GKMCET
Lecture Plan

Subject Name: Web Technology

Subject Code: IT2353

UNIT 1
WEB ESSENTIALS: CLIENTS, SERVERS AND COMMUNICATION

1.1. INTERNET

The Internet is a global system of interconnected computer networks that use the standard Internet Protocol Suite (TCP/IP) to serve billions of users worldwide. It is a network of networks that consists of millions of private, public, academic, business, and government networks, of local to global scope, that are linked by a broad array of electronic and optical networking technologies. The Internet carries a vast range of information resources and services, such as the inter-linked hypertext documents of the World Wide Web (WWW) and the infrastructure to support electronic mail.

A Very Brief History of the Internet

Late 1960's to early 1970's

Dept. of Defense Advanced Research Projects Agency (ARPA)

¨ ARPANET served as basis for early networking research as well as a central backbone during the development of the Internet.

¨ TCP/IP evolved as the standard networking protocol for exchanging data between computers on the network.

Mid-To-Late 1970's

Basic services were developed that make up the Internet:

- Remote connectivity
- File Transfer
- Electronic mail

1979-80

Usenet systems for newsgroups

1982

Internet gopher

GKMCET
Lecture Plan

Subject Name: Web Technology

Subject Code: IT2353

1991

Public introduction to World Wide Web (mostly text based)

- In the early 1990s, the developers at CERN spread word of the Web's capabilities to scientific audiences worldwide.
- By September 1993, the share of Web traffic traversing the NSFNET Internet backbone reached 75 gigabytes per month or one percent. By July 1994 it was one terabyte per month.

1994

Prior to this time the WWW was not used for commercial business purposes

- The Internet is one-third research and education network
- Commercial communications begin to take over the majority of Internet traffic

1.2 BASIC INTERNET PROTOCOLS

The Internet set of networks are all based on IP, the Internet Protocol. ``The Internet Protocol (IP) takes care of addressing, or making sure that the routers know what to do with your data when it arrives." Data is transmitted in a series of small chunks, called packets, each approximately 1200 characters in length. The header of each packet includes the destination address of the system to receive the packet. The address of that system is the IP address. All IP addresses consist of four fields of numbers ieach one less than 256. Any packet containing IP addresses in the form

132.246.xxx.xxx

are for systems owned by NRC (actually, it is more precise to say that they are destined for systems behind the main NRC National Capital Region router). The routers in the network use this number to decide where to forward each packet. Some addresses, however, are used for internal networks only. Three sets of addresses were set aside for that specific purpose:

10.0.0.0 - 10.255.255.255

172.16.0.0 - 172.16.255.255

192.168.0.0 - 192.168.255.255

Those sets of addresses should never be routed through the Internet.

About TCP/IP

TCP and IP were developed by a Department of Defense (DOD) research project to connect a number different networks designed by different vendors into a network of networks (the "Internet"). It was initially successful because it delivered a few basic services that everyone needs (file transfer, electronic mail, remote logon) across a very large number of client and server systems. Several computers in a small department can use TCP/IP (along with other protocols) on a single LAN. As with all other communications protocol, TCP/IP is composed of layers:

IP - is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.

TCP - is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.

Sockets - is a name given to the package of subroutines that provide access to TCP/IP on most systems.

Network of Lowest Bidders

The Army puts out a bid on a computer and DEC wins the bid. The Air Force puts out a bid and IBM wins. The Navy bid is won by Unisys. Then the President decides to invade Grenada and the armed forces discover that their computers cannot talk to each other. The DOD must build a "network" out of systems each of which, by law, was delivered by the lowest bidder on a single contract.

The Internet Protocol was developed to create a Network of Networks (the "Internet"). Individual machines are first connected to a LAN (Ethernet or Token Ring). TCP/IP shares the LAN with other uses (a Novell file server, Windows for Workgroups peer systems). One device provides the TCP/IP connection between the LAN and the rest of the world. To insure that all types of systems from all vendors can communicate, TCP/IP is absolutely standardized on the LAN. However, larger networks based on long distances and phone lines are more volatile. In the US, many large corporations would wish to reuse large internal networks based on IBM's SNA.

Addresses

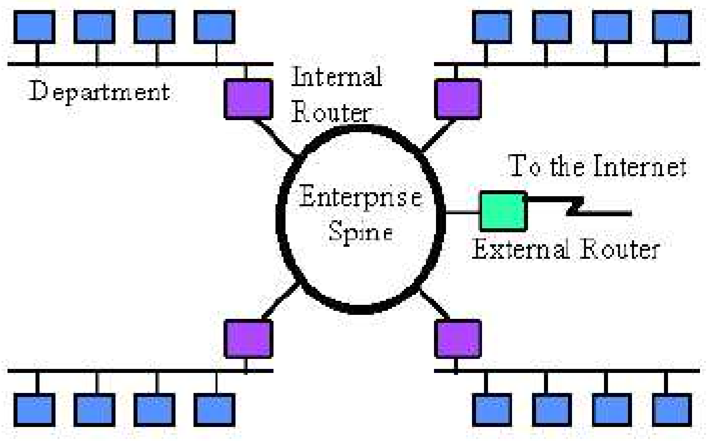
Each technology has its own convention for transmitting messages between two machines within the same network. On a LAN, messages are sent between machines by supplying the six byte unique identifier (the "MAC" address). In an SNA network, every machine has Logical Units with their own network address. DECNET, Appletalk, and Novell IPX all have a scheme for assigning numbers to each local network and to each workstation attached to the network. On top of these local or vendor specific network addresses, TCP/IP assigns a unique number to every workstation in the world. This "IP number" is a four byte value that, by convention, is expressed by converting each byte into a decimal number (0 to 255) and separating the bytes with a period. For example, the PC Lube and Tune server is 130.132.59.234.

Subnets

Although the individual subscribers do not need to tabulate network numbers or provide explicit routing, it is convenient for most Class B networks to be internally managed as a much smaller and simpler version of the larger network organizations. It is common to subdivide the two bytes available for internal assignment into a one byte department number and a one byte workstation ID.

Subject Name: Web Technology

Subject Code: IT2353



The enterprise network is built using commercially available TCP/IP router boxes. Each router has small tables with 255 entries to translate the one byte department number into selection of a destination Ethernet connected to one of the routers. Messages to the PC Lube and Tune server (130.132.59.234) are sent through the national and New England regional networks based on the 130.132 part of the number. Arriving at Yale, the 59 department ID selects an Ethernet connector in the C& IS building. The 234 selects a particular workstation on that LAN.

1.3 THE WORLD WIDE WEB

WWW

The terms Internet and World Wide Web are often used in every-day speech without much
distinction. However, the Internet and the World Wide Web are not one and the same. The
Internet is a global system of interconnected computer networks. In contrast, the Web is one
of the services that runs on the Internet. It is a collection of interconnected documents and other resources, linked by hyperlinks and URLs. In short, the Web is an application running

on the Internet. Viewing a web page on the World Wide Web normally begins either by
typing the URL of the page into a web browser, or by following a hyperlink to that page
or
resource. The web browser then initiates a series of communication messages, behind the
scenes, in order to fetch and display it.First, the server-name portion of the URL is
resolved
into an IP address using the global, distributed Internet database known as the Domain
Name
System (DNS). This IP address is necessary to contact the Web server. The browser then
requests the resource by sending an HTTP request to the Web server at that particular
address.
In the case of a typical web page, the HTML text of the page is requested first and parsed
immediately by the web browser, which then makes additional requests for images and
any
other files that complete the page image.
While receiving these files from the web server, browsers may progressively render the
page
onto the screen as specified by its HTML, Cascading Style Sheets (CSS), or other page
composition languages. Any images and other resources are incorporated to produce the
onscreen
web page that the user sees. Most web pages contain hyperlinks to other related pages
and perhaps to downloadable files, source documents, definitions and other web
resources.
Such a collection of useful, related resources, interconnected via hypertext links is
dubbed a
web of information. Publication on the Internet created what Tim Berners-Lee first called
the
WorldWideWeb

Subject Name: Web Technology

Subject Code: IT2353

Linking
Graphic representation of a minute fraction of the WWW, demonstrating hyperlinks
Over time, many web resources pointed to by hyperlinks disappear, relocate, or are replaced with different content. This makes hyperlinks obsolete, a phenomenon referred to in some circles as link rot and the hyperlinks affected by it are often called dead links. The ephemeral nature of the Web has prompted many efforts to archive web sites. The Internet Archive, active since 1996, is one of the best-known efforts.
Dynamic updates of web pages
JavaScript is a scripting language that was initially developed in 1995 by Brendan Eich, then
of Netscape, for use within web pages.[22] The standardized version is ECMAScript.[22] To
overcome some of the limitations of the page-by-page model described above, some web applications also use Ajax (asynchronous JavaScript and XML). JavaScript is delivered with
the page that can make additional HTTP requests to the server, either in response to user actions such as mouse-clicks, or based on lapsed time. The server's responses are used to modify the current page rather than creating a new page with each response. Thus the server
only needs to provide limited, incremental information. Since multiple Ajax requests can be
handled at the same time, users can interact with a page even while data is being retrieved.
Some web applications regularly poll the server to ask if new information is available.[23]
Caching
If a user revisits a Web page after only a short interval, the page data may not need to be reobtained

from the source Web server. Almost all web browsers cache recently obtained data, usually on the local hard drive. HTTP requests sent by a browser will usually only ask for data that has changed since the last download. If the locally cached data are still current, it
will be reused. Caching helps reduce the amount of Web traffic on the Internet. The decision
about expiration is made independently for each downloaded file, whether image, stylesheet,
JavaScript, HTML, or whatever other content the site may provide. Thus even on sites with
highly dynamic content, many of the basic resources only need to be refreshed occasionally.
Web site designers find it worthwhile to collate resources such as CSS data and JavaScript
into a few site-wide files so that they can be cached efficiently. This helps reduce page download times and lowers demands on the Web server.

1.4 HTTP REQUEST MESSAGES AND RESPONSE MESSAGES

About Hypertext Transfer Protocol -- HTTP/1.0

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods (commands). A feature of HTTP is the typing of data representation, allowing systems to be built independently of the data being transferred. HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification reflects common usage of the protocol referred to as "HTTP/1.0".

