JEPPIAAR INSTITUTE OF TECHNOLOGY

“Self-Belief | Self Discipline | Self Respect”

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

LECTURE NOTES

CS8651-INTERNET PROGRAMMING

(Regulation 2017)

Year/Semester: III / 06 CSE

2020 – 2021

Prepared by

Dr. K. Tamilarasi

Associate Professor /CSE

UNIT I WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0

Web Essentials: Clients, Servers and Communication – The Internet – Basic Internet protocols – World wide web – HTTP Request Message – HTTP Response Message – Web Clients – Web Servers – HTML5 – Tables – Lists – Image – HTML5 control elements – Semantic elements – Drag and Drop – Audio – Video controls - CSS3 – Inline, embedded and external style sheets – Rule cascading – Inheritance – Backgrounds – Border Images – Colors – Shadows – Text – Transformations – Transitions – Animations.

1. WEB ESSENTIALS:CLIENT,SERVERS AND COMMUNICATON

1.1 THE INTERNET:

- The Internet traces its roots to a project of the U.S. Department of Defense’s then named Advanced Research Projects Agency, or ARPA. The ARPANETprojectwas intended to support DoD research on computer networking.
- As this project began in the late 1960s, there had been only a few small experimental networks providing communication between geographically dispersed computers from different manufacturers running different operating systems.
- The purpose of ARPANET was to create a larger such network, both in order to electronically connect DoD-sponsored researchers and in order to experiment with and develop tools for heterogeneous computer networking.
- The ARPANET computer network was launched in 1969 and by year’s end consisted of four computers at four sites running four different operating systems.
- **For example**, e-mail was available on ARPANET beginning in 1972, and it soon became an extremely popular application for those who had ARPANET access.
- The regional U.S. networks were often cooperative efforts between universities.
- As one example, SURAnet (Southeastern University Research Association Network)was organized by the University of Maryland beginning in 1982 and eventually included essentially all of the major universities and research institutions in the southeastern United States.
- Another of these networks,CSNET (Computer Science Network), was partially funded by the U.S. National Science Foundation (NSF) to aid scientists at universities without ARPANET access, laying the groundwork for future network developments that we’ll say more about in a moment.
- Several of the most widely used Internet protocols—including the File Transfer Protocol (FTP) and Simple Mail Transfer Protocol (SMTP),which underlie many of the Internet’s file transfer and e-mail operations, respectively—were initially developed under ARPANET.
- ARPANET switched from using an earlier protocol to TCP/IP during 1982. At around the same time, an ARPA Internet was created, allowing computers on some outside networks such as CSNET to communicate via TCP/IP with computers on the ARPANET.
- One of the primary goals of this network was to connect the NSF’s new regional supercomputing centers. But it was also decided that regional networks should be able to connect to NSFNET, so that the NSFNET would provide a backbone through which other networks could interconnect synchronously.

- NSFNET quickly supplanted ARPANET, which was officially decommissioned in 1990. At this point, NSFNET was at the center of the Internet, that is, the collection of computer networks connected via the public backbone and communicating across networks using TCP/IP.
- One of the arguments for allowing commercial traffic was economic: commercial traffic would increase network usage, leading to reduced unit costs through economies of scale.
- This in turn would provide a less expensive network for research and educational purposes.

1.2 BASIC INTERNET PROTOCOLS:

A computer communication protocol is a detailed specification of how communication between two computers will be carried out in order to serve some purpose.

1.2.1 TCP/IP

- The reason that they are often treated as one is that the bulk of the services we associate with the Internet—e-mail, Web browsing, file downloads, accessing remote databases—are built on top of both the TCP and IP protocols.
- But in reality, only one of these protocols—IP, the Internet Protocol—is fundamental to the definition of the Internet.
- A key element of IP is the IP address, which is simply a 32-bit number.
- IP addresses are normally written as a sequence of four decimal numbers separated by periods (called “dots”), as in 192.0.34.166.
- Each decimal number represents one byte of the IP address. The function of IP software is to transfer data from one computer (the source) to another computer (the destination).
- The IP software running on the source creates a packet, which is a sequence of bits representing the data to be transferred along with the source and destination IP addresses and some other header information, such as the length of the data.
- If the destination computer is on the same local network as the source, then the IP software will send the packet to the destination directly via this network.
- If the destination is on another network, the IP software will send the packet to a gateway, which is a device that is connected to the source computer’s network as well as to at least one other network.
- The gateway will select a computer on one of the other networks to which it is attached and send the packet on to that computer.
- This process will continue, with the packet going through perhaps a dozen or more hops, until the packet reaches the destination computer.
- IP software on that computer will receive the packet and pass its data up to an application that is waiting for the data.
- The sequence of computers that a packet travels through from source to destination is known as its route. How does each computer choose the next computer in the route for a packet?
- A separate protocol (the current standard is BGP-4, the Border Gateway Protocol) is used to pass network connectivity information between gateways so that each can choose a good next hop for each packet it receives.

- IP software also adds some error detection information (a checksum) to each packet it creates, so that if a packet is corrupted during transmission, this can usually be detected by the recipient. The IP standard calls for IP software to simply discard any corrupted packets.
- TCP, the Transmission Control Protocol, is a higher-level protocol that extends IP to provide additional functionality, including reliable communication based on the concept of a connection.
- A and B can both send messages to one another at the same time; this is known as full duplex communication. When A and B are both done sending messages to one another (or at least done for the time being), a similar set of three messages is used to close the connection.
- Once a connection has been established, TCP provides reliable data transmission by demanding an acknowledgment for each packet it sends via IP.
- Essentially, the software sets a timer after sending each packet.
- The TCP software on the receiving side sends a packet containing an acknowledgment for every TCP-based packet it receives that passes the checksum test.
- If theTCPsoftware sending a packet does not receive an acknowledgment packet before its timer expires, then it resends the packet and restarts the timer.
- Another important feature that TCP adds to IP is the concept of a port.
- The port concept allows TCP to be used to communicate with many different applications on a machine.

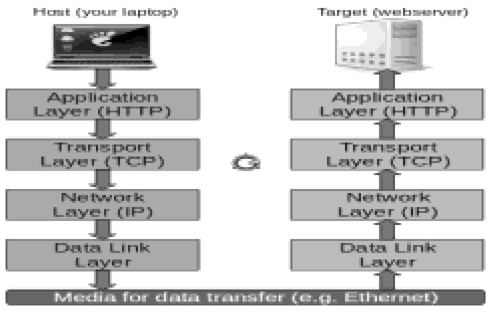


Fig 1.1:Simplified view of communication using TCP/IP

1.2.2 UDP, DNS, and Domain Names

- UDP (User Datagram Protocol) is an alternative protocol to TCP that also builds on IP.
- The main feature that UDP adds to IP is the port concept that we have just seen in TCP.
- However, it does not provide the two-way connection or guaranteed delivery of TCP. Its advantage over TCP is speed for simple tasks.
- One Internet application that is often run using UDP rather than TCP is the Domain Name Service (DNS). DNS provides a mechanism for mapping back and forth between IP addresses and host names.
- IP address—it uses the UDP software running on its system to send a UDP message to one of these DNS servers, requesting the IP address.
- If all goes well, this server will then send back a UDP message containing the IP address. Recall that it took three messages just to get a TCP connection set up, so the UDP approach is much more efficient for sporadic DNS queries. (UDP is

sometimes referred to as a lightweight communication protocol and TCP as a heavyweight protocol, at least in comparison withUDP.

- In general, the terms lightweight and heavyweight in computer science are used to describe alternative software solutions to some problem, with the lightweight solution having less functionality but also less overhead.)
- Internet host names consist of a sequence of labels separated by dots. The final label in a host name is a top-level domain. There are two standard types of top-level domain: generic (such as .com, .edu, .org, and .biz) and country-code (such as .de, .il, and .mx).
- Each top-level domain is divided into subdomains (second-level domains), which may in turn be further divided, and so on. The assignment of second-level domains within each top-level domain is performed (for a fee) by a registry operator selected by ICANN.
- Such a subdomain, consisting of a local host name followed by a domain name (typically consisting of at least two labels) is sometimes called a fully qualified domain name for the computer.
- For example, www.example.org is a fully qualified domain name for a host with local name www that belongs to the example second-level domain of the org top-level domain.
- Typical usage of nslookup is illustrated by the following (user input is italicized):

```
C:\>nslookup www.example.org
Server: slave9.dns.stargate.net
Address: 209.166.161.121

Name: www.example.org
Address: 192.0.34.166

C:\>nslookup 192.0.34.166
Server: slave9.dns.stargate.net
Address: 209.166.161.121

Name: www.example.com
Address: 192.0.34.166
```

- The first two lines following the command line identify the qualified name and IP address of the DNS server that is providing the domain name information that follows. Also notice that a single IP address can be associated with multiple domain names. In this example, both www.example.org and www.example.com are associated with the IP address 192.0.34.166.
- A lookup that specifies an IP address, such as the second lookup in the example, is sometimes referred to as a reverse lookup. As shown, even if multiple qualified names are associated with an IP address, only one of the names will be returned by a reverse lookup. This is known as the canonical name for the host; all other names are considered aliases.
- The reverse lookup in the example indicates that www. exampl e. com is the canonical name for the host with IP address 192.0.34.166.

1.2.3 Higher-Level Protocols

In the cases of both TCP and a phone call, different protocols can be used to communicate once a connection has been established. Similarly, a variety of higher-level protocols are used to communicate once a TCP connection has been established.

- SMTP supports transfer of e-mail between different e-mail servers, while FTP is used for transferring files befween machines. Another higher-level TCP protocol, Telnet, is used to execute commands typed into one computer on a remote computer.
- Telnet can also be used to communicate directly (via keyboard entries) with some TCP-based applications. As described earlier, which protocol will be used to communicate over a TCP connection is normally determined by the port number used to establish the connection.
- The primary TCP-based protocol used for communication between web servers and browsers is called the Hypertext Transport Protocol (HTTP).

1.3 THE WORLD WIDE WEB

- The Usenet newsgroup service began in 1979 and provided a means of “posting” Information.
- Large files were (and still are) often shared by running an FTP server application that allowed any user to transfer the files from their origin machine to the user’s machine.
- The first Internet chat software in widespread use, Internet Relay Chat (IRC), provided both public and private chat facilities.
- Some of the more popular information management technologies in the early 1990s were Gopher information servers, which provided a simple hierarchical view of documents; the Wide Area Information System (WAIS) system for indexing and retrieving information; and the ARCHIE tool for searching online information archives accessible via FTP.
- The server and client applications communicate over the Internet by following a communication protocol built on top of TCP/IP.
- The protocol used by the Web, as just noted, is the Hypertext Transport Protocol, HTTP. As we will learn in the next section, this is a rather generic protocol that for the most part supports a client requesting a document from a server and the server returning the requested document.
- This generic nature of HTTP gives it the advantage of somewhat more flexibility than is present in the protocols used by WAIS and Gopher.
- Most web pages are written using the Hypertext Markup Language, HTML, which along with HTTP is a fundamental web technology. HTML pages can contain the familiar web links (technically called hyperlinks) to other documents on the Web.
- In addition to hyperlinks, modern versions of HTML also provide extensive page layout facilities, including support for inline graphics, which (as you might guess) has added signihcantly to the commercial appeal of the Web.

1.3.1 Hypertext Transport Protocol

- HTTP is a form of communication protocol, in particular a detailed specification of how web clients and servers should communicate.
- The basic structure of HTTP communication follows what is known as a request-response model. Specifically, the protocol dictates that an HTTP interaction is

initiated by a client sending a request message to the server; the server is then expected to generate a response message.

- The format of the request and response messages is dictated by HTTP.
- HTTP does not dictate the network protocol to be used to send these messages, but does expect that the request and response are both sent within a TCP-style connection between the client and the server.
- So most HTTP implementations send these messages using TCP.
- When I pressed the Enter key after typing this address, the browser created a message conforming to the HTTP protocol, used DNS to obtain an IP address for www.examp1e.org, created a TCP connection with the machine at the IP address obtained, sent the HTTP message over this TCP connection, and received back a message containing the information that is shown displayed in the client area of the browser.
- A nice feature of HTTP is that these request and response messages often consist entirely of plain text in a fairly readable form.
- An HTTP request message consists of a start line followed by a message header and optionally a message body.
- The request consists of the three lines beginning with the GET and ending with a blank line, and user input is again italicized):
- The response message in this case begins with the line which is known as the status line of the response, and continues to the end of the example.

HTTP/1.1 200 OK

- The portion of the response between the status line and the first blank line following it is the header of the response.

1.4 HTTP REQUEST MESSAGE

1.4.1 Overall Structure

- Every HTTP request message has the same basic structure:
  - ➢ Start line
  - ➢ Header field(s) (one or more)
  - ➢ Blank line
  - ➢ Message body (optional)
- Every start line consists of three parts, with a single space used to separate adjacent parts:
  - ➢ Request method
  - ➢ Request-URI portion of web address
  - ➢ HTTP version

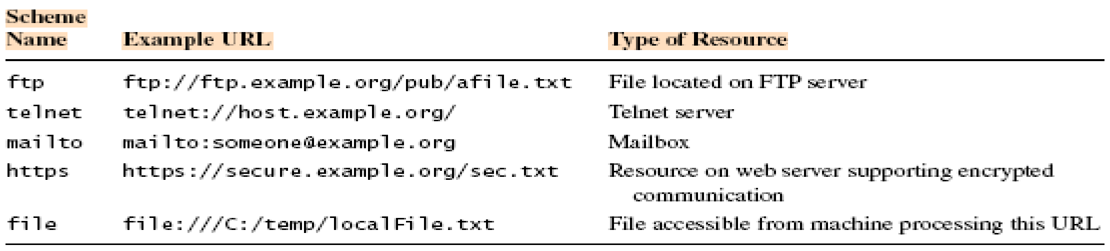
1.4.2 HTTP Version

The initial version of HTTP was referred to as HTTP/0.9, and the first Internet RFC regarding .HTTP described HTTP/1.0. In 1997, HTTP/1.1 was formally defined, and is currently an Internet Draft Standard [RFC-2616].

1.4.3 Request-URI

- The second part of the start line is known as the Request-URI. The concatenation of the string http://, the value of the Host header field (www.example.org, in this example), and the Request-URI (/ in this example) forms a string known as a Uniform Resource Identifier (URI).

