ 69] BlankNode             ::= BLANK_NODE_LABEL | ANON  
[ 73] BLANK_NODE_LABEL      ::= '_:' PN_LOCAL  
[ 94] ANON                  ::= '[' WS* '']
```

4.2.5 Syntax for Triple Patterns

Triple Patterns are written as a whitespace-separated list of a subject, predicate and object; there are abbreviated ways of writing some common triple pattern constructs.

The following examples express the same query:

```

PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { <http://example.org/book/book1> dc:title ?title }
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://example.org/book/>

```

```

SELECT $title
WHERE { :book1 dc:title $title }
BASE <http://example.org/book/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

```

```

SELECT $title
WHERE { <book1> dc:title ?title }

```

Grammar rules:

```

[32] TriplesSameSubject ::= VarOrTerm PropertyListNotEmpty |
                          TriplesNode PropertyList
[33] PropertyListNotEmpty ::= Verb ObjectList ( ';' ( Verb ObjectList
                          ) )*
[34] PropertyList ::= PropertyListNotEmpty?
[35] ObjectList ::= Object( ';' Object ) *
[37] Verb ::= VarOrIRIref | 'a'

```

4.2.6 Predicate-Object Lists

Triple patterns with a common subject can be written so that the subject is only written once and is used for more than one triple pattern by employing the ";" notation.

```

?x foaf:name ?name ;
   foaf:mbox ?mbox .

```

This is the same as writing the triple patterns:

```

?x foaf:name ?name .
?x foaf:mbox ?mbox .

```

4.2.7 Object Lists

If triple patterns share both subject and predicate, the objects may be separated by ",".

```

?x foaf:nick "Alice" , "Alice_" .

```

is the same as writing the triple patterns:

```

?x foaf:nick "Alice" .
?x foaf:nick "Alice_" .

```

Object lists can be combined with predicate-object lists:

```
?x foaf:name ?name ; foaf:nick "Alice" , "Alice_" .
```

is equivalent to:

```
?x foaf:name ?name .  
?x foaf:nick "Alice" .  
?x foaf:nick "Alice_" .
```

4.2.8 RDF Collections

RDF collections can be written in triple patterns using the syntax "(element1 element2 ...)". The form "()" is an alternative for the IRI <http://www.w3.org/1999/02/22-rdf-syntax-ns#nil>. When used with collection elements, such as (1 ?x 3 4), triple patterns with blank nodes are allocated for the collection. The blank node at the head of the collection can be used as a subject or object in other triple patterns. The blank nodes allocated by the collection syntax do not occur elsewhere in the query.

```
(1 ?x 3 4) :p "w" .
```

is syntactic sugar for (noting that b0, b1, b2 and b3 do not occur anywhere else in the query):

```
_:b0 rdf:first 1 ;  
      rdf:rest _:b1 .  
_:b1 rdf:first ?x ;  
      rdf:rest _:b2 .  
_:b2 rdf:first 3 ;  
      rdf:rest _:b3 .  
_:b3 rdf:first 4 ;  
      rdf:rest rdf:nil .  
_:b0 :p "w" .
```

RDF collections can be nested and can involve other syntactic forms:

```
(1 [:p :q] ( 2 ) ) .
```

is syntactic sugar for:

```
_:b0 rdf:first 1 ;  
      rdf:rest _:b1 .  
_:b1 rdf:first _:b2 .  
_:b2 :p :q .  
_:b1 rdf:rest _:b3 .  
_:b3 rdf:first _:b4 .  
_:b4 rdf:first 2 ;  
      rdf:rest rdf:nil .  
_:b3 rdf:rest rdf:nil .
```

Grammar rules:

```
[40] Collection ::= '(' GraphNode+ ')'  
[92] NIL ::= '(' WS* ')'
```

4.2.9 rdf:type

The keyword "a" can be used as a predicate in a triple pattern and is an alternative for the IRI <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>. This keyword is case-sensitive.