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification reflects common usage of the protocol referred to as "HTTP/1.0". This specification describes the features that seem to be consistently implemented in most HTTP/1.0 clients and servers. The specification is split into two sections. Those features of HTTP for which implementations are usually consistent are described in the main body of this document.

Terminology

This specification uses a number of terms to refer to the roles played by participants in, and objects of, the HTTP communication.

connection

A transport layer virtual circuit established between two application programs for the purpose of communication.

message

The basic unit of HTTP communication, consisting of a structured sequence of octets matching the syntax defined in and transmitted via the connection.

request

An HTTP request message.

response

An HTTP response message.

resource

A network data object or service which can be identified by a URI.

entity

A particular representation or rendition of a data resource, or reply from a service resource, that may be enclosed within a request or response message. An entity consists of metainformation in the form of entity headers and content in the form of an entity body.

client

Subject Name: Web Technology

Subject Code: IT2353

An application program that establishes connections for the purpose of
sending requests.
user agent
The client which initiates a request. These are often browsers, editors, spiders
(web-traversing robots), or other end user tools.
server
An application program that accepts connections in order to service requests
by sending back responses.
Overall Operation
The HTTP protocol is based on a request/response paradigm. A client establishes a connection with a server and sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible body content.
Most HTTP communication is initiated by a user agent and consists of a request to be applied to a resource on some origin server. In the simplest case, this may be accomplished via a single connection (v) between the user agent (UA) and the origin server (O).

```
request chain ------------------------>
UA -------------------v------------------- O
<----------------------- response chain
```

A more complicated situation occurs when one or more intermediaries are present in the request/response chain. There are three common forms of intermediary: proxy, gateway, and tunnel. A proxy is a forwarding agent, receiving requests for a URI in its absolute form, rewriting all or parts of the message, and forwarding the reformatted request toward the server identified by the URI. A gateway is a receiving agent, acting as a layer above some other server(s) and, if necessary, translating the requests to the underlying server's protocol. A tunnel acts as a relay point between two connections without changing the messages; tunnels are used when the communication needs to pass through an intermediary (such as a firewall) even when the intermediary cannot understand the contents of the messages.

```
request chain ------------------------------------->
UA -----v----- A -----v----- B -----v----- C -----v----- O
<------------------------------------ response chain
```

The figure above shows three intermediaries (A, B, and C) between the user agent and origin server. A request or response message that travels the whole chain must pass through four separate connections.

On the Internet, HTTP communication generally takes place over TCP/IP connections. The default port is TCP 80 but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used, and the mapping of the HTTP/1.0 request and response structures onto the transport data units of the protocol in question is outside the scope of this specification.

HTTP and MIME

HTTP/1.0 uses many of the constructs defined for MIME, as defined in RFC 1521 describes the ways in which the context of HTTP allows for different use of Internet Media Types than is typically found in Internet mail, and gives the rationale for those differences.

Basic Rules

The following rules are used throughout this specification to describe basic parsing constructs. The US-ASCII coded character set is defined by [17].

OCTET = <any 8-bit sequence of data>

CHAR = <any US-ASCII character (octets 0 - 127)>

UPALPHA = <any US-ASCII uppercase letter "A".."Z">

LOALPHA = <any US-ASCII lowercase letter "a".."z">

ALPHA = UPALPHA | LOALPHA

DIGIT = <any US-ASCII digit "0".."9">

CTL = <any US-ASCII control character

(octets 0 - 31) and DEL (127)>

CR = <US-ASCII CR, carriage return (13)>

LF = <US-ASCII LF, linefeed (10)>

SP = <US-ASCII SP, space (32)>

HT = <US-ASCII HT, horizontal-tab (9)>

<"> = <US-ASCII double-quote mark (34)>

HTTP/1.0 defines the octet sequence CR LF as the end-of-line marker for all protocol elements except the Entity-Body The end-of-line marker within an Entity-Body is defined by its associated media type,

CRLF = CR LF

HTTP/1.0 headers may be folded onto multiple lines if each continuation line begins with a space or horizontal tab. All linear whitespace, including folding, has the same semantics as SP.

LWS = [CRLF] 1*(SP | HT)

However, folding of header lines is not expected by some applications, and should not be generated by HTTP/1.0 applications.

The TEXT rule is only used for descriptive field contents and values that are not intended to be interpreted by the message parser. Words of *TEXT may contain octets from character sets other than US-ASCII.

TEXT = <any OCTET except CTLs,

but including LWS>

Recipients of header field TEXT containing octets outside the US-ASCII character set may assume that they represent ISO-8859-1 characters.

Hexadecimal numeric characters are used in several protocol elements.

HEX = "A" | "B" | "C" | "D" | "E" | "F"

| "a" | "b" | "c" | "d" | "e" | "f" | DIGIT

Many HTTP/1.0 header field values consist of words separated by LWS or special characters. These special characters must be in a quoted string to be used within a parameter value.

word = token | quoted-string

token = 1*<any CHAR except CTLs or tspecials>

tspecials = "(" | ")" | "<" | ">" | "@"

| "," | ";" | ":" | "\" | <">

| "/" | "[" | "]" | "?" | "="

| "{" | "}" | SP | HT

Comments may be included in some HTTP header fields by surrounding the comment text with parentheses. Comments are only allowed in fields containing "comment" as part of their field value definition. In all other fields, parentheses are considered part of the field value.

comment = "(" *(ctext | comment) ")"

ctext = <any TEXT excluding "(" and ")">

A string of text is parsed as a single word if it is quoted using double-quote marks.

quoted-string = (<"> *(qdtext) <">)

qdtext = <any CHAR except <"> and CTLs,

but including LWS>

Single-character quoting using the backslash ("\") character is not permitted in HTTP/1.0.

Protocol Parameters

HTTP Version

HTTP uses a "<major>.<minor>" numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of a message and its capacity for understanding further HTTP communication, rather than the features obtained via that communication. No change is made to the version number for the addition of message components which do not affect communication behavior or which only add to extensible field values. The <minor> number is incremented when the changes made to the protocol add features which do not change the general message parsing algorithm, but which may add to the message semantics and imply additional capabilities of the sender. The <major> number is incremented when the format of a message within the protocol is changed.

The version of an HTTP message is indicated by an HTTP-Version field in the first line of the message. If the protocol version is not specified, the recipient must assume that the message is in the simple HTTP/0.9 format.

HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT

Note that the major and minor numbers should be treated as separate integers and that each may be incremented higher than a single digit. Thus, HTTP/2.4 is a lower version than HTTP/2.13, which in turn is lower than HTTP/12.3. Leading zeros should be ignored by recipients and never generated by senders.

HTTP/1.0 servers must:

recognize the format of the Request-Line for HTTP/0.9 and HTTP/1.0 requests;

understand any valid request in the format of HTTP/0.9 or HTTP/1.0;

respond appropriately with a message in the same protocol version used by the client.

HTTP/1.0 clients must:

recognize the format of the Status-Line for HTTP/1.0 responses;

understand any valid response in the format of HTTP/0.9 or HTTP/1.0.

Proxy and gateway applications must be careful in forwarding requests that are received in a format different than that of the application's native HTTP version. Since the protocol version indicates the protocol capability of the sender, a proxy/gateway must never send a message with a version indicator which is greater than its native version; if a higher version request is received, the proxy/gateway must either downgrade the request version or respond with an error.

Uniform Resource Identifiers

URIs have been known by many names: WWW addresses, Universal Document Identifiers, Universal Resource Identifiers , and finally the combination of Uniform Resource Locators (URL) and Names (URN).As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify--via name, location, or any other characteristic--a network resource.

General Syntax

URIs in HTTP can be represented in absolute form or relative to some known base URI , depending upon the context of their use. The two forms are differentiated by the fact that absolute URIs always begin with a scheme name followed by a colon.

```
URI = ( absoluteURI | relativeURI ) [ "#"
fragment ]
absoluteURI = scheme ":" *( uchar | reserved )
relativeURI = net_path | abs_path | rel_path
net_path = "//" net_loc [ abs_path ]
abs_path = "/" rel_path
rel_path = [ path ] [ ";" params ] [ "?" query ]
path = fsegment *( "/" segment )
fsegment = 1*pchar
segment = *pchar
params = param *( ";" param )
param = *( pchar | "/" )
scheme = 1*( ALPHA | DIGIT | "+" | "-" | "." )
net_loc = *( pchar | ";" | "?" )
query = *( uchar | reserved )
```

fragment = *(uchar | reserved)

pchar = uchar | ":" | "@" | "&" | "=" | "+"

uchar = unreserved | escape

unreserved = ALPHA | DIGIT | safe | extra | national

escape = "%" HEX HEX

reserved = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+"

extra = "!" | "*" | "'" | "(" | ")" | ","

safe = "$" | "-" | "_" | "."

unsafe = CTL | SP | <"> | "#" | "%" | "<" | ">"

national = <any OCTET excluding ALPHA, DIGIT,
reserved, extra, safe, and unsafe>

For definitive information on URL syntax and semantics, see RFC 1738 and RFC 1808 . The BNF above includes national characters not allowed in valid URLs as specified by RFC 1738, since HTTP servers are not restricted in the set of unreserved characters allowed to represent the rel_path part of addresses, and HTTP proxies may receive requests for URIs not defined by RFC 1738.

HTTP URL

The "http" scheme is used to locate network resources via the HTTP protocol. This section defines the scheme-specific syntax and semantics for http URLs.

http_URL = "http:" "//" host [":" port] [abs_path]

host = <A legal Internet host domain name
or IP address (in dotted-decimal form),
as defined by Section 2.1 of RFC 1123>

port = *DIGIT

If the port is empty or not given, port 80 is assumed. The semantics are that the identified resource is located at the server listening for TCP connections on that port of that host, and the Request-URI for the resource is abs_path. If the abs_path is not present in the URL, it must be given as "/" when used as a Request-URI).