- A URI is an identifier that is intended to be associated with a particular resource (such as a web page or graphics image) on the World Wide Web. Every URI consists of two parts: the scheme, which appears before the colon (:), and another part that depends on the scheme.
- Web addresses, for the most part, use the http scheme (the scheme name in URIs is case insensitive, but is generally written in lowercase letters). In this scheme, the URI represents the location of a resource on the Web. A URI of this type is said to be a Uniform Resource Locator (URL).



Scheme Name	Example URL	Type of Resource
ftp	ftp://ftp.example.org/pub/afile.txt	File located on FTP server
telnet	telnet://host.example.org/	Telnet server
mailto	mailto:someone@example.org	Mailbox
https	https://secure.example.org/sec.txt	Resource on web server supporting encrypted communication
file	file:///C:/temp/localFile.txt	File accessible from machine processing this URL

TABLE 1.1 Some Non-http URL Schemes

- In addition to the URL type of URI, there is one other type, called a Uniform Resource Name (URN).
- The URI for a URN always consists of three colon-separated parts, as illustrated here.
- The first part is the scheme name, which is always u rn for a URN-type URI.
- The second part is the namespace identifier, which in this example is ISBN. Other currently registered URN namespace identifiers along with pointers to documentation for each are listed at [IANA- URNS].
- The third part is the namespace-specific string. The exact format and meaning of this string varies with the namespace.

1.4.4 Request Method

- The HTTP/1.1 standard defines a CONNECT method, which can be used to create certain types of secure connections.
- The primary HTTP method is GET. This is the method used when you type a URL into the Location bar of your browser.
- It is also the method that is used by default when you click on a link in a document displayed in your browser and when the browser downloads images for display within an HTML document.
- The POST method is typically used to send information collected from a form displayed within a browser, such as an order-entry form, back to the web server.

Method	Requests server to . . .
GET	return the resource specified by the Request-URI as the body of a response message.
POST	pass the body of this request message on as data to be processed by the resource specified by the Request-URI.
HEAD	return the same HTTP header fields that would be returned if a GET method were used, but not return the message body that would be returned to a GET (this provides information about a resource without the communication overhead of transmitting the body of the response, which may be quite large).
OPTIONS	return (in Allow header field) a list of HTTP methods that may be used to access the resource specified by the Request-URI.
PUT	store the body of this message on the server and assign the specified Request-URI to the data stored so that future GET request messages containing this Request-URI will receive this data in their response messages.
DELETE	respond to future HTTP request messages that contain the specified Request-URI with a response indicating that there is no resource associated with this Request-URI.
TRACE	return a copy of the complete HTTP request message, including start line, header fields, and body, received by the server. Used primarily for test purposes.

Table 1.2 Standard HTTP/1.1 Methods

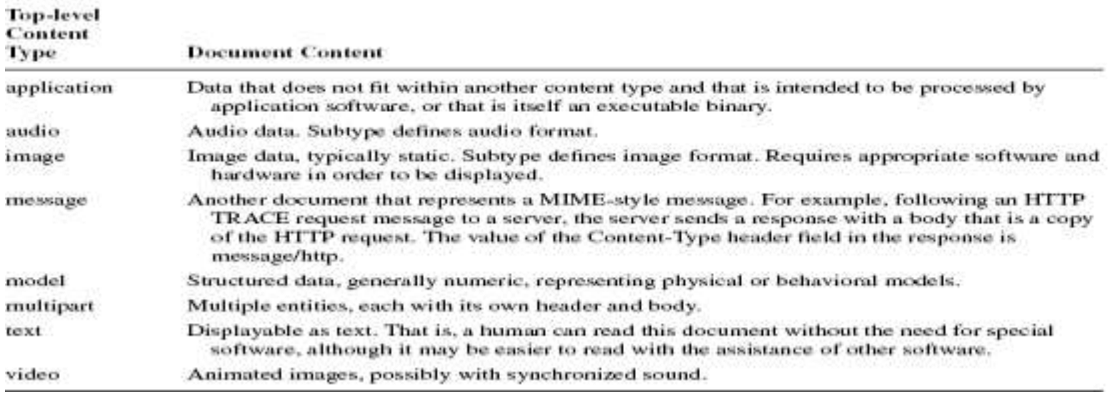
1.4.5 Header Fields and MIME Types

- The Host header field is required in every HTTP/1.1 request message.
- HTTP/1.1 also defines a number of other header fields, several of which are commonly used by modern browsers. Each header field begins with a field name, such as Host, followed by a colon and then a field value.
- The following slightly modified example of an actual HTTP request sent by a browser consists of a start line, 10 header fields, and a short message body:
  host: www.example.org:56789

user-agent: Mozilla/S.O (Windows; U; Windows NT 5.1; en-US; rv:1.4)
Gecko/20030624
accept: text/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,
image/gif;q=0.2,/*;q=0.1*
accept-language: en-us,en;q=0.5
accept-encoding: gzip,deflate
accept-charset: ISO-8859-1,utf-8;q=0.7,;q=0.7*
connection: keep-alive
keep-alive: 300
content-type: application/x-www-form-urlencoded
content-length: 13

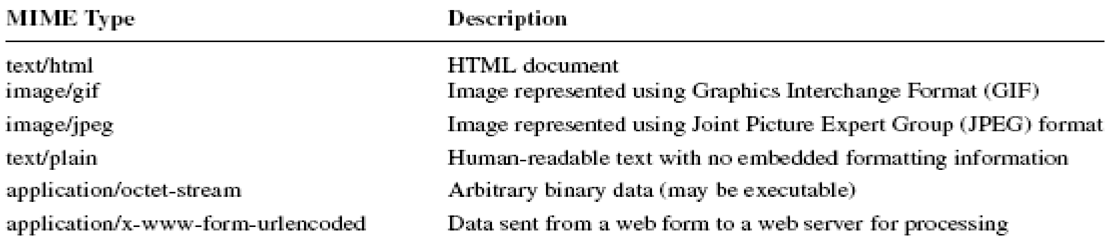
- First, header names are not case sensitive, although I will throughout this text refer to header field names following the capitalization used by the HTTP/1.1 reference [RFC-2616]. So, while the browser used “host” to name the first header Field.
- Second, a header field value may wrap onto several lines by preceding each continuation line with one or more spaces or tabs, as shown for the User-Agent and Accept fields of the preceding example. This also means that a header field name must begin at the first character of a line, with no preceding white space.

- A third common feature is the use of so-called MIME types in several header field
- values. MIME is an acronym standing for Multipurpose Internet Mail Extensions, and refers to a standard that can be used to pass a variety of types of information, including graphics and applications, through e-mail as well as through other Internet message protocols.



Top-level Content Type	Document Content
application	Data that does not fit within another content type and that is intended to be processed by application software, or that is itself an executable binary.
audio	Audio data. Subtype defines audio format.
image	Image data, typically static. Subtype defines image format. Requires appropriate software and hardware in order to be displayed.
message	Another document that represents a MIME-style message. For example, following an HTTP TRACE request message to a server, the server sends a response with a body that is a copy of the HTTP request. The value of the Content-Type header field in the response is message/http.
model	Structured data, generally numeric, representing physical or behavioral models.
multipart	Multiple entities, each with its own header and body.
text	Displayable as text. That is, a human can read this document without the need for special software, although it may be easier to read with the assistance of other software.
video	Animated images, possibly with synchronized sound.

Table 1.3 Standard Top-level MIME Content Types



MIME Type	Description
text/html	HTML document
image/gif	Image represented using Graphics Interchange Format (GIF)
image/jpeg	Image represented using Joint Picture Expert Group (JPEG) format
text/plain	Human-readable text with no embedded formatting information
application/octet-stream	Arbitrary binary data (may be executable)
application/x-www-form-urlencoded	Data sent from a web form to a web server for processing

Table 1.4 Some Common MIME Content Types

1.5 HTTP RESPONSE MESSAGE

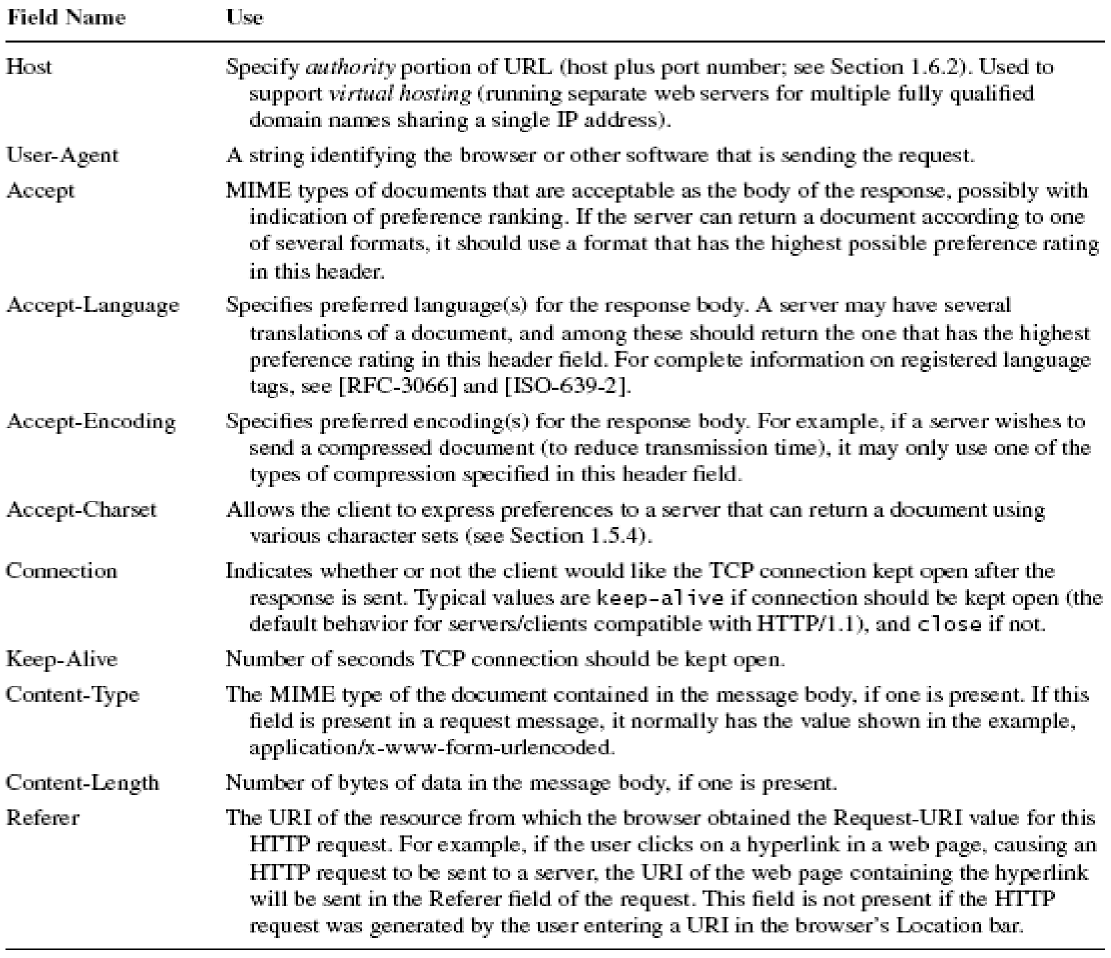
- An HTTP response message consists of a status line, header fields, and the body of the response, in the following format:

Status line
Header field(s) (one or more)
Blank line
Message body (optional)

1.5.1 Response Status Line

The example status line shown earlier was

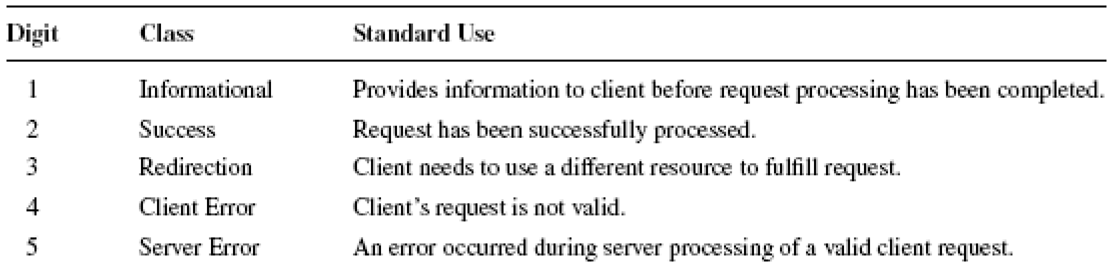
HTTP/1.1 200 OK



Field Name	Use
Host	Specify *authority* portion of URL (host plus port number; see Section 1.6.2). Used to support *virtual hosting* (running separate web servers for multiple fully qualified domain names sharing a single IP address).
User-Agent	A string identifying the browser or other software that is sending the request.
Accept	MIME types of documents that are acceptable as the body of the response, possibly with indication of preference ranking. If the server can return a document according to one of several formats, it should use a format that has the highest possible preference rating in this header.
Accept-Language	Specifies preferred language(s) for the response body. A server may have several translations of a document, and among these should return the one that has the highest preference rating in this header field. For complete information on registered language tags, see [RFC-3066] and [ISO-639-2].
Accept-Encoding	Specifies preferred encoding(s) for the response body. For example, if a server wishes to send a compressed document (to reduce transmission time), it may only use one of the types of compression specified in this header field.
Accept-Charset	Allows the client to express preferences to a server that can return a document using various character sets (see Section 1.5.4).
Connection	Indicates whether or not the client would like the TCP connection kept open after the response is sent. Typical values are `keep-alive` if connection should be kept open (the default behavior for servers/clients compatible with HTTP/1.1), and `close` if not.
Keep-Alive	Number of seconds TCP connection should be kept open.
Content-Type	The MIME type of the document contained in the message body, if one is present. If this field is present in a request message, it normally has the value shown in the example, application/x-www-form-urlencoded.
Content-Length	Number of bytes of data in the message body, if one is present.
Referer	The URI of the resource from which the browser obtained the Request-URI value for this HTTP request. For example, if the user clicks on a hyperlink in a web page, causing an HTTP request to be sent to a server, the URI of the web page containing the hyperlink will be sent in the Referer field of the request. This field is not present if the HTTP request was generated by the user entering a URI in the browser's Location bar.

Table 1.5 Some Common HTTP/1.1 Request Header Fields

- The status line consists of three fields: the HTTP version used by the server software when formatting the response; a numeric status code indicating the type of response; and a text string (the reason phrase) that presents the information represented by the numeric status code in human-readable form.
- The first digit represents the general class of status code. The five classes of HTTP/1.1 status codes are given in Table 1.6. The last two digits of a status code define the specific status within the specified class.