```
?x a :Class1 .
```

```
[ a :appClass ] :p "v" .
```

is syntactic sugar for:

```
?x rdf:type :Class1 .
```

```
_:b0 rdf:type :appClass .
```

```
_:b0 :p "v" .
```

5 OWL Web Ontology Language

5.1. Introduction

The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

OWL is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL.

The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web. The Semantic Web will build on XML's ability to define customized tagging schemes and RDF's flexible approach

to representing data. The first level above RDF required for the Semantic Web is an ontology language what can formally describe the meaning of terminology used in Web documents. If machines are expected to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema. The OWL Use Cases and Requirements Document provides more details on ontologies, motivates the need for a Web Ontology Language in terms of six use cases, and formulates design goals, requirements and objectives for OWL.

OWL has been designed to meet this need for a Web Ontology Language. OWL is part of the growing stack of W3C recommendations related to the Semantic Web.

OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

5.2 The three sublanguages of OWL

OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users

-OWL Lite

supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL, see the section on OWL Lite in the OWL Reference for further details.

-OWL DL

supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with [*description logics*](#), a field of research that has studied the logics that form the

formal foundation of OWL.

-OWL Full

is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold. Their inverses do not.

1. Every legal OWL Lite ontology is a legal OWL DL ontology.
2. Every legal OWL DL ontology is a legal OWL Full ontology.
3. Every valid OWL Lite conclusion is a valid OWL DL conclusion.
4. Every valid OWL DL conclusion is a valid OWL Full conclusion.

Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more-expressive constructs provided by OWL DL. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modeling facilities of RDF Schema (e.g. defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable since complete OWL Full implementations do not currently exist.

OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be viewed as extensions of a restricted view of RDF. Every OWL (Lite, DL, Full) document is an RDF document, and every RDF document is an OWL Full document, but only some RDF documents will be a legal OWL Lite or OWL DL document. Because of this, some care has to be taken when a user wants to migrate an RDF document to OWL. When the expressiveness of OWL DL or OWL Lite is deemed appropriate, some precautions have to be taken to ensure that the original RDF document complies with the additional constraints imposed by OWL DL and OWL Lite. Among others, every URI that is used as a class name must be explicitly asserted to be of type `owl:Class` (and similarly for properties), every individual must be asserted to belong to at least one class (even if only `owl:Thing`), the URI's used for classes, properties and individuals must be mutually disjoint. The details of these and other constraints on OWL DL and

OWL Lite are explained in appendix E of the OWL Reference.

5.3 Language Synopsis

This section provides a quick index to all the language features for OWL Lite, OWL DL, and OWL Full.

5.3.1 OWL Lite Synopsis

The list of OWL Lite language constructs is given below.

RDF Schema Features:

- *Class* (*Thing*, *Nothing*)
- *rdfs:subClassesOf*
- *rdf:Property*
- *rdfs:subPropertyOf*
- *rdfs:domain*
- *rdfs:range*
- *Individual*

(In)Equality:

- *equivalentClass*
- *equivalentProperty*
- *sameAs*
- *differentFrom*
- *AllDifferent*
- *distinctMembers*

Property Characteristics:

- *ObjectProperty*
- *DatatypeProperty*
- *inverseOf*
- *TransitiveProperty*
- *SymmetricProperty*
- *FunctionalProperty*
- *InverseFunctionalProperty*

Property Restrictions:

- *Restriction*
- *onProperty*
- *allValuesFrom*
- *someValuesFrom*

Restricted Cardinality:

- *minCardinality*(only 0 or 1)
- *maxCardinality*(only 0 or 1)
- *cardinality*(only 0 or 1)

Header Information:

- *Ontology*
 - *imports*
-
-

Class	Versioning:	Annotation
Intersection: <ul style="list-style-type: none"> • <i>intersection</i> <i>f</i> 	<ul style="list-style-type: none"> • <i>versionInfo</i> • <i>priorVersion</i> • <i>backwardCompatibleWith</i> • <i>incompatibleWith</i> • <i>DeprecatedClasses</i> • <i>DeprecatedProperty</i> 	Properties: <ul style="list-style-type: none"> • <i>rdfs:label</i> • <i>rdfs:comment</i> • <i>rdfs:seeAlso</i> • <i>rdfs:isDefinedBy</i> • <i>AnnotationProperty</i> • <i>OntologyProperty</i>
Datatypes <ul style="list-style-type: none"> • <i>xsd</i> <i>datatypes</i> 		

5.3.2 OWL DL and Full Synopsis

The list of OWL DL and OWL Full language constructs that are in addition to or expand those of OWL Lite is given below.

Class Axioms:

- *oneOf, dataRange*
- *disjointWith*
- *equivalentClass*
(applied to class expressions)
- *rdfs:subClassOf*
(applied to class expressions)

Boolean Combinations of Class Expressions:

- *unionOf*
- *complementOf*
- *intersectionOf*

Arbitrary Cardinality:

- *minCardinality*
- *maxCardinality*
- *cardinality*

Filler Information:

- *hasValue*

5.4. Language Description of OWL Lite

This section provides an informal description of the OWL Lite language features