Subject Name: Web Technology

Subject Code: IT2353

Note: Although the HTTP protocol is independent of the transport layer protocol, the http URL only identifies resources by their TCP location, and thus non-TCP resources must be identified by some other URI scheme.

Date/Time Formats

HTTP/1.0 applications have historically allowed three different formats for the representation of date/time stamps:

Sun, 06 Nov 1994 08:49:37 GMT ; RFC 822, updated by RFC 1123

Sunday, 06-Nov-94 08:49:37 GMT ; RFC 850, obsoleted by RFC 1036

Sun Nov 6 08:49:37 1994 ; ANSI C's asctime() format

The first format is preferred as an Internet standard and represents a fixed-length subset of that defined by RFC 1123 [6] (an update to RFC 822 [7]). The second format is in common use, but is based on the obsolete RFC 850 [10] date format and lacks a four-digit year. HTTP/1.0 clients and servers that parse the date value should accept all three formats, though they must never generate the third (asctime) format. All HTTP/1.0 date/time stamps must be represented in Universal Time (UT), also known as Greenwich Mean Time (GMT), without exception. This is indicated in the first two formats by the inclusion of "GMT" as the three-letter abbreviation for time zone, and should be assumed when reading the asctime format.

HTTP-date = rfc1123-date | rfc850-date | asctime-date

rfc1123-date = wkday "," SP date1 SP time SP "GMT"

rfc850-date = weekday "," SP date2 SP time SP "GMT"

asctime-date = wkday SP date3 SP time SP 4DIGIT

date1 = 2DIGIT SP month SP 4DIGIT

; day month year (e.g., 02 Jun 1982)

date2 = 2DIGIT "-" month "-" 2DIGIT

; day-month-year (e.g., 02-Jun-82)

date3 = month SP (2DIGIT | (SP 1DIGIT))

; month day (e.g., Jun 2)

Subject Name: Web Technology

Subject Code: IT2353

time = 2DIGIT ":" 2DIGIT ":" 2DIGIT
; 00:00:00 - 23:59:59
wkday = "Mon" | "Tue" | "Wed"
| "Thu" | "Fri" | "Sat" | "Sun"
weekday = "Monday" | "Tuesday" | "Wednesday"
| "Thursday" | "Friday" | "Saturday" | "Sunday"
month = "Jan" | "Feb" | "Mar" | "Apr"
| "May" | "Jun" | "Jul" | "Aug"
| "Sep" | "Oct" | "Nov" | "Dec"

Character Sets

HTTP uses the same definition of the term "character set" as that described for MIME:

The term "character set" is used in this document to refer to a method used with one or more tables to convert a sequence of octets into a sequence of characters. Note that unconditional conversion in the other direction is not required, in that not all characters may be available in a given character set and a character set may provide more than one sequence of octets to represent a particular character.

Note: This use of the term "character set" is more commonly referred to as a "character encoding." However, since HTTP and MIME share the same registry, it is important that the terminology also be shared.

HTTP character sets are identified by case-insensitive tokens. character sets -- and other names specifically recommended for use within MIME charset parameters.

charset = "US-ASCII"
| "ISO-8859-1" | "ISO-8859-2" | "ISO-8859-3"
| "ISO-8859-4" | "ISO-8859-5" | "ISO-8859-6"
| "ISO-8859-7" | "ISO-8859-8" | "ISO-8859-9"
| "ISO-2022-JP" | "ISO-2022-JP-2" | "ISO-2022-KR"
| "UNICODE-1-1" | "UNICODE-1-1-UTF-7" | "UNICODE-1-1-

UTF-8"

| token

Although HTTP allows an arbitrary token to be used as a charset value, any token that has a predefined value within the IANA Character Set registry must represent the character set defined by that registry. Applications should limit their use of character sets to those defined by the IANA registry. The character set of an entity body should be labelled as the lowest common denominator of the character codes used within that body, with the exception that no label is preferred over the labels US-ASCII or ISO-8859-1.

Content Codings

Content coding values are used to indicate an encoding transformation that has been applied to a resource. Content codings are primarily used to allow a document to be compressed or encrypted without losing the identity of its underlying media type. Typically, the resource is stored in this encoding and only decoded before rendering or analogous usage.

content-coding = "x-gzip" | "x-compress" | token

Note: For future compatibility, HTTP/1.0 applications should consider "gzip" and "compress" to be equivalent to "x-gzip" and "x-compress", respectively.

All content-coding values are case-insensitive. HTTP/1.0 uses content-coding values in the Content-Encoding header field.

Media Types

HTTP uses Internet Media Typesin the Content-Type header field) in order to provide open and extensible data typing.

media-type = type "/" subtype *(";" parameter)

type = token

subtype = token

Parameters may follow the type/subtype in the form of attribute/value pairs.

parameter = attribute "=" value

attribute = token

value = token | quoted-string

The type, subtype, and parameter attribute names are case-insensitive. Parameter values may or may not be case-sensitive, depending on the semantics of the parameter name. LWS must not be generated between the type and subtype, nor between an attribute and its value. Upon receipt of a media type with an unrecognized parameter, a user agent should treat the media type as if the unrecognized parameter and its value were not present.

Some older HTTP applications do not recognize media type parameters. HTTP/1.0 applications should only use media type parameters when they are necessary to define the content of a message.

Media-type values are registered with the Internet Assigned Number Authority (IANA [15]). The media type registration process is outlined in RFC 1590 [13]. Use of non-registered media types is discouraged.

HTTP Message

Message Types

HTTP messages consist of requests from client to server and responses from server to client.

HTTP-message = Simple-Request ; HTTP/0.9 messages

| Simple-Response

| Full-Request ; HTTP/1.0 messages

| Full-Response

Full-Request and Full-Response use the generic message format of RFC 822 [7] for transferring entities. Both messages may include optional header fields (also known as "headers") and an entity body. The entity body is separated from the headers by a null line (i.e., a line with nothing preceding the CRLF).

Full-Request = Request-Line ; Section 5.1

*(General-Header ; Section 4.3

Subject Name: Web Technology

Subject Code: IT2353

| Request-Header ; Section 5.2

| Entity-Header) ; Section 7.1

CRLF

[Entity-Body] ; Section 7.2

Full-Response = Status-Line ; Section 6.1

*(General-Header ; Section 4.3

| Response-Header ; Section 6.2

| Entity-Header) ; Section 7.1

CRLF

[Entity-Body] ; Section 7.2

Simple-Request and Simple-Response do not allow the use of any header information and are limited to a single request method (GET).

Simple-Request = "GET" SP Request-URI CRLF

Simple-Response = [Entity-Body]

Use of the Simple-Request format is discouraged because it prevents the server from identifying the media type of the returned entity.

Message Headers

HTTP header fields, which include General-Header (Section 4.3), Request-Header (Section 5.2), Response-Header (Section 6.2), and Entity-Header (Section 7.1) fields, follow the same generic format as that given in Section 3.1 of RFC 822 [7]. Each header field consists of a name followed immediately by a colon (":"), a single space (SP) character, and the field value. Field names are case-insensitive. Header fields can be extended over multiple lines by preceding each extra line with at least one SP or HT, though this is not recommended.

HTTP-header = field-name ":" [field-value] CRLF

field-name = token

field-value = *(field-content | LWS)

field-content = <the OCTETs making up the field-value

and consisting of either *TEXT or
combinations
of token, tspecials, and quoted-string>
The order in which header fields are received is not significant. However, it is "good practice" to send General-Header fields first, followed by Request-Header or Response-Header fields prior to the Entity-Header fields.
Multiple HTTP-header fields with the same field-name may be present in a message if and only if the entire field-value for that header field is defined as a commaseparated list [i.e., #(values)]. It must be possible to combine the multiple header fields into one "field-name: field-value" pair, without changing the semantics of the message, by appending each subsequent field-value to the first, each separated by a comma.
General Header Fields
There are a few header fields which have general applicability for both request and response messages, but which do not apply to the entity being transferred. These headers apply only to the message being transmitted.
General-Header = Date ; |
Pragma ;
General header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of general header fields if all parties in the communication recognize them to be general header fields. Unrecognized header fields are treated as Entity-Header fields.
Request
A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use. For backwards compatibility with the more limited HTTP/0.9 protocol, there are two valid formats for an HTTP request:

Request = Simple-Request | Full-Request
Simple-Request = "GET" SP Request-URI CRLF
Full-Request = Request-Line ;
*(General-Header ;
| Request-Header ;
| Entity-Header) ;
CRLF
[Entity-Body] ;

If an HTTP/1.0 server receives a Simple-Request, it must respond with an HTTP/0.9 Simple-Response. An HTTP/1.0 client capable of receiving a Full-Response should never generate a Simple-Request.

Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF are allowed except in the final CRLF sequence.

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Note that the difference between a Simple-Request and the Request-Line of a Full-Request is the presence of the HTTP-Version field and the availability of methods other than GET.

Method

The Method token indicates the method to be performed on the resource identified by the Request-URI. The method is case-sensitive.

Method = "GET" ;
| "HEAD" ;
| "POST" ;
| extension-method
extension-method = token

The list of methods acceptable by a specific resource can change dynamically; the client is notified through the return code of the response if a method is not allowed on a resource. Servers should return the status code 501 (not implemented) if the method is unrecognized or not implemented.

The methods commonly used by HTTP/1.0 applications are fully defined in .

Request-URI

The Request-URI is a Uniform Resource Identifier and identifies the resource upon which to apply the request.

Request-URI = absoluteURI | abs_path

The two options for Request-URI are dependent on the nature of the request.

The absoluteURI form is only allowed when the request is being made to a proxy. The proxy is requested to forward the request and return the response. If the request is GET or HEAD and a prior response is cached, the proxy may use the cached message if it passes any restrictions in the Expires header field. Note that the proxy may forward the request on to another proxy or directly to the server specified by the absoluteURI. In order to avoid request loops, a proxy must be able to recognize all of its server names, including any aliases, local variations, and the numeric IP address.