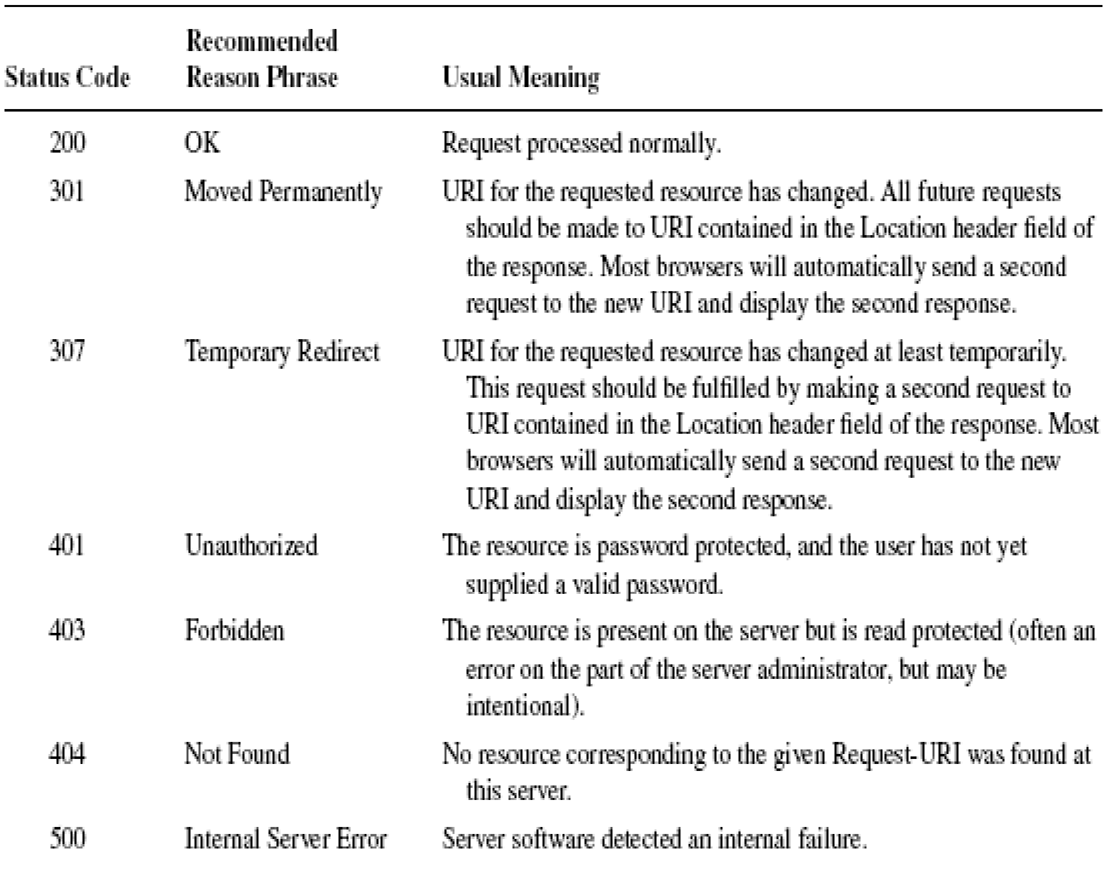
Digit	Class	Standard Use
1	Informational	Provides information to client before request processing has been completed.
2	Success	Request has been successfully processed.
3	Redirection	Client needs to use a different resource to fulfill request.
4	Client Error	Client's request is not valid.
5	Server Error	An error occurred during server processing of a valid client request.

Table 1.6 HTTP/1.1 Status Code Classes (First Digit of Status Code)

1.5.2 Response Header Fields

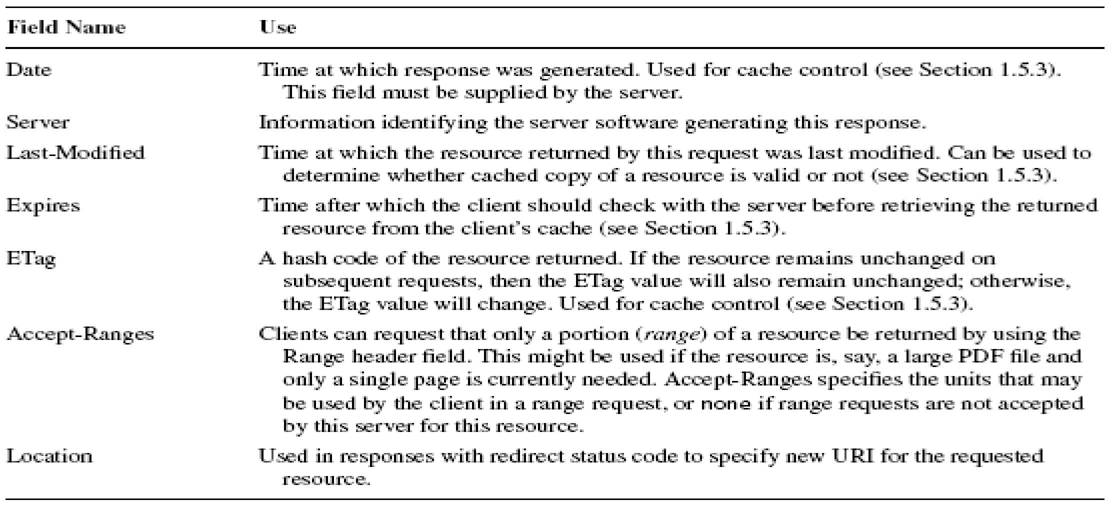
- Some of the header fields used in HTTP request messages, including Connection, Content-Type, and Content-Length, are also valid in response messages. The

Content-Type of a response can be any one of the MIME type values specified by the Accept header field of the corresponding request. Some other common response header fields are shown in Table 1.8.



Status Code	Recommended Reason Phrase	Usual Meaning
200	OK	Request processed normally.
301	Moved Permanently	URI for the requested resource has changed. All future requests should be made to URI contained in the Location header field of the response. Most browsers will automatically send a second request to the new URI and display the second response.
307	Temporary Redirect	URI for the requested resource has changed at least temporarily. This request should be fulfilled by making a second request to URI contained in the Location header field of the response. Most browsers will automatically send a second request to the new URI and display the second response.
401	Unauthorized	The resource is password protected, and the user has not yet supplied a valid password.
403	Forbidden	The resource is present on the server but is read protected (often an error on the part of the server administrator, but may be intentional).
404	Not Found	No resource corresponding to the given Request-URI was found at this server.
500	Internal Server Error	Server software detected an internal failure.

Table 1.7 Some Common HTTP/1.1 Status Codes



Field Name	Use
Date	Time at which response was generated. Used for cache control (see Section 1.5.3). This field must be supplied by the server.
Server	Information identifying the server software generating this response.
Last-Modified	Time at which the resource returned by this request was last modified. Can be used to determine whether cached copy of a resource is valid or not (see Section 1.5.3).
Expires	Time after which the client should check with the server before retrieving the returned resource from the client's cache (see Section 1.5.3).
ETag	A hash code of the resource returned. If the resource remains unchanged on subsequent requests, then the ETag value will also remain unchanged; otherwise, the ETag value will change. Used for cache control (see Section 1.5.3).
Accept-Ranges	Clients can request that only a portion (*range*) of a resource be returned by using the Range header field. This might be used if the resource is, say, a large PDF file and only a single page is currently needed. Accept-Ranges specifies the units that may be used by the client in a range request, or none if range requests are not accepted by this server for this resource.
Location	Used in responses with redirect status code to specify new URI for the requested resource.

Table 1.8 Some Common HTTP/1.1 Response Header Fields

1.5.3 Cache Control

- In computer systems, a cache is a repository for copies of information that originates elsewhere. A copy of information is placed in a cache in order to improve system performance.

- Most web browsers automatically cache on the client machine many of the resources that they request from servers via HTTP.
- However, there is a key drawback to using a cache: information in a cache can become invalid.
- One approach to guaranteeing that a cached copy of a resource is valid is for the client to ask the server whether or not the client's copy is valid. This can be done with relatively little communication by sending an HTTP request for the resource using the HEAD method, which returns only the status line and header portion of the response.
- If the response message contains a Last-Modified time, and this time precedes the value of the Date header field returned with the cached resource, then the cached copy is still valid and can be used. Otherwise, the cached copy is invalid and the browser should send a normal GET request for the resource.

1.5.4 Character Sets

- Character set defines the mapping between these integers, or code points, and characters.
- A character encoding is a bit string that must be decoded into a code-point integer that is then mapped to a character according to the definition provided by some character set. A character encoding often represents characters using variable-length bit strings, with common characters represented using shorter strings and less-common characters using longer strings.
- The Accept-Charset header field is used by a client to tell a server the character sets and character encodings that it will accept as well as its preferred character sets or encodings, if more than one is available for the requested document. In our earlier example, the header field

accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

said that the client would prefer to receive documents using the ISO-8859-1 character set or the UTF-8 encoding of the characters in Unicode, but that it would also accept any other valid Internet character set/encoding.

- A web server can inform a client about the character set/encoding used in a returned document by adding a charset parameter to the value of the Content-Type header field.
- For example, the following Content-Type header field in an HTTP response would indicate that the body of the message is an HTML document written using the UTF-8 character encoding:Content-Type: text/html; charset=UTF-8
- The US-ASCII character set is a subset of both the ISO-8859-1 character set and the UTF-8 character encodings, so the charset parameter is set to one of these two values for many US-ASCII documents in order to ensure international compatibility.

1.6 WEB CLIENTS

- A web client is software that accesses a web server by sending an HTTP request message and processing the resulting HTTP response.
- In general, any web client that is designed to directly support user access to web servers is known as a user agent. Furthermore, some web clients are not designed to be used directly by humans at all. For example, software robots are often used to automatically crawl the Web and download information for use by search engines

1.6.1 Basic Browser Functions

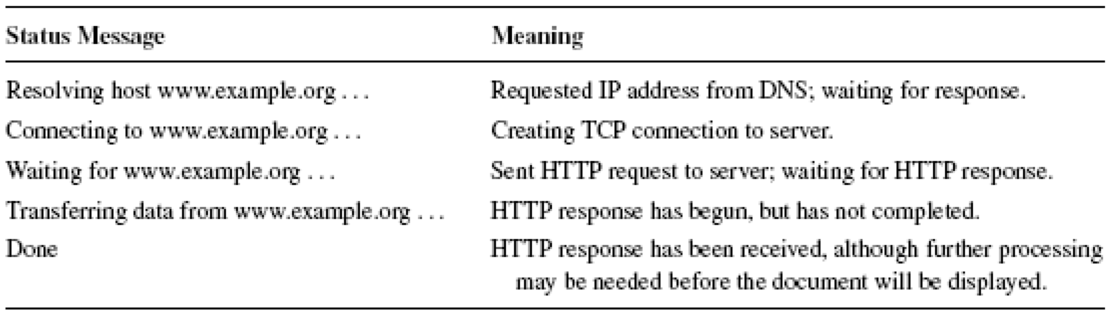
- The window of a typical modern browser is split into several rectangular regions, most of which are known as bars. Figure 1.4 shows five standard regions in a Mozilla 1.4 window.
- The primary region is the client area, which displays a document. For many documents,the title bar displays a title assigned by the document author to the document currently displayed within the client area. The title bar also displays the browser name as well as standard window-management controls. The menu bar contains a set of dropdown menus, much like most other applications that incorporate a graphical user interface (GUI).
- The browser’s Navigation toolbar contains standard push-button controls that allow the user to return to a previously viewed web page (Back), reverse the effect of pressing Back (Forward), ask the server for an updated version of the page currently viewed (Reload), halt page downloading currently in progress (Stop), and print the client area of the window (Print). Clicking the small down-arrow to the right of some buttons produces a menu allowing users to override the default behavior of the associated button.
- The Navigation toolbar also contains a text box, known as the Location bar, where a user can enter a URL and press the Enter key in order to request the browser to display the document located at the specified URL.
- Clicking the Search button instead of pressing Enter causes the information entered in the text box to be sent to a search engine. Clicking the down-arrow at the right side of the Location bar produces a dropdown menu of recently visited URLs that can be visited again with a single click. Finally, the status bar displays messages and icons related to the status of the browser.
- A primary task of any browser is to make HTTP requests on behalf of the browser user.
- If a user types an http-scheme URL in Mozilla’s Location bar, for example, the browser must perform a number of tasks:
    - ➢ Reformat the URL entered as a valid HTTP request message.
    - ➢ If the server is specified using a host name (rather than an IP address), use DNS to convert this name to the appropriate IP address.
    - ➢ Establish a TCP connection using the IP address of the specified web server.
    - ➢ Send the HTTP request over the TCP connection and wait for the server’s response.
    - ➢ Display the document contained in the response. If the document is not a plain-text document but instead is written in a language such as HTML, this involves rendering the document: positioning text and graphics appropriately within the browser window, creating table borders, using appropriate fonts and colors, etc.

1.6.2 URLs

- An http-scheme URL consists of a number of pieces. In order to showthe main possibilities, let’s consider the following example URL:

  http://www.example.org:56789/a/b/c.txt?t=win&s=chess#para5
- The portion of an http URL following the :// string and before the next slash (/) (or through the completion of the URL, if there is no trailing slash) is known as the authority of the URL.

- It consists of either a fully qualified domain name (or other name that can be resolved to an IP address, such as an unqualified name of a machine on the local network) or an IP.



Status Message	Meaning
Resolving host www.example.org . . .	Requested IP address from DNS; waiting for response.
Connecting to www.example.org . . .	Creating TCP connection to server.
Waiting for www.example.org . . .	Sent HTTP request to server; waiting for HTTP response.
Transferring data from www.example.org . . .	HTTP response has begun, but has not completed.
Done	HTTP response has been received, although further processing may be needed before the document will be displayed.

Table 1.9 Some Mozilla Status Messages

- if the port number is omitted, then a TCP connection to port 80 is implied. In this example, the authority is www.example.org:56789 and consists of the fully qualified domain name www.example.org followed by the port number 56789.
- The portion from the slash following the authority through the question mark (?) (or through the end of the URL, if there is no question mark) is called the path of the URL.
- The leading slash is part of the path, but the question mark is not. So the path in the example

URL just given is /a/b/c.txt.