OWL Lite uses only some of the OWL language features and has more limitations on the use of the features than OWL DL or OWL Full. For example, in OWL Lite classes can only be defined in terms of named superclasses (superclasses cannot be arbitrary expressions), and only certain kinds of class restrictions can be used. Equivalence between classes and subclass relationships between classes are also only allowed between named classes, and not between arbitrary class expressions. Similarly, restrictions in OWL Lite use only named classes. OWL Lite also has a limited notion of cardinality - the only cardinalities allowed to be explicitly stated are 0 or 1.

5.4.1 OWL Lite RDF Schema Features

The following OWL Lite features related to RDF Schema are included.

Class: A class defines a group of individuals that belong together because they share some properties. For example, Deborah and Frank are both members of the class Person. Classes can be organized in a specialization hierarchy using *subClassOf*. There is a built-in most general class named Thing that is the class of all individuals and is a superclass of all OWL classes. There is also a built-in most specific class named Nothing that is the class that has no instances and a subclass of all OWL classes.

***rdfs:subClassOf*:** Class hierarchies may be created by making one or more statements that a class is a subclass of another class. For example, the class Person could be stated to be a subclass of the class Mammal. From this a reasoner can deduce that if an individual is a Person, then it is also a Mammal.

***rdf:Property*:** Properties can be used to state relationships between individuals or from individuals to data values. Examples of properties include hasChild, hasRelative, hasSibling, and hasAge. The first three can be used to relate an instance of a class Person to another instance of the class Person (and are thus occurrences of ObjectProperty), and the last (hasAge) can be used to relate an instance of the class Person to an instance of the datatype Integer (and is thus an occurrence of DatatypeProperty). Both owl:ObjectProperty and owl:DatatypeProperty are subclasses of the RDF class rdf:Property.

***rdfs:subPropertyOf*:** Property hierarchies may be created by making one or more statements that a property is a subproperty of one or more other properties. For example, hasSibling may be stated to be a subproperty of hasRelative. From this a reasoner can deduce that if an individual is related to another by the hasSibling property, then it is also related to the other by the hasRelative property.

***rdfs:domain*:** A domain of a property limits the individuals to which the

property can be applied. If a property relates an individual to another individual, and the property has a class as one of its domains, then the individual must belong to the class. For example, the property `hasChild` may be stated to have the domain of `Mammal`. From this a reasoner can deduce that if Frank `hasChild` Anna, then Frank must be a `Mammal`. Note that *`rdfs:domain`* is called a global restriction since the restriction is stated on the property and not just on the property when it is associated with a particular class. See the discussion below on property restrictions for more information.

`rdfs:range`: The range of a property limits the individuals that the property may have as its value. If a property relates an individual to another individual, and the property has a class as its range, then the other individual must belong to the range class. For example, the property `hasChild` may be stated to have the range of `Mammal`. From this a reasoner can deduce that if Louise is related to Deborah by the `hasChild` property, (i.e., Deborah is the child of Louise), then Deborah is a `Mammal`. Range is also a global restriction as is domain above. Again, see the discussion below on local restrictions (e.g. `AllValuesFrom`) for more information.

Individual: Individuals are instances of classes, and properties may be used to relate one individual to another. For example, an individual named Deborah may be described as an instance of the class `Person` and the property `hasEmployer` may be used to relate the individual Deborah to the individual `StanfordUniversity`.

5.4.2 OWL Lite Equality and Inequality

The following OWL Lite features are related to equality or inequality.

`equivalentClass`: Two classes may be stated to be equivalent. Equivalent classes have the same instances. Equality can be used to create synonymous classes. For example, `Car` can be stated to be *`equivalentClass`* to `Automobile`. From this a reasoner can deduce that any individual that is an instance of `Car` is also an instance of `Automobile` and vice versa.

`equivalentProperty`: Two properties may be stated to be equivalent. Equivalent properties relate one individual to the same set of other individuals. Equality may be used to create synonymous properties. For example, `hasLeader` may be stated to be the *`equivalentProperty`* to `hasHead`. From this a reasoner can deduce that if X is related to Y by the property `hasLeader`, X is also related to Y by the property `hasHead` and vice versa. A reasoner can also deduce that `hasLeader` is a subproperty of `hasHead` and `hasHead` is a subProperty of `hasLeader`.