An example Request-Line would be:

GET /TheProject.html HTTP/1.0

The most common form of Request-URI is that used to identify a resource on an origin server or gateway. In this case, only the absolute path of the URI is transmitted . For example, a client wishing to retrieve the resource above directly from the origin server would create a TCP connection to port 80 of the host "www.w3.org" and send the line:

GET /pub/WWW/TheProject.html HTTP/1.0

followed by the remainder of the Full-Request. Note that the absolute path cannot be empty; if none is present in the original URI, it must be given as "/" (the server root).

The Request-URI is transmitted as an encoded string, where some characters may be escaped using the "% HEX HEX" encoding defined by RFC 1738 [4]. The origin server must decode the Request-URI in order to properly interpret the request.

Request Header Fields

The request header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers, with semantics equivalent to the parameters on a programming language method (procedure) invocation.

Request-Header = Authorization ;
| From ;
| If-Modified-Since ;
| Referer ;
| User-Agent ;

Request-Header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of request header fields if all parties in the communication recognize them to be request header fields. Unrecognized header fields are treated as Entity-Header fields.

Response

After receiving and interpreting a request message, a server responds in the form of an HTTP response message.

Response = Simple-Response | Full-Response
Simple-Response = [Entity-Body]
Full-Response = Status-Line ;
*(General-Header ;
| Response-Header ;
| Entity-Header) ;
CRLF
[Entity-Body] ;

A Simple-Response should only be sent in response to an HTTP/0.9 Simple-Request or if the server only supports the more limited HTTP/0.9 protocol. If a client sends an HTTP/1.0 Full-Request and receives a response that does not begin with a Status-Line, it should assume that the response is a Simple-Response and parse it accordingly. Note that the Simple-Response consists only of the entity body and is terminated by the server closing the connection.

Status-Line

The first line of a Full-Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase
CRLF

Since a status line always begins with the protocol version and status code
"HTTP/" 1*DIGIT "." 1*DIGIT SP 3DIGIT SP
(e.g., "HTTP/1.0 200 "), the presence of that expression is sufficient to differentiate a Full-Response from a Simple-Response. Although the Simple-Response format may allow such an expression to occur at the beginning of an entity body, and thus cause a misinterpretation of the message if it was given in response to a Full-Request, most HTTP/0.9 servers are limited to responses of type "text/html" and therefore would never generate such a response.

Status Code and Reason Phrase

The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata and the Reason-Phrase is intended for the human user. The client is not required to examine or display the Reason-Phrase.

The first digit of the Status-Code defines the class of response. The last two digits do

not have any categorization role. There are 5 values for the first digit:
1xx: Informational - Not used, but reserved for future use
2xx: Success - The action was successfully received, understood, and accepted.
3xx: Redirection - Further action must be taken in order to complete the request
4xx: Client Error - The request contains bad syntax or cannot be fulfilled
5xx: Server Error - The server failed to fulfill an apparently valid request
The individual values of the numeric status codes defined for HTTP/1.0, and an example set of corresponding Reason-Phrase's, are presented below. The reason phrases listed here are only recommended -- they may be replaced by local equivalents without affecting the protocol. These codes are fully defined in.

```
Status-Code = "200" ; OK
| "201" ; Created
| "202" ; Accepted
| "204" ; No Content
| "301" ; Moved Permanently
| "302" ; Moved Temporarily
| "304" ; Not Modified
| "400" ; Bad Request
| "401" ; Unauthorized
| "403" ; Forbidden
| "404" ; Not Found
| "500" ; Internal Server Error
| "501" ; Not Implemented
| "502" ; Bad Gateway
| "503" ; Service Unavailable
| extension-code
```

extension-code = 3DIGIT

Reason-Phrase = *<TEXT, excluding CR, LF>

Response Header Fields

The response header fields allow the server to pass additional information about the response which cannot be placed in the Status-Line. These header fields give information about the server and about further access to the resource identified by the Request-URI.

Response-Header = Location ;

| Server ;

| WWW-Authenticate ;

Response-Header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of response header fields if all parties in the communication recognize them to be response header fields. Unrecognized header fields are treated as Entity-Header fields.

Method Definitions

The set of common methods for HTTP/1.0 is defined below. Although this set can be expanded, additional methods cannot be assumed to share the same semantics for separately extended clients and servers.

GET

The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

The semantics of the GET method changes to a "conditional GET" if the request message includes an If-Modified-Since header field. A conditional GET method requests that the identified resource be transferred only if it has been modified since the date given by the If-Modified-Since header. The conditional GET method is intended to reduce network usage by allowing cached entities to be refreshed without requiring multiple requests or transferring unnecessary data.

HEAD

The HEAD method is identical to GET except that the server must not return any Entity-Body in the response. The metainformation contained in the HTTP headers in response to a HEAD request should be identical to the information sent in response to a GET request. This method can be used for obtaining metainformation about the resource identified by the Request-URI without transferring the Entity-Body itself. This method is often used for testing hypertext links for validity, accessibility, and recent modification.

There is no "conditional HEAD" request analogous to the conditional GET. If an If-Modified-Since header field is included with a HEAD request, it should be ignored.

POST

The POST method is used to request that the destination server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow a uniform method to cover the following functions:

Annotation of existing resources;

Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;

Providing a block of data, such as the result of submitting a form, to a datahandling process;

Extending a database through an append operation.

The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI. The posted entity is subordinate to that URI in the same way that a file is subordinate to a directory containing it, a news article is subordinate to a newsgroup to which it is posted, or a record is subordinate to a database.

1.6 WEB SERVERS

A Web server is a program that, using the client/server model and the World Wide Web's Hypertext Transfer Protocol (HTTP), serves the files that form Web pages to Web users (whose computers contain HTTP clients that forward their requests). Every computer on the Internet that contains a Web site must have a Web server program. Two leading Web servers are Apache , the most widely-installed Web server, and Microsoft's Internet Information Server (IIS). Other Web servers include Novell's Web Server for users of its NetWare operating system and IBM's family of Lotus Domino servers, primarily for IBM's OS/390 and AS/400 customers.
Apache is the most popular UNIX web server today. Apache was originally based on code and ideas found in the most popular HTTP server of the time, NCSA httpd 1.3 (early 1995). It has since evolved into a far superior system which can rival (and probably surpass) almost any other UNIX based HTTP server in terms of functionality, efficiency and speed. Take a look at the web server feature chart to see how Apache ranks among the competition.Open Market provides software products that are used to develop the infrastructure for Internet commerce. They pride themselves on scalability, content flexibility, lower entry and maintenance costs, and enhanced security.
Netscape sells several web server software packages. The Netscape Enterprise Server offers built in advanced services such as Internet-based access controls, automatic link management, and revision control. The FastTrack Server is an easy-to-use entry-level Web server designed to let novices create and manage a Web site.
IBM's Secure Server is provided for AIX, HP-UX, and Solaris, as well as NT and OS/2. Version 4.2 servers include enhanced scalability, browser-specific response capability, enhanced CGI support, PICS support, and HTTP Version 1.1 compliance. The IB servers have consistent configuration, management, and API interfaces across all of their supported platforms.

Subject Name: Web Technology

Subject Code: IT2353

Jigsaw is W3C's sample implementation of HTTP, a full blown HTTP server entirely written in Java. Its design goals were: will run on any machine running Java, can be extended by writing new resource objects (a replacement for CGI), minimization of file system accesses.

WebSTAR is a Mac HTTP server which performs dynamic web server file caching, has the ability to run server side Java applets, contains an administration plug-in that lets one administer essential server functions from any web browser on the Internet, honors keep-alive requests, supports a 20,000 username/passwd database, has integrated support for image maps, supports common log format, supports cgi-bin folder, does on the fly bin-hexing of Mac files, and supports an expanded command set for server-side includes.

1.7 Web Clients

A Web client is actually browser. It is the browser on PC/Mac that makes the requests to the remote server. A PC/Mac that uses a web (Client) browser is referred to as a Client Machine.. The alpha version supports some HTML 3.2 features, and the bookmark file is an HTML-ish file that can organize bookmarks under groups. The resource file format is the same as the X resource format (Xrm functions are used) but are completely separate. It is recommended that you not use X resource files at all unless you really want to change the attributes of the Athena widgets (e.g. color, fonts, add hidden buttons, etc.). Each group starts with a string delimited by 'h3' tags. Each bookmark is delimited by 'a' tags.

HotJava is Sun's highly customizable, extensible web browser, written entirely in Java. The HotJava Browser conforms to Web standards and standard practice. It is designed to be highly scalable and customizable, enabling end-users, service providers and intranet providers to easily tailor it to meet their specific requirements. Like Web pages themselves, the user interface of the HotJava Browser is implemented using HTML and applets, and its behavior can be modified by an ASCII-based properties file.

Subject Name: Web Technology

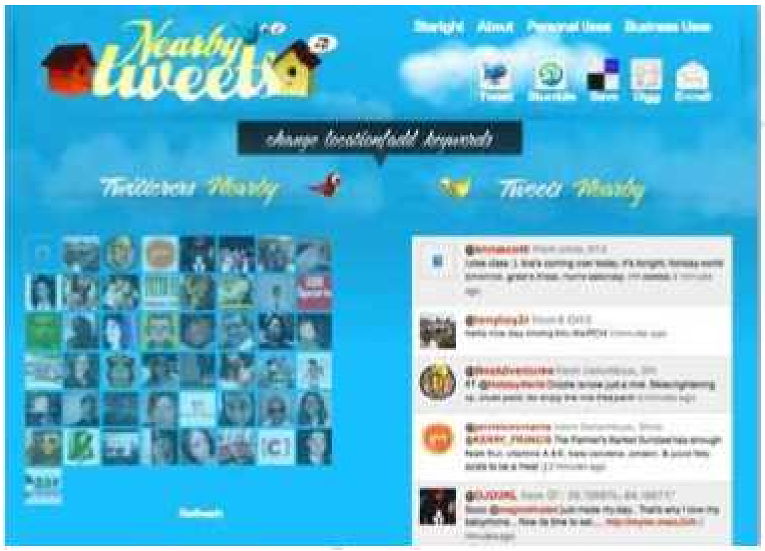
Subject Code: IT2353

Lynx is a text based browser for both UNIX and VMS platforms with more platform development in the works. Lynx is a product of the Distributed Computing Group within Academic Computing Services of The University of Kansas. Lynx was originally developed by Lou Montulli, Michael Grobe, and Charles Rezac. Garrett Blythe created DosLynx and later joined the Lynx effort as well. Currently it is being maintained by members of the Internet community coordinated by Foteos Macrides of the Worcester Foundation for Biological Research.Internet Explorer from Microsoft is available for Win3, Win95, WinNT, and MacOS. MSIE is tightly bundled with the Microsoft Operating environments.NCSA Mosaic is no longer in development, but is still distributing software for the X, MS, and Mac environments.Netscape sells several browsers to the end user and corporate environments. Netscape is the most popular web browser to date and supports the latest HTML version along with other proprietary tags. Netscape browsers are forms, Java, and Java script compliant

1.8 CASE STUDY

Ux Case Study Designing User Focused Web Applications

The beginning there was the first iteration of Nearby Tweets, a simple way to connect with local people and businesses on Twitter.