- The Following the path there may be a question mark followed by information up to a number sign (#). The information between but not including the question mark and number sign is the query portion of the URL, and in general a string of the form shown is known as a query string.
- The query portion of the example URL is t=win&s=chess. Originally, the query portion of a URL was intended to pass search terms to a web server.
- So in this example, it might be that the user is seeking a resource with a title containing the string "win" that is related to the subject "chess."
- A browser forms the Request-URI portion of an HTTP request from a URL by concatenating the path and query portions of the URL with an intervening question mark.Thus, the Request-URI for the example URL would be

/a/b/c.txt?t=win&s=chess

- The final optional part of an http-scheme URL—the portion following but not including the number sign—is known as the fragment of the URL, and the string contained in the fragment is known as a fragment identifier. Fragment identifiers are used by browsers to scroll HTML documents; details are given in the next chapter, which covers HTML.

1.6.3 User-Controllable Features

- Graphical browsers also provide many user-controllable features, including: _
  - **Save:** Most documents can be saved by the user to the client machine's file system. If the document is an HTML page that contains other documents, such as images, then the browser will attempt to save all of these documents locally so that the entire page can be displayed from the local file system. A user saves a document in Mozilla under the **File|Save Page As** menu.

- **_Find in page:** Standard documents (text and HTML) can be searched with a function that is similar to that provided by most word processors. In Mozilla, the find function is accessed under the **Edit|Find in This Page** menu. (Mozilla also provides a “find as you type” feature under Edit that is similar to the incremental search in Emacs, for users familiar with that paradigm.)
- **Automatic form filling:** The browser can “remember” information entered on certain forms, such as billing address, phone numbers, etc. When another form is visited at a later date, the browser can automatically fill in previously saved data. The **Edit|Save Form Info** and **Edit|Fill in Form** menu options can be used to save and retrieve form information in Mozilla. The **Tools|Form Manager** menu can be used to manage saved form information.
- **Preferences:** Users can customize browser functionality in a wide variety of ways. In Mozilla, a window presenting preference options is obtained by selecting **Edit| Preferences** (Figure 1.5). The Appearance, Navigator, and Advanced categories (left subwindow) and their subcategories are used to customize Mozilla. Some preference settings directly related to the HTTP topics covered earlier are:
- Accept-Language: The non-∗ values sent by the browser for this HTTP request header field can be set under the **Navigator|Languages** category, **Languages for Web Pages** box.
- Default character set/encoding: The character set/encoding to be assumed for documents that do not specify one is also set under **Navigator|Languages** in the **Character Coding** box.
- Cache properties: The amount of local storage allocated to the cache and the conditions controlling when a cached file will be validated are set under **Advanced| Cache** in the **Set Cache Options** box.
- HTTP settings: The version of HTTP used and whether or not the client will keep connections alive is set under **Advanced|HTTP Networking** in the **Direct Connection Options** box.
- Style definition: The user can define certain aspects affecting how the browser renders HTML pages, such as font sizes, background and foreground colors, etc. In Mozilla, the font size can be modified using **View|Text Zoom**. If a page offers alternative styles, they can be selected using the **View|Use Style** menu .
- Document meta-information: Interested users can view information about the displayed document, such as the document’s MIME type, character encoding, size, and, if the document was written using HTML, the raw HTML source from which the rendering in the client areawas produced. In Mozilla,**View|Page Source** is used to viewrawHTML, and **View|Page Info** to view other so-called meta-information, that is, information about the document rather than information contained in the document itself.
- Themes: The look of one or more of the browser bars, particularly the navigation bar, can be modified by applying a certain theme (sometimes called a “skin”). In Mozilla, the browser scheme can be modified using **View|Apply Theme**. Additional themes can be obtained from **View|Apply Theme|Get New Themes**.
- **History:** The browser will automatically maintain a list of all pages visited within the last several days. Users can use the history list to easily return to

any recently visited page. In Mozilla, the history list can be reached by selecting **Go|History**.

- **Bookmarks** ("favorites" in Internet Explorer): Users can explicitly bookmark a web page, that is, save the URL for that page for an indefinite length of time. At any later time, the browser's bookmark facility can be used to easily return to any bookmarked page. Options under the **Bookmarks** menu in Mozilla allow users to bookmark a page, return to a bookmarked page, and edit the bookmark list.

1.6.4 Additional Functionality

- In addition to the facilities for end users described in the preceding subsection, browsers perform a number of other functions, including:
  - **Automatic URL completion:** If the user has entered a URL in the Location bar and begins to type it again (within the next several days), the URL will be completed automatically by the browser.
  - **Script execution:** In addition to displaying documents, browsers can run programs (scripts). These programs can perform a variety of tasks, from validating data entered on a form before sending it to a web server to creating various dynamic effects on web pages, such as drop-down menus.
  - **Event handling:** When the user performs an action, such as clicking on a link or a button in a web page, the browser treats this as the occurrence of an event. Browsers recognize a number of different types of events, including mouse button clicks, mouse movement, and even events not directly under user control such as the completion of the browser's rendering of a document. A browser can perform a variety of actions in response to an event—loading a document from a URL, clearing a form, or calling a script function defined by the document author. for example.
  - **Management of form GUI:** If a web page contains a form with fill-in fields, the browser must allow the user to perform standard text-editing functions within these fields. It also needs to automatically provide certain graphical feedback, such as changing a button image when it is pressed or providing a text cursor in a text field that will receive keyboard input.
  - **Secure communication:** When the user sends sensitive information, such as a credit card number, to a web server, the browser can encode this information in a way the prevents any machines along the IP route from the client to the server from obtaining the information.
  - **Plug-in execution:** While the browser itself normally understands only a limited number of MIME types, most browsers support some form of plug-in protocol that allows the browser's capabilities to be supplemented by other software. If a browser has a plugin for displaying, say, a document conforming to the application/pdf MIME type, then when the browser receives such a document it will pass it—via the plug-in protocol—to the appropriate plug-in for display. Some plug-ins may display the document within the browser's client area, while others may display in a separate window that is controlled by the plug-in itself. Plug-ins are often installed automatically, after user permission is obtained, when an unsupported MIME type is encountered. To see a list of plug-ins installed in your copy of Mozilla, select **Help|About Plug-ins**.

1.7 WEB SERVERS

1.7.1 Server Features

- The primary feature of every web server is to accept HTTP requests from web clients and return an appropriate resource (if available) in the HTTP response. Even this basic functionality involves a number of steps .
- The server calls on TCP software and waits for connection requests to one or more ports.
- When a connection request is received, the server dedicates a “subtask” to handling this connection.
- The subtask establishes the TCP connection and receives an HTTP request.
- The subtask examines the Host header field of the request to determine which “virtual host” should receive this request and invokes software for this host.
- The virtual host software maps the Request-URI field of the HTTP request start line to a resource on the server.
- If the resource is a file, the host software determines the MIME type of the file and creates an HTTP response that contains the file in the body of the response message.
- If the resource is a program, the host software runs the program, providing it with information from the request and returning the output from the program as the body of an HTTP response message.
- The server normally logs information about the request and response—such as the IP address of the requester and the status code of the response—in a plain-text file
- If the TCP connection is kept alive, the server subtask continues to monitor the connection until a certain length of time has elapsed, the client sends another request, or the client initiates a connection close.

1.7.2 Server History

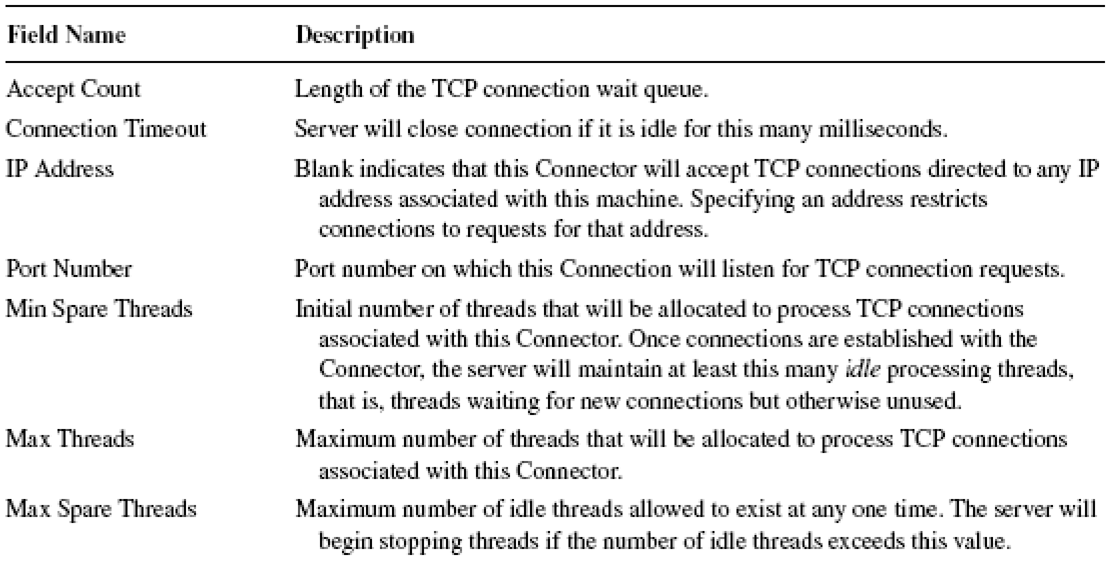
- Both servers can be configured to run a variety of types of programs, although certain programming languages tend to be used more frequently on one system than the other.
- For example, many IIS servers run programs written in VBScript (a derivative of Visual Basic), while a typical Apache server might run programs written in either Perl or the PHP scripting language (PHP stands for “PHP Hypertext Processor”; yes, the definition is infinitely recursive).
- A number of IIS and Apache servers also run Java programs. When running a Java program, both Apache and IIS servers are usually configured to run the program by using separate software called a servlet container. The servlet container provides the Java Virtual Machine that runs the Java program (known as a servlet), and also provides communication between the servlet and the Apache or IIS web server.

1.7.3 Server Configuration and Tuning

- Broadly speaking, server configuration can be broken into two areas: external communication and internal processing. In Tomcat, this corresponds to two separate Java packages: Coyote, which provides the HTTP/1.1 communication, and Catalina, which is the actual servlet container. Some of the Coyote parameters, affecting external communication, include the following:
  - IP addresses and TCP ports that may be used to connect to this server.

  - Number of subtasks (called threads in Java) that will be created when the server is initialized. This many TCP connections can be established simultaneously with minimal overhead.
  - Maximum number of threads that will be allowed to exist simultaneously. If this is larger than the previous value, then the number of threads maintained by the server may change, either up or down, over time.
  - Maximum number of TCP connection requests that will be queued if the server is already running its maximum number of threads. Connection requests received if the queue is full will be refused.
  - Length of time the server will wait after serving an HTTP request over a TCP connection before closing the connection if another request is not received.
  - The settings of these parameters can have a significant influence on the performance of a server; changing the values of these and similar parameters in order to optimize performance is often referred to as tuning the server.

- Load generation or stress test tools can be used to simulate requests to a web server, and can therefore be helpful for experimenting with tuning parameters based on anticipated traffic patterns even before a web site "goes live."
- The internal Catalina portion of Tomcat also has a number of parameter settings that affect functionality. These settings can determine:
  - Which client machines may send HTTP requests to the server.
  - Which virtual hosts are listening for TCP connections on a given port.
  - What logging will be performed.
  - How the path portion of Request-URIs will be mapped to the server's file system or other resources.
  - Whether or not the server's resources will be password protected.
  - Whether or not resources will be cached in the server's memory.

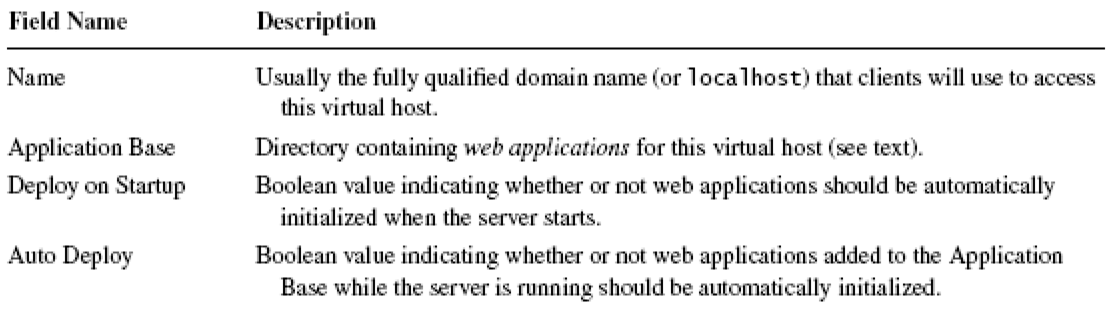


Field Name	Description
Accept Count	Length of the TCP connection wait queue.
Connection Timeout	Server will close connection if it is idle for this many milliseconds.
IP Address	Blank indicates that this Connector will accept TCP connections directed to any IP address associated with this machine. Specifying an address restricts connections to requests for that address.
Port Number	Port number on which this Connection will listen for TCP connection requests.
Min Spare Threads	Initial number of threads that will be allocated to process TCP connections associated with this Connector. Once connections are established with the Connector, the server will maintain at least this many *idle* processing threads, that is, threads waiting for new connections but otherwise unused.
Max Threads	Maximum number of threads that will be allocated to process TCP connections associated with this Connector.
Max Spare Threads	Maximum number of idle threads allowed to exist at any one time. The server will begin stopping threads if the number of idle threads exceeds this value.

Table 1.10 Some Of The Fields For The Connector Component

1.7.4 Defining Virtual Hosts

- The virtual host name should normally be a fully qualified domain name that would be used by visitors to your web site, although the Host supplied as part of the JWSDP Service is given the unqualified name localhost.
- This is a special name that the DNS system treats as a reference to a special IP address, 127.0.0.1. If an IP message is sent to this address, the IP software causes the message to loop back to itself for receipt.
- In short, browsing to a URL with domain name localhost causes the browser to send the HTTP request to a web server on the machine running the browser.

