The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (ie. internet) since 1990. HTTP is a generic and stateless protocol which can be used for other purposes as well using extension of its request methods, error codes and headers.

Basically, HTTP is an TCP/IP based communication protocol, which is used to deliver data (HTML files, image files, query results etc) on the World Wide Web. The default port is TCP 80, but other ports can be used. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients request data will be constructed and sent to the serve, and how servers respond to these requests.

Basic Features

There are following three basic features which makes HTTP a simple but powerful protocol:

* **HTTP is connectionless:** The HTTP client ie. browser initiates an HTTP request and after a request is made, the client disconnects from the server and waits for a response. The server process the request and re-establish the connection with the client to send response back.
* **HTTP is media independent:** This means, any type of data can be sent by HTTP as long as both the client and server know how to handle the data content. This is required for client as well as server to specify the content type using appropriate MIME-type.
* **HTTP is stateless:** As mentioned above, HTTP is a connectionless and this is a direct result that HTTP is a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different request across the web pages.

HTTP/1.0 uses a new connection for each request/response exchange where as HTTP/1.1 connection may be used for one or more request/response exchanges.

Basic Architecture

Following diagram shows a very basic architecture of a web application and depicts where HTTP sits:



The HTTP protocol is a request/response protocol based on client/server based architecture where web browser, robots and search engines, etc. act like HTTP clients and Web server acts as server.

Client

The HTTP client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a TCP/IP connection.

Server

The HTTP server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content.

HTTP - Parameters

This chapter is going to list down few of the important HTTP Protocol Parameters and their syntax in a way they are used in the communication. For example, format for date, format of URL etc. This will help you in constructing your request and response messages while writing HTTP client or server programs. You will see complete usage of these parameters in subsequent chapters while explaining message structure for HTTP requests and responses.

HTTP Version

HTTP uses a **<major>.<minor>** numbering scheme to indicate versions of the protocol. The version of an HTTP message is indicated by an HTTP-Version field in the first line. Here is the general syntax of specifying HTTP version number:

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

Example

HTTP/1.0

or

HTTP/1.1

AD

Uniform Resource Identifiers (URI)

Uniform Resource Identifiers (URI) is simply formatted, case-insensitive string containing name, location etc to identify a resource, for example a website, a web service etc. A general syntax of URI used for HTTP is as follows:

URI = "http:" "//" host [ ":" port ] [ abs\_path [ "?" query ]]

Here if the **port** is empty or not given, port 80 is assumed for HTTP and an empty **abs\_path** is equivalent to an **abs\_path** of "/". The characters other than those in the **reserved** and **unsafe** sets are equivalent to their ""%" HEX HEX" encoding.

Example

Following two URIs are equivalent:

http://abc.com:80/~smith/home.html

http://ABC.com/%7Esmith/home.html

http://ABC.com:/%7esmith/home.html

Date/Time Formats

All HTTP date/time stamps MUST be represented in Greenwich Mean Time (GMT), without exception. HTTP applications are allowed to use any of the following three representations of date/time stamps:

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

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

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

Character Sets

You use character set to specify the character sets that the client prefers. Multiple character sets can be listed separated by commas. If a value is not specified, the default is US-ASCII.

Example

Following are valid character sets:

US-ASCII

or

ISO-8859-1

or

ISO-8859-7

Content Encodings

A content ecoding values indicate an encoding algorithm has been used to encode the content before passing it over the network. Content codings are primarily used to allow a document to be compressed or otherwise usefully transformed without losing the identity.

All content-coding values are case-insensitive. HTTP/1.1 uses content-coding values in the Accept-Encoding and Content-Encoding header fields which we will see in subsequent chapters.

Example

Following are valid encoding schemes:

Accept-encoding: gzip

or

Accept-encoding: compress

or

Accept-encoding: deflate

Media Types

HTTP uses Internet Media Types in the **Content-Type** and **Accept** header fields in order to provide open and extensible data typing and type negotiation. All the Media-type values are registered with the Internet Assigned Number Authority ((IANA). Following is a general syntax to specify media type:

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

The type, subtype, and parameter attribute names are case- insensitive.