Subject Name: Web Technology

Subject Code: IT2353

But in product development there is no such thing as perfection, only iterative design—and customers hold all the answers to a better product. So I reached out to my users as soon as I had a chance with Uservoice. Through Uservoice, my users could vote on features they wanted see in the Nearby Tweets redesign, and vote they did.

Top user requested changes to Nearby Tweets

1. Default location
2. Saved keywords and locations
3. Pullout drawer was annoying
4. Manual directory
5. Mobile version
6. Auto-refresh tweets
7. Block users
8. Block locations
9. Follow feature

Choosing which features to implement

While all user requests are valid opportunities, I had to give some thought to my own resources and agenda before to make the right decisions about which requests to accept.

Sorry mobile version, but you'll have to wait

Since I already had a web presence and I have not completely aligned Nearby Tweets' web presence with my goals, I am holding off on the mobile version, because that requires many resources and a new round of considerations, which would set me back and throw me off.

Sorry manual directory, but you're for someone else

There are plenty of apps out there doing manual Twitter directories. I don't have the reputation to compete them in that space, but the auto geolocation space is still mine, and that's where I'll stay for now. (See Twellow, WeFollow, and just tweet it)

Subject Name: Web Technology

Subject Code: IT2353

The rest of you, come with me
The rest of the user-requested features fit within my current agenda for the web platform, which I'd have to also define more clearly before I started in on the UI design.
Narrowing my design scope
In order to keep myself on track, I laid out three design scope requirements for the Nearby Tweets redesign:
1. Address user concerns. I released the first version of Nearby Tweets quickly with little user feedback, just to quietly hit the market with a fun, useful project. With this redesign I wanted to garner as much user feedback during the design process as possible. I did so through Twitter, Uservoice, as well as a private and public beta.
2. Make Nearby Tweets more powerful. Although the focus of the first version was an advantage, it was very basic with no extended functionality. This time Nearby Tweets would supplement its core functionality with user preferences, and advanced search features.
3. Maintain Nearby Tweets' simplicity and focus. All the while I would retain the simplicity of Nearby Tweets and make any effort to further simplify the Nearby Tweets experience.
Then I came up with my design goal
Allow users to watch nearby tweets without any work, but allow users to tweak Nearby Tweets when ready.
Turning concepts into UI design
To keep Nearby Tweets simple while adding advanced functionality I decided to utilize two UI design concepts: Progressive disclosure and lazy registration.
1.9 MARKUP LANGUAGES :XHTML
XHTML reformulates the existing HTML technology to be an application of XML. It is already used for mobile phone and PDA Internet sites, and expected to gain widespread use as XML becomes more popular. Dan Wellman explains the

differences between XHTML and HTML 4, and shows how easy it is to make the switch.

XHTML was invented to try to clean up the mess left by trying to make HTML a fullfledged
presentation tool, and to reduce the fragmentation of HTML caused by the introduction of non-standard elements, mainly by Microsoft and
Netscape. Additionally, XML is viewed by many as the future of the Internet, so reformulating existing markup technology to be an application of XML is a step towards embracing that future and letting go of the long standing legacy of the past. XHTML is now the specification of choice for mobile phone and PDA Internet sites, so its cross platform functionality is already being taken advantage of in excellent ways.

XHTML and its predecessor HTML 4 are extremely similar languages to use. Anyone that has worked with HTML 4 (or XML even) will find switching to XHTML extremely easy; it is more often then not simply learning to break those bad coding habits that many of us have found so easy to slip into. Remembering the few rules that come with XHTML, and trying to avoid any deprecated tags where possible, is all it really takes. In HTML, the mark-up surrounding your content is traditionally referred to as a tag, the <a> tag for example. In XML and subsequently, XHTML, these tags and the text enclosed within them are known as elements. For example:

<p>This sentence, including the opening and closing tags is known as
a paragraph element</p>

<p>This is a separate paragraph element</p>

AN INTRODUCTION TO HTML

HTML, which stands for HyperText Markup Language, is the predominant markup language for web pages. A markup language is a set of markup tags, and HTML uses markup tags to describe web pages.TML is written in the form of HTML elements consisting of "tags" surrounded by angle brackets (like <html>) within the web page

content. HTML tags normally come in pairs like <b> and </b>. The first tag in a pair is the start tag, the second tag is the end tag (they are also called opening tags and closing tags).
The purpose of a web browser is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts in languages such as JavaScript which affect the behavior of HTML webpages. HTML can also be used to include Cascading Style Sheets (CSS) to define the appearance and layout of text and other material such as The W3C, maintainer of both HTML and CSS standards, encourages the use of CSS over explicit presentational markup.

1.10 HISTORY
In 1980, physicist Tim Berners-Lee, who was a contractor at CERN, proposed and prototyped ENQUIRE, a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system.[2] Berners-Lee specified HTML and wrote the browser and server software in the last part of 1990. In that year, Berners-Lee and CERN data systems engineer Robert Cailliau collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes[3] from 1990 he lists[4] "some of the many areas in which hypertext is used" and puts an encyclopedia first.
The first publicly available description of HTML was a document called HTML Tags, first mentioned on the Internet by Berners-Lee in late 1991.[5][6] It describes 20 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by SGMLguid, an in-house SGML based documentation format at CERN. Thirteen of these elements still exist in HTML

Subject Name: Web Technology

Subject Code: IT2353

4.[7]
HTML is a text and image formatting language used by web browsers to dynamically format web pages. Many of the text elements are found in the 1988 ISO technical report TR 9537 Techniques for using SGML, which in turn covers the features of early text formatting languages such as that used by the RUNOFF command developed in the early 1960s for the CTSS (Compatible Time-Sharing System) operating system: these formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with also the separation of structure and processing; HTML has been progressively moved in this direction with CSS.
Berners-Lee considered HTML to be an application of SGML. It was formally defined as such by the Internet Engineering Task Force (IETF) with the mid-1993 publication of the first proposal for an HTML specification: "Hypertext Markup Language (HTML)" Internet-Draft by Berners-Lee and Dan Connolly, which included an SGML Document Type Definition to define the grammar.[8] The draft expired after six months, but was notable for its acknowledgment of the NCSA Mosaic browser's custom tag for embedding in-line images, reflecting the IETF's philosophy of basing standards on successful prototypes.[9] Similarly, Dave Raggett's competing Internet-Draft, "HTML+ (Hypertext Markup Format)", from late 1993, suggested standardizing already-implemented features like tables and fill-out forms.[10]
1.12 Version history of the standard
HTML version timeline
November 24, 1995
HTML 2.0 was published as IETF RFC 1866. Supplemental RFCs added capabilities:
November 25, 1995: RFC 1867 (form-based file upload)
May 1996: RFC 1942 (tables)

Subject Name: Web Technology

Subject Code: IT2353

August 1996: RFC 1980 (client-side image maps)
January 1997: RFC 2070 (internationalization)
In June 2000, all of these were declared obsolete/historic by RFC 2854.
January 1997
HTML 3.2[14] was published as a W3C Recommendation. It was the first version developed and standardized exclusively by the W3C, as the IETF had closed its HTML Working Group in September 1996.
HTML 3.2 dropped math formulas entirely, reconciled overlap among various proprietary extensions and adopted most of Netscape's visual markup tags. Netscape's blink element and Microsoft's marquee element were omitted due to a mutual agreement between the two companies.[13] A markup for mathematical formulas similar to that in HTML wasn't standardized until 14 months later in MathML.
December 1997
HTML 4.0
was published as a W3C Recommendation. It offers three variations:
Strict, in which deprecated elements are forbidden,
Transitional, in which deprecated elements are allowed,
Frameset, in which mostly only frame related elements are allowed;
Initially code-named "Cougar",[17] HTML 4.0 adopted many browser-specific element types and attributes, but at the same time sought to phase out Netscape's visual markup features by marking them as deprecated in favor of style sheets. HTML 4 is an SGML application conforming to ISO 8879 - SGML
April 1998
HTML 4.0[19] was reissued with minor edits without incrementing the version number.
December 1999

Subject Name: Web Technology

Subject Code: IT2353

HTML 4.01[20] was published as a W3C Recommendation. It offers the same three variations as HTML 4.0 and its last errata were published May 12, 2001.
May 2000
ISO/IEC 15445:2000[21][22] ("ISO HTML", based on HTML 4.01 Strict) was published as an ISO/IEC international standard. In the ISO this standard falls in the domain of the ISO/IEC JTC1/SC34 (ISO/IEC Joint Technical Committee 1, Subcommittee 34 - Document description and processing languages).
As of mid-2008, HTML 4.01 and ISO/IEC 15445:2000 are the most recent versions of HTML. Development of the parallel, XML-based language XHTML occupied the W3C's HTML Working Group through the early and mid-2000s.
1.13 Basic XHTML syntax and semantics
XHTML is a separate language that began as a reformulation of HTML 4.01 using XML 1.0. It continues to be developed:
XHTML 1.0, published January 26, 2000 as a W3C Recommendation, later revised and republished August 1, 2002. It offers the same three variations as HTML 4.0 and 4.01, reformulated in XML, with minor restrictions.
XHTML 1.1, published May 31, 2001 as a W3C Recommendation. It is based on XHTML 1.0 Strict, but includes minor changes, can be customized, is reformulated using modules from Modularization of XHTML, which was published April 10, 2001 as a W3C Recommendation.
XHTML 2.0,. There is no XHTML 2.0 standard. XHTML 2.0 is incompatible with XHTML 1.x and, therefore, would be more accurate to characterize as an XHTML-inspired new language than an update to XHTML 1.x.
XHTML 5, which is an update to XHTML 1.x, is being defined alongside HTML 5 in the HTML 5 draft.
Markup

Subject Name: Web Technology

Subject Code: IT2353

HTML markup consists of several key components, including elements (and their attributes), character-based data types, character references and entity references. Another important component is the document type declaration, which specifies the Document Type Definition. As of HTML 5, no Document Type Definition will need to be specified and will only determine the layout mode.[41]

The Hello world program, a common computer program employed for comparing programming languages, scripting languages and markup languages is made of 9 lines of code although in HTML newlines are optional:

```
<!doctype html>
<html>
<head>
<title>Hello HTML</title>
</head>
<body>
<p>Hello World!</p>
</body>
</html>
```

(The text between <html> and </html> describes the web page, and The text between <body> and </body> is the visible page content.)