`sameAs`: Two individuals may be stated to be the same. These constructs may be used to create a number of different names that refer to the same individual. For example, the individual Deborah may be stated to be the same individual as `DeborahMcGuinness`.

differentFrom: An individual may be stated to be different from other individuals. For example, the individual Frank may be stated to be different from the individuals Deborah and Jim. Thus, if the individuals Frank and Deborah are both values for a property that is stated to be functional (thus the property has at most one value), then there is a contradiction. Explicitly stating that individuals are different can be important in when using languages such as OWL (and RDF) that do not assume that individuals have one and only one name. For example, with no additional information, a reasoner will not deduce that Frank and Deborah refer to distinct individuals.

AllDifferent: A number of individuals may be stated to be mutually distinct in one AllDifferent statement. For example, Frank, Deborah, and Jim could be stated to be mutually distinct using the AllDifferent construct. Unlike the differentFrom statement above, this would also enforce that Jim and Deborah are distinct (not just that Frank is distinct from Deborah and Frank is distinct from Jim). The AllDifferent construct is particularly useful when there are sets of distinct objects and when modelers are interested in enforcing the unique names assumption within those sets of objects. It is used in conjunction with [distinctMembers](#) to state that all members of a list are distinct and pairwise disjoint.

5.4.3 OWL Lite Property Characteristics

There are special identifiers in OWL Lite that are used to provide information concerning properties and their values. The distinction between ObjectProperty and DatatypeProperty is mentioned above in the property description.

inverseOf: One property may be stated to be the inverse of another property. If the property P1 is stated to be the inverse of the property P2, then if X is related to Y by the P2 property, then Y is related to X by the P1 property. For example, if hasChild is the inverse of hasParent and Deborah hasParent Louise, then a reasoner can deduce that Louise hasChild Deborah.

TransitiveProperty: Properties may be stated to be transitive. If a property is transitive, then if the pair (x,y) is an instance of the transitive property P, and the pair (y,z) is an instance of P, then the pair (x,z) is also an instance of P. For example, if ancestor is stated to be transitive, and if Sara is an ancestor of Louise (i.e., (Sara,Louise) is an instance of the property ancestor) and Louise is an ancestor of Deborah (i.e., (Louise,Deborah) is an instance of the property ancestor), then a reasoner can deduce that Sara is an ancestor of Deborah (i.e., (Sara,Deborah) is an instance of the property

ancestor).

OWL Lite (and OWL DL) impose the side condition that transitive properties (and their superproperties) cannot have a `maxCardinality 1` restriction. Without this side-condition, OWL Lite and OWL DL would become undecidable languages. See the property axiom section of the [OWL Semantics and Abstract Syntax](#) document for more information.

SymmetricProperty: Properties may be stated to be symmetric. If a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then the pair (y,x) is also an instance of P. For example, `friend` may be stated to be a symmetric property. Then a reasoner that is given that Frank is a friend of Deborah can deduce that Deborah is a friend of Frank.

FunctionalProperty : Properties may be stated to have a unique value. If a property is a `FunctionalProperty`, then it has no more than one value for each individual (it may have no values for an individual). This characteristic has been referred to as having a unique property. `FunctionalProperty` is shorthand for stating that the property's minimum cardinality is zero and its maximum cardinality is 1. For example, `hasPrimaryEmployer` may be stated to be a `FunctionalProperty`. From this a reasoner may deduce that no individual may have more than one primary employer. This does not imply that every `Person` must have at least one primary employer however.

InverseFunctionalProperty: Properties may be stated to be inverse functional. If a property is inverse functional then the inverse of the property is functional. Thus the inverse of the property has at most one value for each individual. This characteristic has also been referred to as an unambiguous property. For example, `hasUSSocialSecurityNumber` (a unique identifier for United States residents) may be stated to be inverse functional (or unambiguous). The inverse of this property (which may be referred to as `isTheSocialSecurityNumberFor`) has at most one value for any individual in the class of social security numbers. Thus any one person's social security number is the only value for their `isTheSocialSecurityNumberFor` property. From this a reasoner can deduce that no two different individual instances of `Person` have the identical US Social Security Number. Also, a reasoner can deduce that if two instances of `Person` have the same social security number, then those two instances refer to the same individual.

5.4.4 OWL Lite Property Restrictions

OWL Lite allows restrictions to be placed on how properties can be used by instances of a class. These type (and the cardinality restrictions in the next