Example

Accept: image/gif

Language Tags

HTTP uses language tags within the **Accept-Language** and **Content-Language** fields. A language tag is composed of 1 or more parts: A primary language tag and a possibly empty series of subtags:

language-tag = primary-tag \*( "-" subtag )

White space is not allowed within the tag and all tags are case- insensitive.

Example

Example tags include:

en, en-US, en-cockney, i-cherokee, x-pig-latin

Where any two-letter primary-tag is an ISO-639 language abbreviation and any two-letter initial subtag is an ISO-3166 country code.

HTTP - Messages

HTTP is based on client-server architecture model and a stateless request/response protocol that operates by exchanging messages across a reliable TCP/IP connection.

An HTTP "client" is a program (Web browser or any other client) that establishes a connection to a server for the purpose of sending one or more HTTP request messages. An HTTP "server" is a program ( generally a web server like Apache Web Server or Internet Information Services IIS etc. ) that accepts connections in order to serve HTTP requests by sending HTTP response messages.

HTTP makes use of the Uniform Resource Identifier (URI) to identify a given resource and to establish a connection. Once connection is established, **HTTP messages** are passed in a format similar to that used by Internet mail [RFC5322] and the Multipurpose Internet Mail Extensions (MIME) [RFC2045]. These messages are consisted of **requests** from client to server and **responses** from server to client which will have following format:

HTTP-message = <Request> | <Response> ; HTTP/1.1 messages

HTTP request and HTTP response use a generic message format of RFC 822 for transferring the required data. This generic message format consists of following four items.

* A Start-line
* Zero or more header fields followed by CRLF
* An empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields
* Optionally a message-body

Following section will explain each of the entities used in HTTP message.

Message Start-Line

A start-line will have following generic syntax:

start-line = Request-Line | Status-Line

We will discuss Request-Line and Status-Line while discussing HTTP Request and HTTP Response messages respectively. For now let's see the examples of start line in case of request and response:

GET /hello.htm HTTP/1.1 (This is Request-Line sent by the client)

HTTP/1.1 200 OK (This is Status-Line sent by the server)

Header Fields

HTTP deader fields provide required information about the request or response, or about the object sent in the message body. There are following four types of HTTP message headers:

* **General-header:** These header fields have general applicability for both request and response messages.
* **Request-header:** These header fields are applicability only for request messages.
* **Response-header:** These header fields are applicability only for response messages.
* **Entity-header:** These header fields define metainformation about the entity-body or, if no body is present

All the above mentioned headers follow the same generic format and each of the header field consists of a name followed by a colon (**:**) and the field value as follows:

message-header = field-name ":" [ field-value ]

Following are the examples of various header fields:

User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3

Host: www.example.com

Accept-Language: en, mi

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

ETag: "34aa387-d-1568eb00"

Accept-Ranges: bytes

Content-Length: 51

Vary: Accept-Encoding

Content-Type: text/plain

Message Body

The message body part is optional for an HTTP message but if it is available then it is used to carry the entity-body associated with the request or response. If entity body is associated then usually **Content-Type** and **Content-Length** headers lines specify the nature of the body associated.

A message body is the one which carries actual HTTP request data (including form data and uploaded etc.) and HTTP response data from the server ( including files, images etc). Following is a simple content of a message body:

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

HTTP - Requests

An HTTP client sends an HTTP request to a server in the form of a request message which includes following format:

* A Request-line
* Zero or more header (General|Request|Entity) fields followed by CRLF
* An empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields
* Optionally a message-body

Following section will explain each of the entities used in HTTP message.

Message Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by space SP characters.

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

Let's discuss each of the part mentioned in Request-Line.