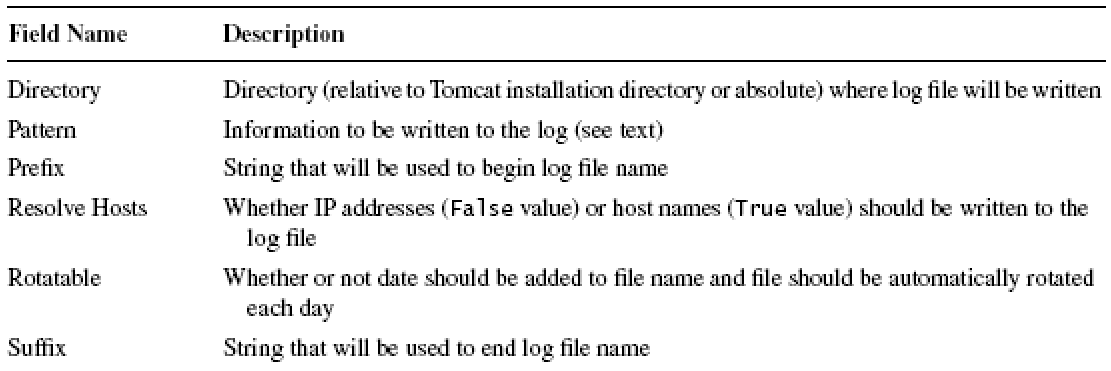


Field Name	Description
Name	Usually the fully qualified domain name (or `localhost`) that clients will use to access this virtual host.
Application Base	Directory containing *web applications* for this virtual host (see text).
Deploy on Startup	Boolean value indicating whether or not web applications should be automatically initialized when the server starts.
Auto Deploy	Boolean value indicating whether or not web applications added to the Application Base while the server is running should be automatically initialized.

Table 1.11 Key Fields for Host Component

1.7.5 Logging

- Web server logs record information about server activity. The primary web server log recording normal activity is an access log, a file that records information about every HTTP request processed by the server.
- A web server may also produce one or more message logs containing a variety of debugging and other information generated by web applications as well as possibly by the web server itself.
- Finally, information written to the standard output and error streams by the web server or applications may also be logged



Field Name	Description
Directory	Directory (relative to Tomcat installation directory or absolute) where log file will be written
Pattern	Information to be written to the log (see text)
Prefix	String that will be used to begin log file name
Resolve Hosts	Whether IP addresses (`False` value) or host names (`True` value) should be written to the log file
Rotatable	Whether or not date should be added to file name and file should be automatically rotated each day
Suffix	String that will be used to end log file name

Table 1.12 Key Fields for Valve Component of Type AccessLogValve

- The combination of values for the Directory, Prefix, Rotatable, and Suffix fields determine the file system path to the access log. The JWSDP Service settings for the values of these Valve fields cause the access log for this Service to be written to the logs directory under the JWSDP 1.3 installation directory in a file that starts with the string access log. and ends with the string .txt. In between these strings, because Rotatable is given the value True, Tomcat inserts the current date, in YYYY-MM-DD (year-month-day) format.

- The Pattern for the JWSDP Service access log Valve is %h %l %u %t "%r" %s %b This corresponds to what is often called the common access log format
- The following information is contained in this log entry:
  - Host name (or IP address; see Table 1.12) of client machine making the request.
  - User name used to log in, if server password protection is enabled.
  - Date and time of response, plus the time zone (offset from GMT) of the time.
  - Start line of HTTP request (quoted)
  - HTTP status code of response
  - Number of bytes sent in body of response

1.7.6 Access Control

- Tomcat can provide automatic password protection for resources that it serves. At its heart, this is a two-stage process.
- First, a database of user names is created. Each user name is assigned a password and a list of roles.
- The second stage of this process—associating resources with required roles—is normally performed by web application developers.
- The first stage—defining one or more user databases—can be performed by web system administrators, application developers, or both. The JWSDP Service contains an example of a database defined at the Service level through the use of a Realm component, which associates a user database with a Service.
- A coarser-grained access control can be provided by using Valve objects of type RemoteHostValve and RemoteAddressValve. Both are used to specify client machines that should be rejected if they request a connection to the server. They differ only in whether client machine host names or IP addresses are specified. Each type ofValve has two possible lists of clients: an Allow list and a Deny list. If one or more host names (comma-separated) is entered in the Allow list, then only these hosts can access the server.

1.7.7 Secure Servers

- In general, any machine other than the sender or receiver that extracts information from network messages is known as an eavesdropper.
- To prevent eavesdroppers from obtaining sensitive information, such as credit card numbers, all such sensitive information should be encrypted before being transmitted over any public communication network. The standard means of indicating to a browser that it should encrypt an HTTP request is to use the https scheme on the URL for the request. For example, entering the URL https://www.example.org in Mozilla's Location bar will cause the browser to attempt to send an encrypted HTTP GET request to www.example.org.
- A client browser that wishes to communicate securely with a server begins by initiating (over TCP/IP) a TLS Handshake with the server. During the Handshake process, the server and client agree on various parameters that will be used to encrypt messages sent between them. The server also sends a certificate to the client. The certificate enables the client to be sure that the machine it is communicating with is the one the client intends (as specified by the host name in the URL the browser is requesting). Certificates are necessary to avoid so-called man-in-the-middle attacks, in which some machine intercepts a message intended

for another machine (the target), prevents the message from further forwarding, and returns an HTTP reply to the sender pretending to be from the target.

- Tomcat supports the TLS 1.0 and earlier protocols. To enable the secure server Tomcat features, you must do two things:
    - ➢ Obtain and install a certificate.
    - ➢ Configure the server to listen for TLS connections on some port.

1.8 HTML5

- HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language.
- HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages.
- Markup language is used to define the text document within tag which defines the structure of web pages. HTML 5 is the fifth and current version of HTML.
- It has improved the markup available for documents and has introduced application programming interfaces(API) and Document Object Model(DOM).

Features:

- It has introduced new multimedia features which supports audio and video controls by using <audio> and <video> tags.
- There are new graphics elements including vector graphics and tags.
- Enrich semantic content by including <header> <footer>, <article>, <section> and <figure> are added.
- Drag and Drop- The user can grab an object and drag it further dropping it on a new location.
- Geo-location services- It helps to locate the geographical location of a client.
- Web storage facility which provides web application methods to store data on web browser.
- Uses SQL database to store data offline.
- Allows to draw various shapes like triangle, rectangle, circle, etc.
- Capable of handling incorrect syntax.
- Easy DOCTYPE declaration i.e. <!doctype html>
- Easy character encoding i.e. <meta charset="UTF-8″>

Removed elements from HTML 5: There are many elements which are depreciated from HTML 5 are listed below:

Removed Elements	**Use Instead Elements**
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
<center>	CSS
<dir>	<ul>
<font>	CSS
<frame>	
<frameset>	
<noframes>	

Removed Elements	**Use Instead Elements**
<isindex>	
<strike>	CSS, <s> or <del>
<tt>	CSS

New Added Elements in HTML 5:

- **<article>:** The <article> tag is used to represent an article. More specifically, the content within the <article> tag is independent from the other content of the site (even though it can be related).
- **<aside>:** The <aside> tag is used to describe the main object of the web page in a shorter way like a highlighter. It basically identifies the content that is related to the primary content of the web page but does not constitute the main intent of the primary page. The <aside> tag contains mainly author information, links, related content and so on.
- **<figcaption>:** The <figurecaption> tag in HTML is used to set a caption to the figure element in a document.
- **<figure>:** The <figure> tag in HTML is used to add self-contained content like illustrations, diagrams, photos or codes listing in a document. It is related to main flow but it can be used in any position of a document and the figure goes with the flow of the document and if remove it then it should not affect the flow of the document.
- **<header>:** It contains the section heading as well as other content, such as a navigation links, table of contents, etc.
- **<footer>:** The <footer> tag in HTML is used to define a footer of HTML document. This section contains the footer information (author information, copyright information, carriers etc). The footer tag are used within body tag. The <footer> tag is new in the HTML 5. The footer elements require a start tag as well as an end tag.
- **<main>:** Delineates the main content of the body of a document or web app.
- **<mark>:** The <mark> tag in HTML is used to define the marked text. It is used to highlight the part of the text in the paragraph.
- **<nav>:** The <nav> tag is used to declaring the navigational section in HTML documents. Websites typically have sections dedicated to navigational links, which enables user to navigate the site. These links can be placed inside a nav tag.
- **<section>:** It demarcates a thematic grouping of content.
- **<details>:** The <details> tag is used for the content/information which is initially hidden but could be displayed if the user wishes to see it. This tag is used to create interactive widget which user can open or close it. The content of details tag is visible when open the set attributes.
- **<summary>:** The <summary> tag in HTML is used to define a summary for the <details> element. The <summary> element is used along with the <details> element and provides a summary visible to the user. When the summary is clicked by the user, the content placed inside the <details> element becomes visible which was previously hidden. The <summary> tag was added in HTMl 5. The <summary> tag requires both starting and ending tag.
- **<time>:** The <time> tag is used to display the human-readable data/time. It can also be used to encode dates and times in a machine-readable form. The main advantage

for users is that they can offer to add birthday reminders or scheduled events in their calender's and search engines can produce smarter search results.

- **<bdi>:** The <bdi> tag refers to the Bi-Directional Isolation. It differentiate a text from other text that may be formatted in different direction. This tag is used when a user generated text with an unknown directions.
- **<wbr>:** The <wbr> tag in HTML stands for word break opportunity and is used to define the position within the text which is treated as a line break by the browser. It is mostly used when the used word is too long and there are chances that the browser may break lines at the wrong place for fitting the text.
- **<datalist>:** The <datalist> tag is used to provide autocomplete feature in the HTML files. It can be used with input tag, so that users can easily fill the data in the forms using select the data.
- **<keygen>:** The <keygen> tag in HTML is used to specify a key-pair generator field in a form. The purpose of <keygen> element is to provide a secure way to authenticate users. When a from is submitted then two keys are generated, private key and public key. The private key stored locally, and the public key is sent to the server. The public key is used to generate client certificate to authenticate user for future.
- **<output>:** The <output> tag in HTML is used to represent the result of a calculation performed by the client-side script such as JavaScript.
- **<progress>:** It is used to represent the progress of a task. It is also define that how much work is done and how much is left to download a things. It is not used to represent the disk space or relevant query.
- **<svg>:** It is the Scalable Vector Graphics.
- **<canvas>:** The <canvas> tag in HTML is used to draw graphics on web page using JavaScript. It can be used to draw paths, boxes, texts, gradient and adding images. By default it does not contains border and text.
- **<audio>:** It defines the music or audio content.
- **<embed>:** Defines containers for external applications (usually a video player).
- **<source>:** It defines the sources for <video> and <audio>.
- **<track>:** It defines the tracks for <video> and <audio>.
- **<video>:** It defines the video content.

Advantages:

- All browsers supported.
- More device friendly.
- Easy to use and implement.
- HTML 5 in integration with CSS, JavaScript, etc can help build beautiful websites.

Disadvantages:

- Long codes have to be written which is time consuming.
- Only modern browsers support it.

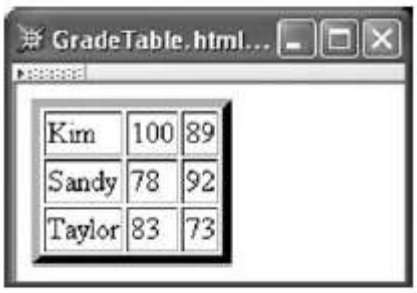
1.9 TABLES

- HTML provides a fairly sophisticated model for presenting data in tabular form.
- Columns and rows will automatically size to contain their data, although there are also various ways to specify column widths; individual table cells can span multiple rows and/or columns; header and/or footer rows can be supplied; and so on. There are also various options for changing the visual appearance of a table, such as the widths of its internal cell-separating lines (rules) and external borders.

- Simple tables are simple to represent in HTML. For example, a table of student grades could be written as follows and produces the table shown in Figure 1.1

```
<table border="5">
<tr>
<td>Kim</td><td>100</td><td>89</td>
</tr>
<tr>
<td>Sandy</td><td>78</td><td>92</td>
</tr>
<tr>
<td>Taylor</td><td>83</td><td>73</td>
</tr>
</table>
```

- A tr (table row) element is used to contain each row. Within a row, a td (table data) element marks each element of the row. Notice that we don't need to specify the number of rows and columns in the table explicitly. Instead, these values are determined automatically: in a simple table, the number of rows is determined by the number of tr elements in the table, and the number of columns is determined by the maximum number of td elements contained within any row



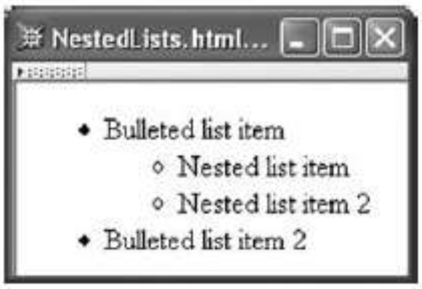
Kim	100	89
Sandy	78	92
Taylor	83	73

Fig 1.16 A simple table of grades.

- In this example, since there are three tr elements, each containing three td elements, the table is 3 by 3. Finally, notice that the width of table columns is also automatically adjusted to contain the maximum width item in any column, although this can be overridden via the style attribute

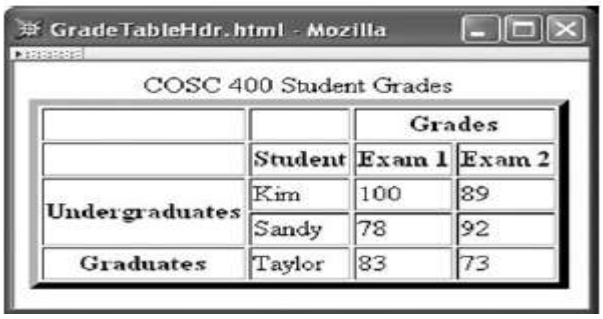
```
<table border="5">
<caption>
COSC 400 Student Grades
</caption>
<tr>
<td>&nbsp;</td><td>&nbsp;</td><th colspan="2">Grades</th>
</tr>
<tr>
<td>&nbsp;</td><th>Student</th><th>Exam 1</th><th>Exam 2</th>
</tr>
<tr>
<th rowspan="2">Undergraduates</th><td>Kim</td><td>100</td><td>89</td>
</tr>
<tr>
<td>Sandy</td><td>78</td><td>92</td>
```

</tr>
<tr>



- Bulleted list item
  - Nested list item
  - Nested list item 2
- Bulleted list item 2

<th>Graduates</th><td>Taylor</td><td>83</td><td>73</td>
</tr>
</table>



COSC 400 Student Grades

		Grades	
	Student	Exam 1	Exam 2
Undergraduates	Kim	100	89
	Sandy	78	92
Graduates	Taylor	83	73

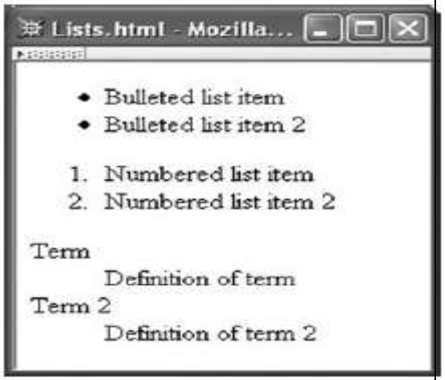
Fig.1.2 Table with headings and caption.