element indicates the restricted property. The following two restrictions limit which values can be used while the next section's restrictions limit how many values can be used.

allValuesFrom: The restriction *allValuesFrom* is stated on a property with respect to a class. It means that this property on this particular class has a local range restriction associated with it. Thus if an instance of the class is related by the property to a second individual, then the second individual can be inferred to be an instance of the local range restriction class. For example, the class Person may have a property called *hasDaughter* restricted to have *allValuesFrom* the class Woman. This means that if an individual person Louise is related by the property *hasDaughter* to the individual Deborah, then from this a reasoner can deduce that Deborah is an instance of the class Woman. This restriction allows the property *hasDaughter* to be used with other classes, such as the class Cat, and have an appropriate value restriction associated with the use of the property on that class. In this case, *hasDaughter* would have the local range restriction of Cat when associated with the class Cat and would have the local range restriction Person when associated with the class Person. Note that a reasoner can not deduce from an *allValuesFrom* restriction alone that there actually is at least one value for the property.

someValuesFrom: The restriction *someValuesFrom* is stated on a property with respect to a class. A particular class may have a restriction on a property that at least one value for that property is of a certain type. For example, the class SemanticWebPaper may have a *someValuesFrom* restriction on the *hasKeyword* property that states that some value for the *hasKeyword* property should be an instance of the class SemanticWebTopic. This allows for the option of having multiple keywords and as long as one or more is an instance of the class SemanticWebTopic, then the paper would be consistent with the *someValuesFrom* restriction. Unlike *allValuesFrom*, *someValuesFrom* does not restrict all the values of the property to be instances of the same class. If *myPaper* is an instance of the SemanticWebPaper class, then *myPaper* is related by the *hasKeyword* property to at least one instance of the SemanticWebTopic class. Note that a reasoner can not deduce (as it could with *allValuesFrom* restrictions) that all values of *hasKeyword* are instances of the SemanticWebTopic class

5.4.5 OWL Lite Restricted Cardinality

OWL Lite includes a limited form of cardinality restrictions. OWL (and OWL Lite) cardinality restrictions are referred to as local restrictions, since they are stated on properties with respect to a particular class. That is, the restrictions constrain the cardinality of that property on instances of that class. OWL Lite cardinality restrictions are limited because they only allow statements concerning cardinalities of value 0 or 1 (they do not allow arbitrary values for cardinality, as is the case in OWL DL and OWL Full).

minCardinality: Cardinality is stated on a property with respect to a particular class. If a *minCardinality* of 1 is stated on a property with respect to a class, then any instance of that class will be related to at least one individual by that property. This restriction is another way of saying that the property is required to have a value for all instances of the class. For example, the class Person would not have any minimum cardinality restrictions stated on a hasOffspring property since not all persons have offspring. The class Parent, however would have a minimum cardinality of 1 on the hasOffspring property. If a reasoner knows that Louise is a Person, then nothing can be deduced about a minimum cardinality for her hasOffspring property. Once it is discovered that Louise is an instance of Parent, then a reasoner can deduce that Louise is related to at least one individual by the hasOffspring property. From this information alone, a reasoner can not deduce any maximum number of offspring for individual instances of the class parent. In OWL Lite the only minimum cardinalities allowed are 0 or 1. A minimum cardinality of zero on a property just states (in the absence of any more specific information) that the property is optional with respect to a class. For example, the property hasOffspring may have a minimum cardinality of zero on the class Person (while it is stated to have the more specific information of minimum cardinality of one on the class Parent).

maxCardinality: Cardinality is stated on a property with respect to a particular class. If a *maxCardinality* of 1 is stated on a property with respect to a class, then any instance of that class will be related to at most one individual by that property. A *maxCardinality* 1 restriction is sometimes called a functional or unique property. For example, the property hasRegisteredVotingState on the class UnitedStatesCitizens may have a maximum cardinality of one (because people are only allowed to vote in only one state). From this a reasoner can deduce that individual instances of the class USCitizens may not be related to two or more distinct individuals through the hasRegisteredVotingState property. From a maximum cardinality one restriction alone, a reasoner can not deduce a minimum cardinality of 1. It may be useful to state that certain classes have no values for a particular property. For example, instances of the class UnmarriedPerson should not be related to any individuals by the property hasSpouse. This situation is represented by a maximum cardinality of zero on the hasSpouse property on the class UnmarriedPerson.