Request Method

The request **Method** indicates the method to be performed on the resource identified by the given **Request-URI**. The method is case-sensitive ans should always be mentioned uppercase. Following are supported methods in HTTP/1.1

|  |  |
| --- | --- |
| **S.N.** | **Method and Description** |
| 1 | **GET** The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. |
| 2 | **HEAD** Same as GET, but only transfer the status line and header section. |
| 3 | **POST** A POST request is used to send data to the server, for example customer information, file upload etc using HTML forms. |
| 4 | **PUT** Replace all current representations of the target resource with the uploaded content. |
| 5 | **DELETE** Remove all current representations of the target resource given by URI. |
| 6 | **CONNECT** Establish a tunnel to the server identified by a given URI. |
| 7 | **OPTIONS** Describe the communication options for the target resource. |
| 8 | **TRACE** Perform a message loop-back test along the path to the target resource. |

Request-URI

The Request-URI is a Uniform Resource Identifier and identifies the resource upon which to apply the request. Following are the most commonly used forms to specify an URI:

Request-URI = "\*" | absoluteURI | abs\_path | authority

|  |  |
| --- | --- |
| **S.N.** | **Method and Description** |
| 1 | The asterisk **\*** is used when HTTP request does not apply to a particular resource, but to the server itself, and is only allowed when the method used does not necessarily apply to a resource. For example:  **OPTIONS \* HTTP/1.1** |
| 2 | The **absoluteURI** is used when HTTP request is being made to a proxy. The proxy is requested to forward the request or service it from a valid cache, and return the response. For example:  **GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1** |
| 3 | The most common form of Request-URI is that used to identify a resource on an origin server or gateway. For example, a client wishing to retrieve the resource above directly from the origin server would create a TCP connection to port 80 of the host "www.w3.org" and send the lines:  **GET /pub/WWW/TheProject.html HTTP/1.1 Host: www.w3.org**  Note that the absolute path cannot be empty; if none is present in the original URI, it MUST be given as "/" (the server root) |

Request Header Fields

We will study General-header and Entity-header in a separate chapter when we will learn HTTP header fields. For now let's check what are Request header fields.

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers and there are following important Request-header fields available which can be used based on requirement.

* Accept-Charset
* Accept-Encoding
* Accept-Language
* Authorization
* Expect
* From
* Host
* If-Match
* If-Modified-Since
* If-None-Match
* If-Range
* If-Unmodified-Since
* Max-Forwards
* Proxy-Authorization
* Range
* Referer
* TE
* User-Agent

You can introduce your custom fields in case you are going to write your own custom Client and Web Server.

Request Message Examples

Now let's put it all together to form an HTTP request to fetch **hello.htm** page from the web server running on tutorialspoint.com

GET /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Here we are not sending any request data to the server because we are fetching a plan HTML page from the server. Connection is a general-header used here and rest of the headers are request headers. Following is one more example where we send form data to the server using request message body:

POST /cgi-bin/process.cgi HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Content-Type: application/x-www-form-urlencoded

Content-Length: **length**

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

licenseID=string&content=string&/paramsXML=string

Here given URL */cgi-bin/process.cgi* will be used to process the passed data and accordingly a response will be retuned. Here **content-type** tells the server that passed data is simple web form data and **length** will be actual length of the data put in the message body. Following example shows how you can pass plan XML to your web server:

POST /cgi-bin/process.cgi HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Content-Type: text/xml; charset=utf-8

Content-Length: **length**

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>

<string xmlns="http://clearforest.com/">string</string>

HTTP - Responses

After receiving and interpreting a request message, a server responds with an HTTP response message:

* A Status-line
* Zero or more header (General|Response|Entity) fields followed by CRLF
* An empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields
* Optionally a message-body

Following section will explain each of the entities used in HTTP message.

Message Status-Line

The Status-Line consisting of the protocol version followed by a numeric status code and its associated textual phrase. The elements are separated by space SP characters.

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

Let's discuss each of the part mentioned in Status-Line.

HTTP Version

A server supporting HTTP version 1.1 will return following version information:

HTTP-Version = HTTP/1.1

Status Code

The Status-Code element is a 3-digit integer where first digit of the Status-Code defines the class of response and the last two digits do not have any categorization role. There are 5 values for the first digit:

|  |  |
| --- | --- |
| **S.N.** | **Code and Description** |
| 1 | **1xx: Informational** This means request received and continuing process. |
| 2 | **2xx: Success** This means the action was successfully received, understood, and accepted. |
| 3 | **3xx: Redirection** This means further action must be taken in order to complete the request. |
| 4 | **4xx: Client Error** This means the request contains bad syntax or cannot be fulfilled |
| 5 | **5xx: Server Error** The server failed to fulfill an apparently valid request |

HTTP status codes are extensible and HTTP applications are not required to understand the meaning of all registered status codes. A list of all the status code has been given in a separate chapter for you reference.