- Two new elements are used in this example. The caption element, as the name implies, is used to define a caption for the table. If a caption element is used with a table, the caption start tag must must appear immediately after the start tag of the table element.
- The second new element, th (table header), is much like the td element, except that a typical browser will format the content of a th element in boldface and center it horizontally within the column.

1.10 LISTS

- Fig 1.3 illustrates the three types of lists supported by HTML:
  - ➢ Unordered: A bullet list
  - ➢ Ordered: A numbered list
  - ➢ Definition: A list of terms and definitions for each
  - ➢ This figure was produced by the following HTML:

```
<ul>
<li>Bulleted list item</li>
<li>Bulleted list item 2</li>
        </ul>
<ol>
<li>Numbered list item</li>
<li>Numbered list item 2</li>
</ol>
<dl>
<dt>Term</dt>
<dd>Definition of term</dd>
<dt>Term 2</dt>
        <dd>Definition of term 2</dd></dl>
```



- Bulleted list item
- Bulleted list item 2

1. Numbered list item
2. Numbered list item 2

Term
    Definition of term
Term 2
    Definition of term 2

Fig1.4 Browser rendering nested unordered lists.

- Lists can be nested to produce an outline layout. For example, the markup

```
<ul
<li>Bulleted list item                                  .
<ul>
<li>Nested list item</li>
<li>Nested list item 2</li>
</ul>
</li>
<li>Bulleted list item 2</li>
        </ul>
```

1.11 IMAGES

Images can improve the design and the appearance of a web page.

HTML Images Syntax

- In HTML, images are defined with the <img> tag.
- The <img> tag is empty, it contains attributes only, and does not have a closing tag.
- The src attribute specifies the URL (web address) of the image:

```
<img src="url">
```

The alt Attribute

- The alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).
- The value of the alt attribute should describe the image:

Example

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

Image Size - Width and Height

The style attribute to specify the width and height of an image.

Example

```
<img src="img_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">
```

Note: Always specify the width and height of an image. If width and height are not specified, the page might flicker while the image loads. Width and Height, or Style?.**The width, height, and style attributes are valid in HTML.**However, we suggest using the style attribute. It prevents styles sheets from changing the size of images:

Example

```
<!DOCTYPE.html>
<html>
<head>
<style>
img{
  width:100%;
}
</style>
</head>
<body>
```

```
<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">
<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">
</body>
</html>
```

Images in Another Folder

If not specified, the browser expects to find the image in the same folder as the web page.

However, it is common to store images in a sub-folder. You must then include the folder name in the src attribute:

Example

```
<img src="/images/html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">
```

Images on Another Server

- Some web sites store their images on image servers.
- Actually, you can access images from any web address in the world:

Example

```
<img src="https://www.w3schools.com/images/w3schools_green.jpg" alt="W3Schools.com">
```

Animated Images

HTML allows animated GIFs:

Example

```
<img src="programming.gif" alt="Computer Man" style="width:48px;height:48px;">
```

Image as a Link

To use an image as a link, put the <img> tag inside the <a> tag:

Example

```
<ahref="default.asp">
  <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;border:0;">
</a>
```

Image Floating

Use the CSS float property to let the image float to the right or to the left of a text:

Example

```
<p><img src="smiley.gif" alt="Smiley face" style="float:right;width:42px;height:42px;">
The image will float to the right of the text.</p>
<p><img src="smiley.gif" alt="Smiley face" style="float:left;width:42px;height:42px;">
The image will float to the left of the text.</p>
```

HTML Image Tags

Tag	**Description**
<img>	Defines an image
<map>	Defines an image-map
<area>	Defines a clickable area inside an image-map
<picture>	Defines a container for multiple image resources

Image Maps

- The <map> tag defines an image-map. An image-map is an image with clickable areas.

- Click on the computer, the phone, or the cup of coffee in the image below:

Example

```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap">
<mapname="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
  <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">
</map>
```

The Image

The image is inserted using the <img> tag. The only difference from other images is that you must add a usemap attribute:

<img src="workplace.jpg" alt="Workplace" usemap="#workmap">

The usemap value starts with a hash tag # followed by the name of the image map, and is used to create a relationship between the image and the image map.

1.12 HTML5 CONTROL ELEMENTS

- An HTML element usually consists of a **start** tag and an **end** tag, with the content inserted in between:

<tagname>Content goes here...</tagname>

- The HTML **element** is everything from the start tag to the end tag:

<p>My first paragraph.</p>

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
<p>	My first paragraph.	</p>
<br>		

1.12.1 Nested HTML Elements

- HTML elements can be nested (elements can contain elements).
- All HTML documents consist of nested HTML elements.
- This example contains four HTML elements:

Example

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

- The <html> element defines the **whole document**.
- It has a **start** tag <html> and an **end** tag </html>.
- Inside the <html> element is the <body> element.

1.12.2 HTML New Tag Elements

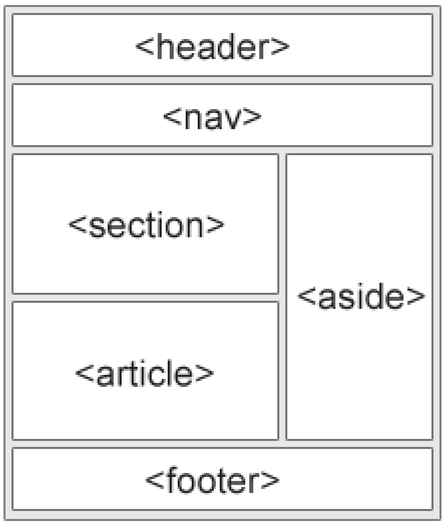
Tags (Elements)	**Description**
<article>	Represents an independent piece of content of a document, such as a blog entry or newspaper article
<aside >	Represents a piece of content that is only slightly related to the rest of the page.
<audio>	Defines an audio file.
<canvas>	This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<command>	Represents a command the user can invoke.
<datalist>	Together with the a new list attribute for input can be used to make comboboxes
<details>	Represents additional information or controls which the user can obtain on demand
<embed>	Defines external interactive content or plugin.
<figure>	Represents a piece of self-contained flow content, typically referenced as a single unit from the main flow of the document.
<footer>	Represents a footer for a section and can contain information about the author, copyright information, et cetera.
<header>	Represents a group of introductory or navigational aids.
<hgroup>	Represents the header of a section.
<keygen>	Represents control for key pair generation.
<mark>	Represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.
<meter>	Represents a measurement, such as disk usage.
<nav>	Represents a section of the document intended for navigation.
<output>	Represents some type of output, such as from a calculation done through scripting.
<progress>	Represents a completion of a task, such as downloading or when performing a series of expensive operations.
<ruby>	Together with <rt> and <rp> allow for marking up ruby annotations.
<section>	Represents a generic document or application section
<time>	Represents a date and/or time.
<video>	Defines a video file.
<wbr>	Represents a line break opportunity.

- New types for <input> tag
- The input element's type attribute now has the following new values −

Type	**Description**
color	Color selector, which could be represented by a wheel or swatch picker
date	Selector for calendar date

datetime-local	Date and time display, with no setting or indication for time zones
datetime	Full date and time display, including a time zone.
email	Input type should be an email.
month	Selector for a month within a given year
number	A field containing a numeric value only
range	Numeric selector within a range of values, typically visualized as a slider
search	Term to supply to a search engine. For example, the search bar atop a browser.
tel	Input type should be telephone number.
time	Time indicator and selector, with no time zone information
url	Input type should be URL type.
week	Selector for a week within a given year

1.13 SEMANTIC ELEMENTS



- Semantics is the study of the meanings of words and phrases in a language.
- Semantic elements = elements with a meaning.

What are Semantic Elements?

- A semantic element clearly describes its meaning to both the browser and the developer.
- Examples of non-semantic elements: <div> and <span> - Tells nothing about its content.
- Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.

New Semantic Elements in HTML5

- Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer">to indicate navigation, header, and footer.
- HTML5 offers new semantic elements to define different parts of a web page:
  - ➢ <article>
  - ➢ <aside>
  - ➢ <details>
  - ➢ <figcaption>
  - ➢ <figure>
  - ➢ <footer>
  - ➢ <header>
  - ➢ <main>
  - ➢ <mark>
  - ➢ <nav>
  - ➢ <section>
  - ➢ <summary>
  - ➢ <time>

HTML5 <section> Element

- The <section> element defines a section in a document.
- According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."
- A home page could normally be split into sections for introduction, content, and contact information.

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

HTML5 <article> Element

- The <article> element specifies independent, self-contained content.
- An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.
- Examples of where an <article> element can be used:
    - Forum post
    - Blog post
    - Newspaper article

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural
environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

Nesting <article> in <section> or Vice Versa?

- The<article> element specifies independent, self-contained content.
- The <section> element defines section in a document.

HTML5 <header> Element

- The <header> element specifies a header for a document or section.
- The <header> element should be used as a container for introductory content.
- The following example defines a header for an article:

Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML5 <footer> Element

- The <footer> element specifies a footer for a document or section.
- A <footer> element should contain information about its containing element.
- A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

HTML5 <nav> Element

- The <nav> element defines a set of navigation links.

Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

HTML5 <aside> Element

- The <aside> element defines some content aside from the content it is placed in (like a sidebar).
- The <aside> content should be related to the surrounding content.

Example

```
<p>My family and I visited The Epcot center this summer.</p>

<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

HTML5 <figure> and <figcaption> Elements

- The purpose of a figure caption is to add a visual explanation to an image.
- In HTML5, an image and a caption can be grouped together in a <figure> element:

Example

```
<figure>
  <img src="pic_trulli.jpg" alt="Trulli">
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

Why Semantic Elements?

- With HTML4, developers used their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.
- This made it impossible for search engines to identify the correct web page content.
- With the new HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier.
- According to the W3C, a Semantic Web: "Allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML5

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content

<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

1.14 HTML5 DRAG AND DROP

- Drag and Drop (DnD) is powerful User Interface concept which makes it easy to copy, reorder and deletion of items with the help of mouse clicks. This allows the user to click and hold the mouse button down over an element, drag it to another location, and release the mouse button to drop the element there.
- To achieve drag and drop functionality with traditional HTML4, developers would either have to either have to use complex JavaScript programming or other JavaScript frameworks like jQuery etc.
- Now HTML 5 came up with a Drag and Drop (DnD) API that brings native DnD support to the browser making it much easier to code up.
- HTML 5 DnD is supported by all the major browsers like Chrome, Firefox 3.5 and Safari 4 etc.

Drag and Drop Events

There are number of events which are fired during various stages of the drag and drop operation. These events are listed below −

S.No	Events & Description
1	Dragstart Fires when the user starts dragging of the object.
2	Dragenter Fired when the mouse is first moved over the target element while a drag is occurring. A listener for this event should indicate whether a drop is allowed over this location. If there are no listeners, or the listeners perform no operations, then a drop is not allowed by default.
3	Dragover This event is fired as the mouse is moved over an element when a drag is occurring. Much of the time, the operation that occurs during a listener will be the same as the dragenter event.
4	Dragleave This event is fired when the mouse leaves an element while a drag is occurring. Listeners should remove any highlighting or insertion markers used for drop

	feedback.
5	Drag Fires every time the mouse is moved while the object is being dragged.
6	Drop The drop event is fired on the element where the drop was occurred at the end of the drag operation. A listener would be responsible for retrieving the data being dragged and inserting it at the drop location.
7	Dragend Fires when the user releases the mouse button while dragging an object.

The DataTransfer Object

- The event listener methods for all the drag and drop events accept Event object which has a readonly attribute called dataTransfer.
- The event.dataTransfer returns DataTransfer object associated with the event as follows −

```
function EnterHandler(event) {
  DataTransfer dt = event.dataTransfer;
  ............
}
```

- The *DataTransfer* object holds data about the drag and drop operation. This data can be retrieved and set in terms of various attributes associated with DataTransfer object as explained below −

Sr.No.	DataTransfer attributes and their description
1	dataTransfer.dropEffect [= value] Returns the kind of operation that is currently selected. This attribute can be set, to change the selected operation. The possible values are none, copy, link, and move.
2	dataTransfer.effectAllowed [= value
	Returns the kinds of operations that are to be allowed. This attribute can be set, to change the allowed operations. The possible values are none, copy, copyLink, copyMove, link, linkMove, move, all and uninitialized.
3	dataTransfer.types
	Returns a DOMStringList listing the formats that were set in the dragstart event. In addition, if any files are being dragged, then one of the types will be the string "Files".
4	dataTransfer.clearData ([format])
	Removes the data of the specified formats. Removes all data if the argument is omitted.
5	dataTransfer.setData(format, data)
	Adds the specified data.

6	data = dataTransfer.getData(format)
	Returns the specified data. If there is no such data, returns the empty string.
7	dataTransfer.files
	Returns a FileList of the files being dragged, if any.

Drag and Drop Process

- Following are the steps to be carried out to implement Drag and Drop operation −

Step 1 - Making an Object Draggable

Here are steps to be taken −

- If you want to drag an element, you need to set the **draggable** attribute to **true** for that element.
- Set an event listener for **dragstart** that stores the data being dragged.
- The event listener **dragstart** will set the allowed effects (copy, move, link, or some combination).

```
<!DOCTYPE HTML>

<html>

  <head>

        <style type = "text/css">

            #boxA, #boxB {

        float:left;padding:10px;margin:10px; -moz-user-select:none;

      }

      #boxA { background-color: #6633FF; width:75px; height:75px;  }

      #boxB { background-color: #FF6699; width:150px; height:150px; }

    </style>

        <script type = "text/javascript">

            function dragStart(ev) {

        ev.dataTransfer.effectAllowed = 'move';

        ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));

        ev.dataTransfer.setDragImage(ev.target,0,0);

                return true;

      }

    </script>

     </head>
  <body>
        <center>
      <h2>Drag and drop HTML5 demo</h2>
      <div>Try to drag the purple box around.</div>
            <div id = "boxA" draggable = "true"
        ondragstart = "return dragStart(ev)">
        <p>Drag Me</p>
      </div>
         <div id = "boxB">Dustbin</div>
    </center>
      </body>
</html>
```

Step 2 - Dropping the Object

To accept a drop, the drop target has to listen to at least three events.