cardinality: Cardinality is provided as a convenience when it is useful to state that a property on a class has both *minCardinality* 0 and *maxCardinality* 0 or both *minCardinality* 1 and *maxCardinality* 1. For example, the class Person has exactly one value for the property hasBirthMother. From this a reasoner can deduce that no two distinct individual instances of the class Mother may be values for the hasBirthMother property of the same person.

5.4.6 OWL Lite Class Intersection

OWL Lite contains an intersection constructor but limits its usage.

intersectionOf: OWL Lite allows intersections of named classes and restrictions. For example, the class `EmployedPerson` can be described as the *intersectionOf* `Person` and `EmployedThings` (which could be defined as things that have a minimum cardinality of 1 on the `hasEmployer` property). From this a reasoner may deduce that any particular `EmployedPerson` has at least one employer.

5.4.7 OWL Lite Header Information

OWL Lite supports notions of ontology inclusion and relationships and attaching information to ontologies. See the OWL Reference for details and the OWL Guide for examples.

5.4.8 OWL Lite Annotation Properties

OWL Lite allows annotations on classes, properties, individuals and ontology headers. The use of these annotations is subject to certain restrictions. See the section on Annotations in the OWL Reference for details.

5.4.9 OWL Lite Versioning

RDF already has a small vocabulary for describing versioning information. OWL significantly extends this vocabulary. See the OWL Reference for further details.

5.5. Incremental Language Description of OWL DL and OWL Full

Both OWL DL and OWL Full use the same vocabulary although OWL DL is subject to some restrictions. Roughly, OWL DL requires type separation (a class can not also be an individual or property, a property can not also be an individual or class). This implies that restrictions cannot be applied to the language elements

of OWL itself (something that is allowed in OWL Full). Furthermore, OWL DL requires that properties are either ObjectProperties or DatatypeProperties: DatatypeProperties are relations between instances of classes and RDF literals and XML Schema datatypes, while ObjectProperties are relations between instances of two classes. The OWL Semantics and Abstract Syntax document explains the distinctions and limitations. We describe the OWL DL and OWL Full vocabulary that extends the constructions of OWL Lite below.

oneOf: (enumerated classes): Classes can be described by enumeration of the individuals that make up the class. The members of the class are exactly the set of enumerated individuals; no more, no less. For example, the class of daysOfTheWeek can be described by simply enumerating the individuals Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday. From this a reasoner can deduce the maximum cardinality (7) of any property that has daysOfTheWeek as its allValuesFrom restriction.

hasValue: (property values): A property can be required to have a certain individual as a value (also sometimes referred to as property values). For example, instances of the class of dutchCitizens can be characterized as those people that have theNetherlands as a value of their nationality. (The nationality value, theNetherlands, is an instance of the class of Nationalities).

- ***disjointWith***: Classes may be stated to be disjoint from each other. For example, Man and Woman can be stated to be disjoint classes. From this disjointWith statement, a reasoner can deduce an inconsistency when an individual is stated to be an instance of both and similarly a reasoner can deduce that if A is an instance of Man, then A is *not* an instance of Woman.

unionOf, complementOf, intersectionOf(Boolean combinations): OWL DL and OWL Full allow arbitrary Boolean combinations of classes and restrictions: unionOf, complementOf, and intersectionOf. For example, using unionOf, we can state that a class contains things that are either USCitizens or DutchCitizens. Using complementOf, we could state that children are *not* SeniorCitizens. (i.e. the class Children is a subclass of the complement of SeniorCitizens). Citizenship of the European Union could be described as the union of the citizenship of all member states.

minCardinality, maxCardinality, cardinality (full cardinality): While in OWL Lite, cardinalities are restricted to at least, at most or exactly 1 or 0, full OWL allows cardinality statements for arbitrary non-negative integers. For example the class of DINKs ("Dual Income, No Kids") would restrict the cardinality of the property hasIncome to a minimum cardinality of two (while the property hasChild would have to be restricted to cardinality 0).

complex classes : In many constructs, OWL Lite restricts the syntax to single class names (e.g. in `subClassOf` or `equivalentClass` statements). OWL Full extends this restriction to allow arbitrarily complex class descriptions, consisting of enumerated classes, property restrictions, and Boolean combinations. Also, OWL Full allows classes to be used as instances (and OWL DL and OWL Lite do not). For more on this topic, see the "Design for Use" section of the Guide document.