Response Header Fields

We will study General-header and Entity-header in a separate chapter when we will learn HTTP header fields. For now let's check what are Response header fields.

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

* Accept-Ranges
* Age
* ETag
* Location
* Proxy-Authenticate
* Retry-After
* Server
* Vary
* WWW-Authenticate

You can introduce your custom fields in case you are going to write your own custom Web Client and Server.

Response Message Examples

Now let's put it all together to form an HTTP response for a request to fetch **hello.htm** page from the web server running on tutorialspoint.com

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

Content-Length: 88

Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

Following is an example of HTTP response message showing error condition when web server could not find requested page:

HTTP/1.1 404 Not Found

Date: Sun, 18 Oct 2012 10:36:20 GMT

Server: Apache/2.2.14 (Win32)

Content-Length: 230

Connection: Closed

Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">

<html>

<head>

<title>404 Not Found</title>

</head>

<body>

<h1>Not Found</h1>

<p>The requested URL /t.html was not found on this server.</p>

</body>

</html>

Following is an example of HTTP response message showing error condition when web server encountered a wrong HTTP version in given HTTP request:

HTTP/1.1 400 Bad Request

Date: Sun, 18 Oct 2012 10:36:20 GMT

Server: Apache/2.2.14 (Win32)

Content-Length: 230

Content-Type: text/html; charset=iso-8859-1

Connection: Closed

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">

<html>

<head>

<title>400 Bad Request</title>

</head>

<body>

<h1>Bad Request</h1>

<p>Your browser sent a request that this server could not understand.<p>

<p>The request line contained invalid characters following the protocol string.<p>

</body>

</html>

HTTP - Methods

The set of common methods for HTTP/1.1 is defined below and this set can be expanded based on requirement. These method names are case sensitive and they must be used in uppercase.

|  |  |
| --- | --- |
| **S.N.** | **Method and Description** |
| 1 | **GET** The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. |
| 2 | **HEAD** Same as GET, but only transfer the status line and header section. |
| 3 | **POST** A POST request is used to send data to the server, for example customer information, file upload etc using HTML forms. |
| 4 | **PUT** Replace all current representations of the target resource with the uploaded content. |
| 5 | **DELETE** Remove all current representations of the target resource given by URI. |
| 6 | **CONNECT** Establish a tunnel to the server identified by a given URI. |
| 7 | **OPTIONS** Describe the communication options for the target resource. |
| 8 | **TRACE** Perform a message loop-back test along the path to the target resource. |

GET Method

A GET request retrieves data from a web server by specifying parameters in the URL portion of the request. This is the main method used for document retrieval. Following is a simple example which makes use of GET method to fetch hello.htm:

GET /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

The server response against the above GET request will be as follows:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

ETag: "34aa387-d-1568eb00"

Vary: Authorization,Accept

Accept-Ranges: bytes

Content-Length: 88

Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

HEAD Method

The HEAD method is functionally like GET, except that the server replies with a response line and headers, but no entity-body. Following is a simple example which makes use of HEAD method to fetch header information about hello.htm:

HEAD /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Following will be a server response against the above GET request:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

ETag: "34aa387-d-1568eb00"

Vary: Authorization,Accept

Accept-Ranges: bytes

Content-Length: 88

Content-Type: text/html

Connection: Closed

You can notice that here server does not send any data after header.

POST Method

The POST method is used when you want to send some data to the server, for example file update, form data etc. Following is a simple example which makes use of POST method to send a form data to the server which will be processed by a process.cgi and finally a response will be returned:

POST /cgi-bin/process.cgi HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Content-Type: text/xml; charset=utf-8

Content-Length: 88

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>

<string xmlns="http://clearforest.com/">string</string>

Server side script process.cgi process the passed data and send following response:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

ETag: "34aa387-d-1568eb00"

Vary: Authorization,Accept

Accept-Ranges: bytes

Content-Length: 88

Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Request Processed Successfully</h1>

</body>

</html>

PUT Method

The PUT method is used to request the server to store the included entity-body at a location specified by the given URL. The following example request server to save the given entity-body in **hello.htm** at the root of the server:

PUT /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Connection: Keep-Alive

Content-type: text/html

Content-Length: 182

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

The server will store given entity-body in **hello.htm** file and will send following response back to the client:

HTTP/1.1 201 Created

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Content-type: text/html

Content-length: 30

Connection: Closed

<html>

<body>

<h1>The file was created.</h1>

</body>

</html>

DELETE Method

The DELETE method is used to request the server to delete file at a location specified by the given URL. The following example request server to delete the given file **hello.htm** at the root of the server:

DELETE /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Connection: Keep-Alive

The server will delete mentioned file **hello.htm** and will send following response back to the client:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Content-type: text/html

Content-length: 30

Connection: Closed

<html>

<body>

<h1>URL deleted.</h1>

</body>

</html>

CONNECT Method

The CONNECT method is used by the client to establish a network connection to a web server over HTTP. The following example request a connection with a web server running on host tutorialspoint.com:

CONNECT www.tutorialspoint.com HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

The connection is established with the server and following response is sent back to the client:

HTTP/1.1 200 Connection established

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

OPTIONS Method

The OPTIONS method is used by the client to find out what are the HTTP methods and other options supported by a web server. The client can specify a URL for the OPTIONS method, or an asterisk (\*) to refer to the entire server. The following example request a list of methods supported by a web server running on tutorialspoint.com:

OPTIONS \* HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

The server will send information based on the current configuration of the server, for example:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Allow: GET,HEAD,POST,OPTIONS,TRACE

Content-Type: httpd/unix-directory

TRACE Method

The TRACE method is used to eacho the contents of an HTTP Request back to the requester which can be used for debugging purpose at the time of development. The following example shows the usage of TRACE method:

TRACE / HTTP/1.1

Host: www.tutorialspoint.com

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

The server will send following message in response of the above request:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Content-Type: message/http

Content-Length: 39

Connection: Closed

TRACE / HTTP/1.1

Host: www.tutorialspoint.com

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

HTTP - Status Codes

The Status-Code element in a server response, is a 3-digit integer where first digit of the Status-Code defines the class of response and the last two digits do not have any categorization role. There are 5 values for the first digit:

|  |  |
| --- | --- |
| **S.N.** | **Code and Description** |
| 1 | **1xx: Informational** This means request received and continuing process. |
| 2 | **2xx: Success** This means the action was successfully received, understood, and accepted. |
| 3 | **3xx: Redirection** This means further action must be taken in order to complete the request. |
| 4 | **4xx: Client Error** This means the request contains bad syntax or cannot be fulfilled |
| 5 | **5xx: Server Error** The server failed to fulfill an apparently valid request |

HTTP status codes are extensible and HTTP applications are not required to understand the meaning of all registered status codes. Following is a list of all the status code.

1xx: Information

|  |  |
| --- | --- |
| **Message:** | **Description:** |
| 100 Continue | Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request |
| 101 Switching Protocols | The server switches protocol |

2xx: Successful

|  |  |
| --- | --- |
| **Message:** | **Description:** |
| 200 OK | The request is OK |
| 201 Created | The request is complete, and a new resource is created |
| 202 Accepted | The request is accepted for processing, but the processing is not complete |
| 203 Non-authoritative Information | The information in the entity header is from a local or third-party copy, not from the original server. |
| 204 No Content | A status code and header are given in the response, but there is no entity-body in the reply. |
| 205 Reset Content | The browser should clear the form used for this transaction for additional input. |
| 206 Partial Content | The server is returning partial data of the size requested. Used in response to a request specifying a *Range* header. The server must specify the range included in the response with the *Content-Range* header. |

3xx: Redirection

|  |  |
| --- | --- |
| **Message:** | **Description:** |
| 300 Multiple Choices | A link list. The user can select a link and go to that location. Maximum five addresses |
| 301 Moved Permanently | The requested page has moved to a new url |
| 302 Found | The requested page has moved temporarily to a new url |
| 303 See Other | The requested page can be found under a different url |
| 304 Not Modified | This is the response code to an *If-Modified-Since* or *If-None-Match* header, where the URL has not been modified since the specified date. |
| 305 Use Proxy | The requested URL must be accessed through the proxy mentioned in the *Location* header. |
| 306 *Unused* | This code was used in a previous version. It is no longer used, but the code is reserved |
| 307 Temporary Redirect | The requested page has moved temporarily to a new url |