- The **dragenter** event, which is used to determine whether or not the drop target is to accept the drop. If the drop is to be accepted, then this event has to be canceled.
- The **dragover** event, which is used to determine what feedback is to be shown to the user. If the event is canceled, then the feedback (typically the cursor) is updated based on the dropEffect attribute's value.
- Finally, the **drop** event, which allows the actual drop to be performed.

Live Demo

```
<html>
  <head>
    <style type="text/css">
      #boxA, #boxB {
        float:left;padding:10px;margin:10px;-moz-user-select:none;
      }
      #boxA { background-color: #6633FF; width:75px; height:75px;  }
      #boxB { background-color: #FF6699; width:150px; height:150px; }
    </style>
    <script type="text/javascript">
      function dragStart(ev) {
            ev.dataTransfer.effectAllowed='move';
            ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
            ev.dataTransfer.setDragImage(ev.target,0,0);
            return true;
         }
         function dragEnter(ev) {
            event.preventDefault();
            return true;
         }
         function dragOver(ev) {
            return false;
         }
         function dragDrop(ev) {
            var src = ev.dataTransfer.getData("Text");
            ev.target.appendChild(document.getElementById(src));
            ev.stopPropagation();
            return false;
         }
      </script>
   </head>
   <body>
      <center>
         <h2>Drag and drop HTML5 demo</h2>
         <div>Try to move the purple box into the pink box.</div>
         <div id="boxA" draggable="true" ondragstart="return dragStart(event)">
            <p>Drag Me</p>
         </div>
         <div id="boxB" ondragenter="return dragEnter(event)" ondrop="return
dragDrop(event)" ondragover="return dragOver(event)">Dustbin</div>
      </center>  </body>
</html>
```

1.15 HTML AUDIO/VIDEO DOM

HTML Audio and Video DOM Reference

- The HTML5 DOM has methods, properties, and events for the <audio> and <video> elements.
- These methods, properties, and events allow you to manipulate <audio> and <video> elements using JavaScript.

HTML Audio/Video Methods

Method	**Description**
addTextTrack()	Adds a new text track to the audio/video
canPlayType()	Checks if the browser can play the specified audio/video type
load()	Re-loads the audio/video element
play()	Starts playing the audio/video
pause()	Pauses the currently playing audio/video

HTML Audio/Video Properties

Property	**Description**
audioTracks	Returns an AudioTrackList object representing available audio tracks Autoplay
buffered	Returns a TimeRanges object representing the buffered parts of the audio/video
controller	Returns the MediaController object representing the current media controller of the audio/video
controls	Sets or returns whether the audio/video should display controls (like play/pause etc.)
currentTime	Returns the URL of the current audio/video
defaultMuted	Sets or returns whether the audio/video should be muted by default

HTML Audio/Video Events

Event	**Description**
error	Fires when an error occurred during the loading of an audio/video
loadeddata	Fires when the browser has loaded the current frame of the audio/video
loadedmetadata	Fires when the browser has loaded meta data for the audio/video
loadstart	Fires when the browser starts looking for the audio/video
pause	Fires when the audio/video has been paused
play	Fires when the audio/video has been started or is no longer paused
playing	Fires when the audio/video is playing after having been paused or

	stopped for buffering
progress	Fires when the browser is downloading the audio/video
ratechange	Fires when the playing speed of the audio/video is changed
seeked	Fires when the user is finished moving/skipping to a new position in the audio/video
seeking	Fires when the user starts moving/skipping to a new position in the audio/video
stalled	Fires when the browser is trying to get media data, but data is not available
suspend	Fires when the browser is intentionally not getting media data
timeupdate	Fires when the current playback position has changed
volumechange	Fires when the volume has been changed
waiting	Fires when the video stops because it needs to buffer the next frame

1.16 Css3

Types of CSS (Cascading Style Sheet)

- Cascading Style Sheet(CSS) is used to set the style in web pages which contain HTML elements. It sets the background color, font-size, font-family, color, … etc property of elements in a web pages.
- There are three types of CSS which are given below:
    - Inline CSS
    - Internal or Embedded CSS
    - External CSS

Inline CSS:

Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using style attribute.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline CSS</title>
  </head>

  <body>
    <p style = "color:#009900; font-size:50px;
        font-style:italic; text-align:center;">
      GeeksForGeeks
    </p>
  </body>
</html>
```

Internal or Embedded CSS:

This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.

```
<!DOCTYPE html>
<html>
```

```
  <head>
    <title>Internal CSS</title>
    <style>
      .main {
        text-align:center;
      }
      .GFG {
        color:#009900;
        font-size:50px;
        font-weight:bold;
      }
      .geeks {
        font-style:bold;
        font-size:20px;
      }
    </style>
  </head>
  <body>
    <div class = "main">
      <div class ="GFG">GeeksForGeeks</div>

      <div class ="geeks">
        A computer science portal for geeks
      </div>
    </div>
  </body>
</html>
```

External CSS:

External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, … etc). CSS property written in a separate file with .css extension and should be linked to the HTML document using **link** tag. This means that for each element, style can be set only once and that will be applied across web pages.

Example: The file given below contains CSS property. This file save with .css extension. For Ex: **geeks.css**

```
body {
  background-color:powderblue;
}
.main {
  text-align:center;
}
.GFG {
  color:#009900;
  font-size:50px;
  font-weight:bold;
}
#geeks {
  font-style:bold;
  font-size:20px;
}
```

Below is the HTML file that is making use of the created external style sheet

- **link** tag is used to link the external style sheet with the html webpage.
- **href** attribute is used to specify the location of the external style sheet file.

filter_none

edit

play_arrow

brightness_4

```
<!DOCTYPE html>
<html>
   <head>
      <link rel="stylesheet" href="geeks.css"/>
   </head>

   <body>
      <div class = "main">
         <div class ="GFG">GeeksForGeeks</div>
         <div id ="geeks">
            A computer science portal for geeks
         </div>
      </div>
   </body>
</html>
```

Properties of CSS: Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page. If for an HTML tag, styles are defined in multiple style sheets then the below order will be followed.

- As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.
- Internal or Embedded stands second in the priority list and overrides the styles in the external style sheet.
- External style sheets have the least priority. If there are no styles defined either in inline or internal style sheet then external style sheet rules are applied for the HTML tags.

Inheritance and cascade

Inheritance and the cascade are two fundamental concepts in CSS, that are important to understand. The two concepts are closely related, yet different:

- Inheritance is associated with how the elements in the HTML markup inherit properties from their parent (containing) elements and pass them on to their children.
- The cascade relates to CSS declarations being applied to a document, and how conflicting rules do or do not override each other.

1.17 RULE CASCADING

- A single style sheet associated with one or more web pages is valuable, but in quite a limited way.
- For small sites, the single style sheet is sufficient, but for larger sites, especially sites managed by more than one person (perhaps several teams who may never

communicate) single style sheets don't provide the ability to share common styles, and extend these styles where necessary. This can be a significant limitation.

- Cascading style sheets are unlike the style sheets you might have worked with using word processors, because they can be linked together to create a hierarchy of related style sheets.

Font Family

- The font family of a text is set with the font-family property.The font-family property should hold several font names as a "fallback" system.
- If the browser does not support the first font, it tries the next font.Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Example

```
p{font-family:"Times New Roman", Times, serif;}
```

Font Style

The font-style property is mostly used to specify italic text. This property has three values:

- ➢ normal - The text is shown normally italic - The text is shown in italics
- ➢ oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Font Size

- The font-size property sets the size of the text.Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- The font-size value can be an absolute, or relative size.
    - ➢ Absolute size:Sets the text to a specified size Does not allow a user to change the text size in all browsers (bad for accessibility reasons).Absolute size is useful when the physical size of the output is known
    - ➢ Relative size: Sets the size relative to surrounding elements.Allows a user to change the text size in browsers

The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

- The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.
- The box model allows us to place a border around elements and space elements in relation to other elements.

Explanation of the different parts:

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent.
- **Border** - A border that goes around the padding and content. The border is affected by the background color of the box**.**
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box .
- **Content** - The content of the box, where text and images appear.
- In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

1.18 INHERITANCE AND CASCADE

Inheritance and the cascade are two fundamental concepts in CSS, that are important to understand. The two concepts are closely related, yet different:

- Inheritance is associated with how the elements in the HTML markup inherit properties from their parent (containing) elements and pass them on to their children.
- The cascade relates to CSS declarations being applied to a document, and how conflicting rules do or do not override each other.

Inheritance

- Inheritance is the mechanism by which certain properties are passed on from a parent element down to its children, in the same fashion as genetics: if parents have blue eyes, their children will probably also have blue eyes.
- Not all CSS properties are inherited, because it does not make sense for some of them to be.

Why inheritance is useful

- CSS has an inheritance mechanism because otherwise CSS rules would be redundant. Without inheritance, it would be necessary to specify styles like font family, font size, and text color individually — for every single element type. The code would become bloated and repetitive.
- Using inheritance, you can specify the font properties for the html or body elements and the styles will be inherited by all other elements. You can specify background and text colors for a specific container element and the text color will automatically be inherited by any child elements in that container.
- The background color is not inherited, but the initial value for background-color is transparent, which means a parent’s background will shine through. The effect is similar to the page’s appearance if background colors were inherited.

How inheritance works

- Every element in an HTML document inherits all inheritable properties from its parent except the root element (<html>), which does not have a parent.
- Whether or not the inherited properties will have any effect depends on other things, as described later in the section about the cascade.
- Just as a blue-eyed mother can have a brown-eyed child if the father has brown eyes, inherited properties in CSS can be overridden.
- **An example of inheritance** Copy the following HTML document into a new file in your favorite text editor and save it as inherit.html.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Inheritance</title>
  </head>
  <body>
   <h1>Heading</h1>
   <p>A paragraph of text.</p>
  </body>
</html>
```

Create a new empty file in your text editor, copy the following CSS rule into it, and save the file as style.css in the same location as the HTML file.

html {
font: 75% Verdana, sans-serif;
}

Link the style sheet to your HTML document by inserting the following line before the *</head> tag:*
<link rel="stylesheet" type="text/css" href="style.css">
Save the modified HTML file and reload the document in your browser.

The CSS rule only specified two properties — the font size and the font family — but that rule is equivalent to the following:

```
html {
  font-style: normal;
  font-variant: normal;
  font-weight: normal;
  font-size: 75%;
  line-height: normal;
  font-family: Verdana,sans-serif;
}
```

1.19 BACKGROUNDS

CSS Background

CSS background properties are used to define the background effects of an element. CSS properties used for background effects:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

1.19.1 Background Color

The background-color property specifies the background color of an element. The background color of a page is defined in the body selector:

Example

body {background-color:#b0c4de;}

The background color can be specified by:

name - a color name, like "red"

RGB - an RGB value, like "rgb(255,0,0)" Hex - a hex value, like "#ff0000"

1.19.2 Background Image

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element. The background image for a page can be set like this:

Example

body {background-image:url('paper.gif');}

1.19.3 Background Image - Repeat Horizontally or Vertically

By default, the background-image property repeats an image both horizontally and vertically. Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Example

```
body
{
 background-image:url('gradient2.png');
 }
```

- If the image is repeated only horizontally (repeat-x), the background will look better:

Example

```
body
{
 background-image:url('gradient2.png'); background-repeat:repeat-x;
}
```

1.19.4 CSS Multiple Backgrounds

- CSS allows you to add multiple background images for an element, through the background-image property.
- The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.
- The following example has two background images, the first image is a flower (aligned to the bottom and right) and the second image is a paper background (aligned to the top-left corner):

Example

```
#example1 {
  background-image: url(img_flwr.gif), url(paper.gif);
  background-position: right bottom, left top;
  background-repeat: no-repeat, repeat;
}
```

- Multiple background images can be specified using either the individual background properties (as above) or the background shorthand property.

1.19.5 CSS Background Size

- The CSS background-size property allows you to specify the size of background images.
- The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.
- The following example resizes a background image to much smaller than the original image (using pixels):

Example

```
#div1 {
  background: url(img_flower.jpg);
  background-size: 100px 80px;
  background-repeat: no-repeat;
}
```

- The two other possible values for background-size are contain and cover.
- The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there

may be some areas of the background which are not covered by the background image.

- The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area.
- The following example illustrates the use of contain and cover:

Example

```
#div1 {
  background: url(img_flower.jpg);
  background-size: contain;
  background-repeat: no-repeat;
}
#div2 {
  background: url(img_flower.jpg);
  background-size: cover;
  background-repeat: no-repeat;
}
```

1.20 BACKGROUND IMAGES

- The background-size property also accepts multiple values for background size (using a comma-separated list), when working with multiple backgrounds.
- The following example has three background images specified, with different background-size value for each image:

Example

```
#example1 {
  background: url(img_tree.gif) left top no-repeat, url(img_flwr.gif) right bottom no-repeat,
url(paper.gif) left top repeat;
  background-size: 50px, 130px, auto;
}
```

Full Size Background Image

- Now we want to have a background image on a website that covers the entire browser window at all times.
- The requirements are as follows:
  - ➢ Fill the entire page with the image (no white space)
  - ➢ Scale image as needed
  - ➢ Center image on page
  - ➢ Do not cause scrollbars
  - ➢ The following example shows how to do it; Use the <html> element (the <html> element is always at least the height of the browser window). Then set a fixed and centered background on it. Then adjust its size with the background-size property:

Example

```
html {
  background: url(img_man.jpg) no-repeat center fixed;
  background-size: cover;
}
```

CSS background-origin Property

- The CSS background-origin property specifies where the background image is positioned.
- The property takes three different values:
  - border-box - the background image starts from the upper left corner of the border
  - padding-box - (default) the background image starts from the upper left corner of the padding edge
  - content-box - the background image starts from the upper left corner of the content
- The following example illustrates the background-origin property:

Example

```
#example1 {
  border: 10px solid black;
  padding: 35px;
  background: url(img_flwr.gif);
  background-repeat: no-repeat;
  background-origin: content-box;
}
```

CSS background-clip Property

- The CSS background-clip property specifies the painting area of the background.
- The property takes three different values:
  - border-box - (default) the background is painted to the outside edge of the border
  - padding-box - the background is painted to the outside edge of the padding
  - content-box - the background is painted within the content box
- The following example illustrates the background-clip property:

Example

```
#example1 {
  border: 10px dotted black;
  padding: 35px;
  background: yellow;
  background-clip: content-box;
}
```

1.21 CSS COLORS

CSS supports 140+ color names, HEX values, RGB values, RGBA values, HSL values, HSLA values, and opacity.