This Document Type Declaration is for HTML 5. If the <!doctype html> declaration is not included, Windows Internet Explorer will render using "quirks mode".[42]

1.14 SOME FUNDAMENTAL HTML ELEMENTS

HTML Introducion

What is HTML?

HTML is a language for describing web pages.

HTML stands for Hyper Text Markup Language

HTML is not a programming language, it is a markup language

Subject Name: Web Technology

Subject Code: IT2353

A markup language is a set of markup tags

HTML uses markup tags to describe web pages

HTML Tags

HTML markup tags are usually called HTML tags

HTML tags are keywords surrounded by angle brackets like <html>

HTML tags normally come in pairs like <b> and </b>

The first tag in a pair is the start tag, the second tag is the end tag

Start and end tags are also called opening tags and closing tags

HTML Documents = Web Pages

HTML documents describe web pages

HTML documents contain HTML tags and plain text

HTML documents are also called web pages

The purpose of a web browser (like Internet Explorer or Firefox) is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page:

```
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Example Explained

The text between <html> and </html> describes the web page

The text between <body> and </body> is the visible page content

The text between <h1> and </h1> is displayed as a heading

The text between <p> and </p> is displayed as a paragraph

Editing HTML

In this tutorial we use a plain text editor (like Notepad) to edit HTML. We believe this is the best way to learn HTML.However, professional web developers often prefer HTML editors like FrontPage or Dreamweaver, instead of writing plain text.

HTML Elements

HTML documents are defined by HTML elements.

HTML Elements

An HTML element is everything from the start tag to the end tag:

Start tag * Element content End tag *

<p> This is a paragraph </p>

<a href="default.htm" > This is a link </a>

<br />

* The start tag is often called the opening tag. The end tag is often called the closing tag.

HTML Element Syntax

An HTML element starts with a start tag / opening tag

An HTML element ends with an end tag / closing tag

The element content is everything between the start and the end tag

Some HTML elements have empty content

Empty elements are closed in the start tag

Most HTML elements can have attributes

Tip: You will learn about attributes in the next chapter of this tutorial.

Nested HTML Elements

Most HTML elements can be nested (can contain other HTML elements).

HTML documents consist of nested HTML elements.

HTML Document Example

```
<html>
<body>
<p>This is my first paragraph.</p>
</body>
```

</html>

The example above contains 3 HTML elements.

HTML Example Explained

The <p> element:

<p>This is my first paragraph.</p>

The <p> element defines a paragraph in the HTML document.

The element has a start tag <p> and an end tag </p>.

The element content is: This is my first paragraph.

The <body> element:

<body>

<p>This is my first paragraph.</p>

</body>

The <body> element defines the body of the HTML document.

The element has a start tag <body> and an end tag </body>.

The element content is another HTML element (a p element).

The <html> element:

<html>

<body>

<p>This is my first paragraph.</p>

</body>

</html>

The <html> element defines the whole HTML document.

The element has a start tag <html> and an end tag </html>.

The element content is another HTML element (the body element).

Don't Forget the End Tag

Most browsers will display HTML correctly even if you forget the end tag:

<p>This is a paragraph

<p>This is a paragraph

The example above will work in most browsers, but don't rely on it. Forgetting the end tag can produce unexpected results or errors.

Note: Future version of HTML will not allow you to skip end tags.

EMPTY HTML ELEMENTS

HTML elements with no content are called empty elements. Empty elements can be closed in the start tag.

<br> is an empty element without a closing tag (the <br> tag defines a line break).

In XHTML, XML, and future versions of HTML, all elements must be closed.

Adding a slash to the start tag, like <br />, is the proper way of closing empty elements, accepted by HTML, XHTML and XML.

Even if <br> works in all browsers, writing <br /> instead is more future proof.

HTML Tip: Use Lowercase Tags

HTML tags are not case sensitive: <P> means the same as <p>. Many web sites use uppercase HTML tags.

W3Schools use lowercase tags because the World Wide Web Consortium (W3C) recommends lowercase in HTML 4, and demands lowercase tags in future versions of (X)HTML.

HTML Attributes

HTML elements can have attributes

Attributes provide additional information about an element

Attributes are always specified in the start tag

Attributes come in name/value pairs like: name="value"

Attribute Example

HTML links are defined with the <a> tag. The link address is specified in the href attribute:

<a href="http://www.w3schools.com">This is a link</a>

Attribute values should always be enclosed in quotes.

Double style quotes are the most common, but single style quotes are also allowed.

Subject Name: Web Technology

Subject Code: IT2353

Tip: In some rare situations, when the attribute value itself contains quotes, it is necessary to use single quotes: name='John "ShotGun" Nelson'

HTML Tip: Use Lowercase Attributes

Attribute names and attribute values are case-insensitive.

However, the World Wide Web Consortium (W3C) recommends lowercase attributes/attribute values in their HTML 4 recommendation.

Newer versions of (X)HTML will demand lowercase attributes.

HTML Attributes Reference

A complete list of legal attributes for each HTML element is listed in our:

Below is a list of some attributes that are standard for most HTML elements:

Attribute Value Description

class classname Specifies a classname for an element

id id Specifies a unique id for an element

style style_definition Specifies an inline style for an element

title tooltip_text Specifies extra information about an element (displayed as a tool tip)

HTML HEADINGS

HTML headings are defined with the <h1> to <h6> tags.

Example:

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

HTML Paragraphs

HTML paragraphs are defined with the <p> tag.

Example:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

HTML Links

Subject Name: Web Technology

Subject Code: IT2353

HTML links are defined with the <a> tag.

<a href="http://www.w3schools.com">This is a link</a>

Note: The link address is specified in the href attribute.

HTML Images

HTML images are defined with the <img> tag.

Example:

<img src="w3schools.jpg" width="104" height="142" />

HTML documents are divided into paragraphs.

HTML Paragraphs

Paragraphs are defined with the <p> tag.

Example

<p>This is a paragraph</p>

<p>This is another paragraph</p>

Note: Browsers automatically add an empty line before and after a paragraph.

Most browsers will display HTML correctly even if you forget the end tag:

Example

<p>This is a paragraph

<p>This is another paragraph

HTML Line Breaks

Use the <br /> tag if you want a line break (a new line) without starting a new paragraph:

Example

<p>This is<br />a para<br />graph with line breaks</p>

The <br /> element is an empty HTML element. It has no end tag.

<br> or <br />

In XHTML, XML, and future versions of HTML, HTML elements with no end tag (closing tag) are not allowed.

Even if <br> works in all browsers, writing <br /> instead is more future proof.

Subject Name: Web Technology

Subject Code: IT2353

HTML Text Formatting

This text is bold

This text is big

This text is italic

This is computer output

This is subscript and superscript

HTML Text Formatting Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
<em>	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
<strong>	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text

HTML "Computer Output" Tags

Tag	Description
<code>	Defines computer code text
<kbd>	Defines keyboard text
<samp>	Defines sample computer code
<tt>	Defines teletype text
<var>	Defines a variable
<pre>	Defines preformatted text

HTML Citations, Quotations, and Definition Tags

Tag Description

<abbr> Defines an abbreviation

<acronym> Defines an acronym

<address> Defines contact information for the author/owner of a document

<bdo> Defines the text direction

<blockquote> Defines a long quotation

<q> Defines a short quotation

<cite> Defines a citation

<dfn> Defines a definition term

Your browser does not support inline frames or is currently configured not to display inline frames.

HTML <tt> <i> <b> <big> <small> Tags

Example

Format text in a document:

tt>Teletype text</tt>

<i>Italic text</i>

<b>Bold text</b>

<big>Big text</big>

<small>Small text</small>

HTML Formatting Tags

HTML uses tags like <b> and <i> for formatting output, like bold or italic text.

These HTML tags are called formatting tags (look at the bottom of this page for a complete reference).

Often <strong> renders as <b>, and <em> renders as <i>.

However, there is a difference in the meaning of these tags:

<b> or <i> defines bold or italic text only.

<strong> or <em> means that you want the text to be rendered in a way that the user understands as "important". Today, all major browsers render strong as bold and em as italics.

HTML <pre> Tag

Example

Preformatted text:

```
<pre>
Text in a pre element
is displayed in a fixed-width
font, and it preserves
both spaces and
line breaks
</pre>
```

Definition and Usage

The <tt>, <i>, <b>, <big>, and <small> tags are all font-style tags. They are not deprecated, but it is possible to achieve richer effect with CSS.

Tag	Description
<i>	Renders as italic text
<b>	Renders as bold text
<big>	Renders as bigger text
<small>	Renders as smaller text

1.15 RELATIVE URLS

A URL is another word for a web address.

A URL can be composed of words, such as "w3schools.com", or an Internet Protocol (IP) address: 192.68.20.50. Most people enter the name of the website when surfing, because names are easier to remember than numbers.

URL - Uniform Resource Locator

When you click on a link in an HTML page, an underlying <a> tag points to an address on the world wide web.A Uniform Resource Locator (URL) is used to address a document (or other data) on the world wide web.

scheme://host.domain:port/path/filename

Explanation:

scheme - defines the type of Internet service. The most common type is http
host - defines the domain host (the default host for http is www)
domain - defines the Internet domain name, like w3schools.com
:port - defines the port number at the host (the default port number for http is 80)
path - defines a path at the server (If omitted, the document must be stored at the root directory of the web site
filename - defines the name of a document/resource

Common websites start with http://. Pages starting with http:// are not encrypted, so all information exchanged between your computer and the Internet can be "seen" by hackers.

Secure websites start with https://. The "s" stands for "secure". Here, the information exchanged will be encrypted, making it useless to hackers.

Common URL Schemes

The table below lists some common schemes:

Scheme	Short for....	Which pages will the scheme be used for...
http	HyperText Transfer Protocol	Common web pages starts with http://. Not encrypted. Unwise to enter personal information in http:// pages
https	Secure HyperText Transfer Protocol	Secure web pages. All information exchanged are encrypted, cannot be read by hackers
ftp	File Transfer Protocol	For downloading or uploading files to a website. Useful for domain maintenance
file		A file on your computer
gopher		A Gopher document or menu
news		A newsgroup

WAIS Wide Area Information

Search

A database or document on a WAIS database

1.16 HTML 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). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc.

Table 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 the HTML code above looks in a browser:

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

HTML Tables and the Border Attribute

If you do not specify a border attribute, the table will be displayed without borders. Sometimes this can be useful, but most of the time, we want the borders to show.

To display a table with borders, specify the border attribute:

```
<table border="1">
<tr>
```

```
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

HTML Table Headers

Header information in a table are defined with the <th> tag.

The text in a th element will be bold and centered.

```
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<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 the HTML code above looks in a browser:

```
Header 1 Header 2
row 1, cell 1 row 1, cell 2
row 2, cell 1 row 2, cell 2
```

1.17 HTML LISTS

The most common HTML lists are ordered and unordered lists:

HTML Lists

An ordered list:

1. The first list item
2. The second list item
3. The third list item

An unordered list:

List item

List item

List item

HTML Unordered Lists

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

The list items are marked with bullets (typically small black circles).

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

How the HTML code above looks in a browser:

Coffee

Milk

HTML Ordered Lists

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items are marked with numbers.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

How the HTML code above looks in a browser:

1. Coffee
2. Milk

Subject Name: Web Technology

Subject Code: IT2353

HTML Definition Lists

A definition list is a list of items, with a description of each item.

The <dl> tag defines a definition list.

The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list):

```
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>
<dd>- white cold drink</dd>
</dl>
```

How the HTML code above looks in a browser:

Coffee

- black hot drink

Milk

- white cold drink

Basic Notes - Useful Tips

Tip: Inside a list item you can put text, line breaks, images, links, other lists, etc.

HTML List Tags

Tag	Description
<ol>	Defines an ordered list
<ul>	Defines an unordered list
<li>	Defines a list item
<dl>	Defines a definition list
<dt>	Defines an item in a definition list
<dd>	Defines a description of an item in a definition list

1.18 HTML FORMS

HTML forms are used to pass data to a server.

A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

The <form> tag is used to create an HTML form:

```
<form>
.
input elements
.
</form>
```

HTML Forms - The Input Element

The most important form element is the input element. The input element is used to select user information.An input element can vary in many ways, depending on the type attribute. An input element can be of type text field, checkbox, password, radio button, submit button, and more.The most used input types are described below.

Text Fields

<input type="text" /> defines a one-line input field that a user can enter text into:

```
<form>
First name: <input type="text" name="firstname" /><br />
Last name: <input type="text" name="lastname" />
</form>
```

How the HTML code above looks in a browser:

First name:

Last name:

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters.

Password Field

<input type="password" /> defines a password field:

```
<form>
```

Password: <input type="password" name="pwd" />

</form>

How the HTML code above looks in a browser:

Password:

Note: The characters in a password field are masked (shown as asterisks or circles).

Radio Buttons

<input type="radio" /> defines a radio button. Radio buttons let a user select ONLY ONE one of a limited number of choices:

<form>

<input type="radio" name="sex" value="male" /> Male<br />

<input type="radio" name="sex" value="female" /> Female

</form>

How the HTML code above looks in a browser:

Male

Female

Checkboxes

<input type="checkbox" /> defines a checkbox. Checkboxes let a user select ONE or MORE options of a limited number of choices.

<form>

<input type="checkbox" name="vehicle" value="Bike" /> I have a bike<br />

<input type="checkbox" name="vehicle" value="Car" /> I have a car

</form>

How the HTML code above looks in a browser:

I have a bike

I have a car

Submit Button

<input type="submit" /> defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="html_form_action.asp" method="get">
Username: <input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

How the HTML code above looks in a browser:

Username:

Submit ubmit

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "html_form_action.asp". The page will show you the received input.

HTML Form Tags

Tag	Description
<form>	Defines an HTML form for user input
<input />	Defines an input control
<textarea>	Defines a multi-line text input control
<label>	Defines a label for an input element
<fieldset>	Defines a border around elements in a form
<legend>	Defines a caption for a fieldset element
<select>	Defines a select list (drop-down list)
<optgroup>	Defines a group of related options in a select list
<option>	Defines an option in a select list
<button>	Defines a push button

1.19 HTML FRAMES

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 HTML frameset Element

The frameset element holds two or more frame elements. Each frame element holds a separate document.The frameset element states only HOW MANY columns or rows there will be in the frameset.

The HTML frame Element

The <frame> tag defines one particular window (frame) within a frameset.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 document "frame_a.htm" is put into the first column, and the document "frame_b.htm" is put into the second column:

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

Navigation frame

How to make a navigation frame. The navigation frame contains a list of links with the second frame as the target. The file called "tryhtml_contents.htm" contains three links. The source code of the links:

```
<a href ="frame_a.htm" target ="showframe">Frame a</a><br>
<a href ="frame_b.htm" target ="showframe">Frame b</a><br>
<a href ="frame_c.htm" target ="showframe">Frame c</a>
```

The second frame will show the linked document.

Jump to a specified section with frame navigation

Two frames. The navigation frame (content.htm) to the left contains a list of links with the second frame (link.htm) as a target. The second frame shows the linked document. One of the links in the navigation frame is linked to a specified section in the target file. The HTML code in the file "content.htm" looks like this: <a href ="link.htm" target ="showframe">Link without Anchor</a><br><a href ="link.htm#C10" target ="showframe">Link with Anchor</a>.

HTML Frame Tags

Tag	Description
<frameset>	Defines a set of frames
<frame />	Defines a sub window (a frame)
<noframes>	Defines a noframe section for browsers that do not handle frames
<iframe>	Defines an inline sub window (frame)

1.20 XML AND CREATING HTML DOCUMENTS CASE STUDY.

Elements

HTML documents are composed entirely of HTML elements that, in their most general form have three components: a pair of element tags, a "start tag" and "end tag"; some element attributes within the start tag; and finally, any textual and graphical content between the start and end tags. The HTML element is everything between and including the tags. Each tag is enclosed in angle brackets.

The general form of an HTML element is therefore: <tag attribute1="value1" attribute2="value2">content to be rendered</tag> The name of the HTML element is also the name of the tag. Note that the end tag's name is preceded by a slash character, "/". If attributes are not assigned, default values are used.

Element examples

Header of the HTML document:<head>...</head>. Usually the title should be included in the head, for example:

<head>

<title>The title</title>

</head>
Headings: HTML headings are defined with the <h1> to <h6> tags:
<h1>Heading1</h1>
<h2>Heading2</h2>
<h3>Heading3</h3>
<h4>Heading4</h4>
<h5>Heading5</h5>
<h6>Heading6</h6>
Paragraphs:
<p>Paragraph 1</p> <p>Paragraph 2</p>
Line breaks:<br>. The difference between <br> and <p> is that 'br' breaks a line without altering the semantic structure of the page, whereas 'p' sections the page into paragraphs. Note also that 'br' is an empty element in that, while it may have attributes, it can take no content or end tag.
<code><p>This <br> is a paragraph <br> with <br> line
breaks</p></code>
Comments:
<!-- Explanation here -->
Comments can help understanding of the markup and do not display in the webpage.
There are several types of markup elements used in HTML.
Structural markup describes the purpose of text. For example,
<h2>Golf</h2> establishes "Golf" as a second-level heading, which would be rendered in a browser in a manner similar to the "HTML markup" title at the start of this section. Structural markup does not denote any specific rendering, but most web browsers have default styles for element formatting. Text may be further styled with Cascading Style Sheets (CSS).
Presentational markup describes the appearance of the text, regardless of its purpose. For example <b>boldface</b> indicates that visual output devices

should render "boldface" in bold text, but gives little indication what devices which are unable to do this (such as aural devices that read the text aloud) should do. In the case of both <b>bold</b> and <i>italic</i>, there are other elements that may have equivalent visual renderings but which are more semantic in nature, such as <strong>strong emphasis</strong> and <em>emphasis</em> respectively. It is easier to see how an aural user agent should interpret the latter two elements.

Hypertext markup makes parts of a document into links to other documents. An anchor element creates a hyperlink in the document with the href attribute set to the link URL. For example, the HTML markup, <a href="http://en.wikipedia.org/">Wikipedia</a>, will render the word "Wikipedia" as a hyperlink. To render an image as a hyperlink, an 'img' element is inserted as content into the 'a' element. Like 'br', 'img' is an empty element with attributes but no content or closing tag. <a href="http://example.org"><img src="image.gif" alt="descriptive text" width="50" height="50"></a>.

Attributes

Most of the attributes of an element are name-value pairs, separated by "=" and written within the start tag of an element after the element's name. The value may be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML) Leaving attribute values unquoted is considered unsafe. In contrast with name-value pair attributes, there are some attributes that affect the element simply by their presence in the start tag of the element, like the ismap attribute for the img element.[46]

There are several common attributes that may appear in many elements:

The id attribute provides a document-wide unique identifier for an element. This is used to identify the element so that stylesheets can alter its presentational properties, and scripts may alter, animate or delete its contents

or presentation. Appended to the URL of the page, it provides a globally unique identifier for the element, typically a sub-section of the page. For example, the ID "Attributes" in
http://en.wikipedia.org/wiki/HTML#Attributes
The class attribute provides a way of classifying similar elements. This can be used for semantic or presentation purposes. For example, an HTML document might semantically use the designation class="notation" to indicate that all elements with this class value are subordinate to the main text of the document. In presentation, such elements might be gathered together and presented as footnotes on a page instead of appearing in the place where they occur in the HTML source. Class attributes are used semantically in microformats. Multiple class values may be specified; for example class="notation important" puts the element into both the 'notation' and the 'important' classes.
An author may use the style attribute to assign presentational properties to a particular element. It is considered better practice to use an element's id or class attributes to select the element from within a stylesheet, though sometimes this can be too cumbersome for a simple, specific, or ad hoc styling.
The title attribute is used to attach subtextual explanation to an element. In most browsers this attribute is displayed as a tooltip.
The lang attribute identifies the natural language of the element's contents, which may be different from that of the rest of the document. For example, in an English-language document:
<p>Oh well, <span lang="fr">c'est la vie</span>, as they say
in France.</p>
The abbreviation element, abbr, can be used to demonstrate some of these attributes:
<abbr id="anId" class="jargon" style="color:purple;" title="Hypertext

Subject Name: Web Technology

Subject Code: IT2353

Markup Language">HTML</abbr>
This example displays as HTML; in most browsers, pointing the cursor at the abbreviation should display the title text "Hypertext Markup Language."Most elements also take the language-related attribute dir to specify text direction, such as with "rtl" for right-to-left text in, for example, Arabic, Persian or Hebrew.
Character and entity references
See also: List of XML and HTML character entity references
See also: Unicode and HTML
As of version 4.0, HTML defines a set of 252 character entity references and a set of 1,114,050 numeric character references, both of which allow individual characters to be written via simple markup, rather than literally. A literal character and its markup counterpart are considered equivalent and are rendered identically.
The ability to "escape" characters in this way allows for the characters < and & (when written as &lt; and &amp;, respectively) to be interpreted as character data, rather than markup. For example, a literal < normally indicates the start of a tag, and & normally indicates the start of a character entity reference or numeric character reference; writing it as &amp; or &#x26; or &#38; allows & to be included in the content of an element or in the value of an attribute.
Escaping also allows for characters that are not easily typed, or that are not available in the document's character encoding, to be represented within element and attribute content. For example, the acute-accented e (é), a character typically found only on Western European keyboards, can be written in any HTML document as the entity reference &eacute; or as the numeric references &#233; or &#xE9;, using characters that are available on all keyboards and are supported in all character encodings.
Unicode character encodings such as UTF-8 are compatible with all modern browsers and allow direct access to almost all the characters of the world's writing systems.[49]
Data types
HTML defines several data types for element content, such as script data and

stylesheet data, and a plethora of types for attribute values, including IDs, names, URIs, numbers, units of length, languages, media descriptors, colors, character encodings, dates and times, and so on. All of these data types are specializations of character data.

Document type declaration

HTML documents are required to start with a Document Type Declaration (informally, a "doctype"). In browsers, the doctype helps to define the rendering mode—particularly whether to use quirks mode.

The original purpose of the doctype was to enable parsing and validation of HTML documents by SGML tools based on the Document Type Definition (DTD). The DTD to which the DOCTYPE refers contains a machine-readable grammar specifying the permitted and prohibited content for a document conforming to such a DTD. Browsers, on the other hand, do not implement HTML as an application of SGML and by consequence do not read the DTD. HTML 5 does not define a DTD, because of the technology's inherent limitations, so in HTML 5 the doctype declaration, <!doctype html>, does not refer to a DTD.

An example of an HTML 4 doctype is

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

This declaration references the DTD for the 'strict' version of HTML 4.01. SGMLbased validators read the DTD in order to properly parse the document and to perform validation. In modern browsers, a valid doctype activates standards mode as opposed to quirks mode.

In addition, HTML 4.01 provides Transitional and Frameset DTDs, as explained below.

Semantic HTML

Semantic HTML is a way of writing HTML that emphasizes the meaning of the encoded information over its presentation (look). HTML has included semantic

Subject Name: Web Technology

Subject Code: IT2353

markup from its inception,[50] but has also included presentational markup such as <font>, <i> and <center> tags. There are also the semantically neutral span and div tags. Since the late 1990s when Cascading Style Sheets were beginning to work in most browsers, web authors have been encouraged to avoid the use of presentational HTML markup with a view to the separation of presentation and content.
An important type of web agent that does trawl and read web pages automatically, without prior knowledge of what it might find, is the Web crawler or search-engine spider. These software agents are dependent on the semantic clarity of web pages they find as they use various techniques and algorithms to read and index millions of web pages a day and provide web users with search facilities without which the World Wide Web would be only a fraction of its current usefulness.In order for searchengine spiders to be able to rate the significance of pieces of text they find in HTML documents, and also for those creating mashups and other hybrids as well as for more automated agents as they are developed, the semantic structures that exist in HTML need to be widely and uniformly applied to bring out the meaning of published text.[53]
Delivery
HTML documents can be delivered by the same means as any other computer file. However, they are most often delivered either by HTTP from a web server or by email.
HTTP
The World Wide Web is composed primarily of HTML documents transmitted from web servers to web browsers using the Hypertext Transfer Protocol (HTTP). However, HTTP is used to serve images, sound, and other content, in addition to HTML. To allow the Web browser to know how to handle each document it receives, other information is transmitted along with the document. This meta data usually includes the MIME type (e.g. text/html or application/xhtml+xml) and the character encoding (see Character encoding in HTML).In modern browsers, the MIME type

Subject Name: Web Technology

Subject Code: IT2353

that is sent with the HTML document may affect how the document is initially interpreted. A document sent with the XHTML MIME type is expected to be wellformed XML; syntax errors may cause the browser to fail to render it. The same document sent with the HTML MIME type might be displayed successfully, since some browsers are more lenient with HTML.
The W3C recommendations state that XHTML 1.0 documents that follow guidelines set forth in the recommendation's Appendix C may be labeled with either MIME Type.[54] The current XHTML 1.1 Working Draft also states that XHTML 1.1 documents should[55] be labeled with either MIME type.[56]
HTML e-mail
Main article: HTML email
Most graphical email clients allow the use of a subset of HTML (often ill-defined) to provide formatting and semantic markup not available with plain text. This may include typographic information like coloured headings, emphasized and quoted text, inline images and diagrams. Many such clients include both a GUI editor for composing HTML e-mail messages and a rendering engine for displaying them. Use of HTML in e-mail is controversial because of compatibility issues, because it can help disguise phishing attacks, because it can confuse spam filters and because the message size is larger than plain text.
Naming conventions
The most common filename extension for files containing HTML is .html. A common abbreviation of this is .htm, which originated because some early operating systems and file systems, such as DOS and FAT, limited file extensions to three letters.
HTML Application
An HTML Application (HTA; file extension ".hta") is a Microsoft Windows application that uses HTML and Dynamic HTML in a browser to provide the application's graphical interface. A regular HTML file is confined to the security

model of the web browser, communicating only to web servers and manipulating only webpage objects and site cookies. An HTA runs as a fully trusted application and therefore has more privileges, like creation/editing/removal of files and Windows Registry entries. Because they operate outside the browser's security model, HTAs cannot be executed via HTTP, but must be downloaded (just like an EXE file) and executed from local file system.

SGML-based versus XML-based HTML

One difference in the latest HTML specifications lies in the distinction between the SGML-based specification and the XML-based specification. The XML-based specification is usually called XHTML to distinguish it clearly from the more traditional definition. However, the root element name continues to be 'html' even in the XHTML-specified HTML. The W3C intended XHTML 1.0 to be identical to HTML 4.01 except where limitations of XML over the more complex SGML require workarounds. Because XHTML and HTML are closely related, they are sometimes documented in parallel. In such circumstances, some authors conflate the two names as (X)HTML or X(HTML).

Like HTML 4.01, XHTML 1.0 has three sub-specifications: strict, loose and frameset.Aside from the different opening declarations for a document, the differences between an HTML 4.01 and XHTML 1.0 document—in each of the corresponding DTDs—are largely syntactic.

A fix for this is to include a space before closing the tag, as such: <br />.To understand the subtle differences between HTML and XHTML, consider the transformation of a valid and well-formed XHTML 1.0 document that adheres to Appendix C (see below) into a valid HTML 4.01 document. To make this translation requires the following steps:

1. The language for an element should be specified with a lang attribute rather than the XHTML xml:lang attribute. XHTML uses XML's built in language-defining functionality attribute.

2. Remove the XML namespace (xmlns=URI). HTML has no facilities for namespaces.
3. Change the document type declaration from XHTML 1.0 to HTML 4.01. (see DTD section for further explanation).
4. If present, remove the XML declaration. (Typically this is: <?xml version="1.0" encoding="utf-8"?>).
5. Ensure that the document's MIME type is set to text/html. For both HTML and XHTML, this comes from the HTTP Content-Type header sent by the server.
6. Change the XML empty-element syntax to an HTML style empty element (<br/> to <br>).

Those are the main changes necessary to translate a document from XHTML 1.0 to HTML 4.01. To translate from HTML to XHTML would also require the addition of any omitted opening or closing tags. Whether coding in HTML or XHTML it may just be best to always include the optional tags within an HTML document rather than remembering which tags can be omitted.

The W3C recommends several conventions to ensure an easy migration between HTML and XHTML (see HTML Compatibility Guidelines). The following steps can be applied to XHTML 1.0 documents only:

Include both xml:lang and lang attributes on any elements assigning language.

Use the empty-element syntax only for elements specified as empty in HTML.

Include an extra space in empty-element tags: for example <br /> instead of <br/>.

Include explicit close tags for elements that permit content but are left empty (for example, <div></div>, not <div />).

Omit the XML declaration.