4xx: Client Error

|  |  |
| --- | --- |
| **Message:** | **Description:** |
| 400 Bad Request | The server did not understand the request |
| 401 Unauthorized | The requested page needs a username and a password |
| 402 Payment Required | *You can not use this code yet* |
| 403 Forbidden | Access is forbidden to the requested page |
| 404 Not Found | The server can not find the requested page |
| 405 Method Not Allowed | The method specified in the request is not allowed |
| 406 Not Acceptable | The server can only generate a response that is not accepted by the client |
| 407 Proxy Authentication Required | You must authenticate with a proxy server before this request can be served |
| 408 Request Timeout | The request took longer than the server was prepared to wait |
| 409 Conflict | The request could not be completed because of a conflict |
| 410 Gone | The requested page is no longer available |
| 411 Length Required | The "Content-Length" is not defined. The server will not accept the request without it |
| 412 Precondition Failed | The precondition given in the request evaluated to false by the server |
| 413 Request Entity Too Large | The server will not accept the request, because the request entity is too large |
| 414 Request-url Too Long | The server will not accept the request, because the url is too long. Occurs when you convert a "post" request to a "get" request with a long query information |
| 415 Unsupported Media Type | The server will not accept the request, because the media type is not supported |
| 416 Requested Range Not Satisfiable | The requested byte range is not available and is out of bounds. |
| 417 Expectation Failed | The expectation given in an Expect request-header field could not be met by this server. |

5xx: Server Error

|  |  |
| --- | --- |
| **Message:** | **Description:** |
| 500 Internal Server Error | The request was not completed. The server met an unexpected condition |
| 501 Not Implemented | The request was not completed. The server did not support the functionality required |
| 502 Bad Gateway | The request was not completed. The server received an invalid response from the upstream server |
| 503 Service Unavailable | The request was not completed. The server is temporarily overloading or down |
| 504 Gateway Timeout | The gateway has timed out |
| 505 HTTP Version Not Supported | The server does not support the "http protocol" version |

HTTP - Header Fields

HTTP deader fields provide required information about the request or response, or about the object sent in the message body. There are following four types of HTTP message headers:

* **General-header:** These header fields have general applicability for both request and response messages.
* **Client Request-header:** These header fields are applicability only for request messages.
* **Server Response-header:** These header fields are applicability only for response messages.
* **Entity-header:** These header fields define metainformation about the entity-body or, if no body is present

General Headers

Cache-control

The Cache-Control general-header field is used to specify directives that MUST be obeyed by all caching system. Following is the syntax:

Cache-Control : cache-request-directive|cache-response-directive

An HTTP clients or servers can use the **Cache-control** general header to specify parameters for the cache or to request certain kinds of documents from the cache. The caching directives are specified in a comma-separated list. For example:

Cache-control: no-cache

There are following important cache request directives which can be used by the client in its HTTP request:

|  |  |
| --- | --- |
| **S.N.** | **Cache Request Directive and Description** |
| 1 | **no-cache** A cache must not use the response to satisfy a subsequent request without successful revalidation with the origin server. |
| 2 | **no-store** The cache should not store anything about the client request or server response. |
| 3 | **max-age = seconds** Indicates that the client is willing to accept a response whose age is no greater than the specified time in seconds. |
| 4 | **max-stale [ = seconds ]** Indicates that the client is willing to accept a response that has exceeded its expiration time. If seconds are given, it must not be expired by more than that time. |
| 5 | **min-fresh = seconds** Indicates that the client is willing to accept a response whose freshness lifetime is no less than its current age plus the specified time in seconds. |
| 6 | **no-transform** Do not convert the entity-body. |
| 7 | **only-if-cached** Do not retrieve new data. The cache can send a document only if it is in the cache, and should not contact the origin-server to see if a newer copy exists. |