1.21.1 RGBA Colors

- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
rgba(255, 0, 0, 0.2);
rgba(255, 0, 0, 0.4);
rgba(255, 0, 0, 0.6);
rgba(255, 0, 0, 0.8);
```

The following example defines different RGBA colors:

Example

#p1 {background-color: rgba(255, 0, 0, 0.3);} /* red with opacity */
#p2 {background-color: rgba(0, 255, 0, 0.3);} /* green with opacity */
#p3 {background-color: rgba(0, 0, 255, 0.3);} /* blue with opacity */

1.21.2 HSL Colors

- HSL stands for Hue, Saturation and Lightness.
- An HSL color value is specified with: hsl(hue, saturation, lightness).
- Hue is a degree on the color wheel (from 0 to 360):
    - 0 (or 360) is red
    - 120 is green
    - 240 is blue
    - Saturation is a percentage value: 100% is the full color.
    - Lightness is also a percentage; 0% is dark (black) and 100% is white.

Hsl(0, 100%, 30%);
hsl(0, 100%, 50%);
hsl(0, 100%, 70%);
hsl(0, 100%, 90%);
The following example defines different HSL colors:

Example

#p1 {background-color: hsl(120, 100%, 50%);} /* green */
#p2 {background-color: hsl(120, 100%, 75%);} /* light green */
#p3 {background-color: hsl(120, 100%, 25%);} /* dark green */
#p4 {background-color: hsl(120, 60%, 70%);}   /* pastel green */

1.21.3 HSLA Colors

- HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.
- An HSLA color value is specified with: hsla(hue, saturation, lightness, alpha), where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

  hsla(0, 100%, 30%, 0.3);
  hsla(0, 100%, 50%, 0.3);
  hsla(0, 100%, 70%, 0.3);
  hsla(0, 100%, 90%, 0.3);
- The following example defines different HSLA colors:

Example

#p1 {background-color: hsla(120, 100%, 50%, 0.3);} /* green with opacity */
#p2 {background-color: hsla(120, 100%, 75%, 0.3);} /* light green with opacity */
#p3 {background-color: hsla(120, 100%, 25%, 0.3);} /* dark green with opacity */
#p4 {background-color: hsla(120, 60%, 70%, 0.3);}   /* pastel green with opacity */

1.21.4 Opacity

- The CSS opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent).
- The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Example

#p1 {background-color:rgb(255,0,0);opacity:0.6;} /* red with opacity */
#p2 {background-color:rgb(0,255,0);opacity:0.6;} /* green with opacity */
#p3 {background-color:rgb(0,0,255);opacity:0.6;} /* blue with opacity */

1.22 CSS SHADOW EFFECTS

CSS Shadow Effects

With CSS you can add shadow to text and to elements.

- text-shadow
- box-shadow

1.22.1 CSS Text Shadow

- The CSS text-shadow property applies shadow to text.
- In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Text shadow effect!

Example

```
h1 {
  text-shadow: 2px 2px;
}
```

Text shadow effect!

Example

```
h1 {
  text-shadow: 2px 2px red;
}
```

Multiple Shadows

- To add more than one shadow to the text, you can add a comma-separated list of shadows.
- The following example shows a red and blue neon glow shadow:

Text shadow effect!

Example

```
h1 {
  text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
}
```

CSS box-shadow Property

- The CSS box-shadow property applies shadow to elements.
- In its simplest use, you only specify the horizontal shadow and the vertical shadow:
- This is a yellow <div> element with a black box-shadow

Example

```
div {
  box-shadow: 10px 10px;
}
```

- Next, add a color to the shadow:
- This is a yellow <div> element with a grey box-shadow

Example

```
div {
  box-shadow: 10px 10px grey;
}
```

- Next, add a blur effect to the shadow:
- This is a yellow <div> element with a blurred, grey box-shadow

Example

```
div {
  box-shadow: 10px 10px 5px grey;
}
```

Example

```
#boxshadow {
  position: relative;
  box-shadow: 1px 2px 4px rgba(0, 0, 0, .5);
  padding: 10px;
  background: white;
}
#boxshadow img {
  width: 100%;
  border: 1px solid #8a4419;
  border-style: inset;
}
#boxshadow::after {
  content: '';
  position: absolute;
  z-index: -1; /* hide shadow behind image */
  box-shadow: 0 15px 20px rgba(0, 0, 0, 0.3);
  width: 70%;
  left: 15%; /* one half of the remaining 30% */
  height: 100px;
  bottom: 0;
}
```

1.23 CSS TEXT EFFECTS

CSS Text Overflow, Word Wrap, Line Breaking Rules, and Writing Modes

- text-overflow
- word-wrap
- word-break
- writing-mode

1.23.1 CSS Text Overflow

- The CSS text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.It can be clipped:
- This is some long text that will not fit in the box or it can be rendered as an ellipsis (...):This is some long text that will not fit in the box.The CSS code is as follows:

Example

```
p.test1 {
  white-space: nowrap;
  width: 200px;
  border: 1px solid #000000;
  overflow: hidden;
  text-overflow: clip;
}
p.test2 {
    white-space: nowrap;
    width: 200px;
```

```
   border: 1px solid #000000;
   overflow: hidden;
   text-overflow: ellipsis;
}
```

The following example shows how you can display the overflowed content when hovering over the element:

Example

```
div.test:hover {
  overflow: visible;
}
```

CSS Word Wrapping

- The CSS word-wrap property allows long words to be able to be broken and wrap onto the next line.
- If a word is too long to fit within an area, it expands outside:
- This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.
- The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:
- This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

The CSS code is as follows:

Example

Allow long words to be able to be broken and wrap onto the next line:

```
p {
  word-wrap: break-word;
}
```

CSS Word Breaking

- The CSS word-break property specifies line breaking rules.
- This paragraph contains some text. This line will-break-at-hyphens.
- This paragraph contains some text. The lines will break at any character.

The CSS code is as follows:

Example

```
p.test1 {
  word-break: keep-all;
}

p.test2 {
  word-break: break-all;
}
```

CSS Writing Mode

- The CSS writing-mode property specifies whether lines of text are laid out horizontally or vertically.
- Some text with a span element with a vertical-rl writing-mode.
- The following example shows some different writing modes:

Example

```
p.test1 {
  writing-mode: horizontal-tb;
}

span.test2 {
  writing-mode: vertical-rl;
}

p.test2 {
  writing-mode: vertical-rl;
}
```

CSS Text Effect Properties

The following table lists the CSS text effect properties:

Property	**Description**
text-align-last	Specifies how to align the last line of a text
text-justify	Specifies how justified text should be aligned and spaced
text-overflow	Specifies how overflowed content that is not displayed should be signaled to the user
word-break	Specifies line breaking rules for non-CJK scripts
word-wrap	Allows long words to be able to be broken and wrap onto the next line
writing-mode	Specifies whether lines of text are laid out horizontally or vertically

1.24 CSS 2D TRANSFORMS

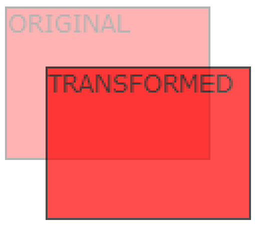
- CSS transforms allow you to move, rotate, scale, and skew elements.Mouse over the element below to see a 2D transformation:

CSS 2D Transforms Methods

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

1.24.1 The translate() Method



- The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).
- The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:

Example

```
div {
  transform: translate(50px, 100px);
}
```

1.24.2 The rotate() Method

- The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.
- The following example rotates the <div> element clockwise with 20 degrees:

Example

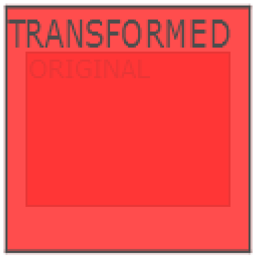
```
div {
  transform: rotate(20deg);
}
```

Using negative values will rotate the element counter-clockwise.
The following example rotates the <div> element counter-clockwise with 20 degrees:

Example

```
div {
  transform: rotate(-20deg);
}
```

1.24.3 The scale() Method



- The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).
- The following example increases the <div> element to be two times of its original width, and three times of its original height:

Example

```
div {
  transform: scale(2, 3);
}
```

The following example decreases the <div> element to be half of its original width and height:

Example

```
div {
  transform: scale(0.5, 0.5);
}
```

1.24.4 The scaleX() Method

- The scaleX() method increases or decreases the width of an element.
- The following example increases the <div> element to be two times of its original width:

Example

```
div {
  transform: scaleX(2);
}
```

The following example decreases the <div> element to be half of its original width:

Example

```
div {
  transform: scaleX(0.5);
}
```

1.24.5 The scaleY() Method

- The scaleY() method increases or decreases the height of an element.
- The following example increases the <div> element to be three times of its original height:

Example

```
div {
  transform: scaleY(3);
}
```

The following example decreases the <div> element to be half of its original height:

Example

```
div {
  transform: scaleY(0.5);
}
```

1.24.6 The skewX() Method

- The skewX() method skews an element along the X-axis by the given angle.
- The following example skews the <div> element 20 degrees along the X-axis:

Example

```
div {
  transform: skewX(20deg);
}
```

1.24.7 The skewY() Method

- The skewY() method skews an element along the Y-axis by the given angle.
- The following example skews the <div> element 20 degrees along the Y-axis:

Example

```
div {
  transform: skewY(20deg);
}
```

1.24.8 The skew() Method

- The skew() method skews an element along the X and Y-axis by the given angles.
- The following example skews the <div> element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

Example

```
div {
  transform: skew(20deg, 10deg);
}
```

- If the second parameter is not specified, it has a zero value. So, the following example skews the <div> element 20 degrees along the X-axis:

ORIGINAL
TRANSFORMED

Example

```
div {
  transform: skew(20deg);
}
```

1.24.9 The matrix() Method

- The matrix() method combines all the 2D transform methods into one.
- The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.
- The parameters are as follow: matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

Example

```
div {
  transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```

CSS Transform Properties

The following table lists all the 2D transform properties:

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements

CSS 2D Transform Methods

Function	Description
matrix(n,n,n,n,n,n)	Defines a 2D transformation, using a matrix of six values
translate(x,y)	Defines a 2D translation, moving the element along the X- and the Y-axis
translateX(n)	Defines a 2D translation, moving the element along the X-axis
translateY(n)	Defines a 2D translation, moving the element along the Y-axis
scale(x,y)	Defines a 2D scale transformation, changing the elements width and height
scaleX(n)	Defines a 2D scale transformation, changing the element's width
scaleY(n)	Defines a 2D scale transformation, changing the element's height
rotate(angle)	Defines a 2D rotation, the angle is specified in the parameter
skew(x-angle,y-angle)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(angle)	Defines a 2D skew transformation along the X-axis
skewY(angle)	Defines a 2D skew transformation along the Y-axis

1.25 CSS TRANSITIONS

- CSS transitions allows you to change property values smoothly, over a given duration.
- Mouse over the element below to see a CSS transition effect:

CSS

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

Browser Support for Transitions

The numbers in the table specify the first browser version that fully supports the property.

Property					
transition	26.0	10.0	16.0	6.1	12.1
transition-delay	26.0	10.0	16.0	6.1	12.1
transition-duration	26.0	10.0	16.0	6.1	12.1
transition-property	26.0	10.0	16.0	6.1	12.1

transition-timing-function	26.0	10.0	16.0	6.1	12.1

Browser Specific Prefixes

Some older browsers need specific prefixes (-webkit-) to understand the transition properties:

Example

```
div {
  width: 100px;
  height: 100px;
  background: red;
  -webkit-transition: width 2s; /* Safari prior 6.1 */
  transition: width 2s;
}
```

How to Use CSS Transitions?

To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

The following example shows a 100px * 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

Example

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
```

- The transition effect will start when the specified CSS property (width) changes value.

Now, let us specify a new value for the width property when a user mouses over the <div> element:

Example

```
div:hover {
  width: 300px;
}
```

Notice that when the cursor mouses out of the element, it will gradually change back to its original style.

Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

Example

```
div {
  transition: width 2s, height 4s;
}
```

Specify the Speed Curve of the Transition

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end
- cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

The following example shows the some of the different speed curves that can be used:

Example

```
#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}
```

Delay the Transition Effect

The transition-delay property specifies a delay (in seconds) for the transition effect.
The following example has a 1 second delay before starting:

Example

```
div {
  transition-delay: 1s;
}
```

Transition + Transformation

The following example adds a transition effect to the transformation:

Example

```
div {
  transition: width 2s, height 2s, transform 2s;
}
```

1.26 CSS ANIMATIONS

- CSS allows animation of HTML elements without using JavaScript or Flash!

CSS

- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

Browser Support for Animations

The numbers in the table specify the first browser version that fully supports the property.

Property					
@keyframes	43.0	10.0	16.0	9.0	30.0
animation-name	43.0	10.0	16.0	9.0	30.0
animation-duration	43.0	10.0	16.0	9.0	30.0
animation-delay	43.0	10.0	16.0	9.0	30.0
animation-iteration-count	43.0	10.0	16.0	9.0	30.0
animation-direction	43.0	10.0	16.0	9.0	30.0
animation-timing-function	43.0	10.0	16.0	9.0	30.0
animation-fill-mode	43.0	10.0	16.0	9.0	30.0
animation	43.0	10.0	16.0	9.0	30.0

What are CSS Animations?

- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

1.26.1 The @keyframes Rule

- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.
- To get an animation to work, you must bind the animation to an element.
- The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

Example

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

- **Note:** The animation-duration property defines how long time an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).
- In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

- It is also possible to use percent. By using percent, you can add as many style changes as you like.
- The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Example

```
/* The animation code */
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

- The following example will change both the background-color and the position of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	**Description**
@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties
animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long time an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies the name of the @keyframes animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of the animation