There are following important cache response directives which can be used by the server in its HTTP response:

|  |  |
| --- | --- |
| **S.N.** | **Cache Request Directive and Description** |
| 1 | **public** Indicates that the response may be cached by any cache. |
| 2 | **private** Indicates that all or part of the response message is intended for a single user and must not be cached by a shared cache. |
| 3 | **no-cache** A cache must not use the response to satisfy a subsequent request without successful revalidation with the origin server. |
| 4 | **no-store** The cache should not store anything about the client request or server response. |
| 5 | **no-transform** Do not convert the entity-body. |
| 6 | **must-revalidate** The cache must verify the status of stale documents before using it and expired one should not be used. |
| 7 | **proxy-revalidate** The proxy-revalidate directive has the same meaning as the must- revalidate directive, except that it does not apply to non-shared user agent caches. |
| 8 | **max-age = seconds** Indicates that the client is willing to accept a response whose age is no greater than the specified time in seconds. |
| 9 | **s-maxage = seconds** The maximum age specified by this directive overrides the maximum age specified by either the max-age directive or the Expires header. The s-maxage directive is always ignored by a private cache. |

Connection

The Connection general-header field allows the sender to specify options that are desired for that particular connection and must not be communicated by proxies over further connections. Following is the simple syntax of using connection header:

Connection : "Connection"

HTTP/1.1 defines the "closed" connection option for the sender to signal that the connection will be closed after completion of the response. For example:

Connection: Closed

By default, HTTP 1.1 uses persistent connections, where the connection does not automatically close after a transaction. HTTP 1.0, on the other hand, does not have persistent connections by default. If a 1.0 client wishes to use persistent connections, it uses the **keep-alive** parameter as follows:

Connection: keep-alive

Date

All HTTP date/time stamps MUST be represented in Greenwich Mean Time (GMT), without exception. HTTP applications are allowed to use any of the following three representations of date/time stamps:

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

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

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

Here first format is the most preferred one.

Pragma

The Pragma general-header field is used to include implementation- specific directives that might apply to any recipient along the request/response chain. For example:

Pragma: no-cache

The only directive defined in HTTP/1.0 is the no-cache directive and is maintained in HTTP 1.1 for backward compatibility. No new Pragma directives will be defined in the future.

Trailer

The Trailer general field value indicates that the given set of header fields is present in the trailer of a message encoded with chunked transfer-coding. Following is the syntax of Trailer header field:

Trailer : field-name

Message header fields listed in the Trailer header field must not include the following header fields:

* Transfer-Encoding
* Content-Length
* Trailer

Transfer-Encoding

The *Transfer-Encoding* general-header field indicates what type of transformation has been applied to the message body in order to safely transfer it between the sender and the recipient. This is not the same as content-encoding because transfer-encodings are a property of the message, not of the entity-body. Following is the syntax of Transfer-Encoding header field:

Transfer-Encoding: chunked

All transfer-coding values are case-insensitive.

Upgrade

The *Upgrade* general-header allows the client to specify what additional communication protocols it supports and would like to use if the server finds it appropriate to switch protocols. For example:

Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11

The Upgrade header field is intended to provide a simple mechanism for transition from HTTP/1.1 to some other, incompatible protocol

Via

The *Via* general-header must be used by gateways and proxies to indicate the intermediate protocols and recipients. For example, a request message could be sent from an HTTP/1.0 user agent to an internal proxy code-named "fred", which uses HTTP/1.1 to forward the request to a public proxy at nowhere.com, which completes the request by forwarding it to the origin server at www.ics.uci.edu. The request received by www.ics.uci.edu would then have the following Via header field:

Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)

The Upgrade header field is intended to provide a simple mechanism for transition from HTTP/1.1 to some other, incompatible protocol

Warning

The *Warning* general-header is used to carry additional information about the status or transformation of a message which might not be reflected in the message. A response may carry more than one Warning header.

Warning : warn-code SP warn-agent SP warn-text SP warn-date

Client Request Headers

Accept

The *Accept* request-header field can be used to specify certain media types which are acceptable for the response. Following is the general syntax:

Accept: type/subtype [q=qvalue]

Multiple media types can be listed separated by commas and the optional qvalue represents an acceptable quality level for accept types on a scale of 0 to 1. Following is an example:

Accept: text/plain; q=0.5, text/html, text/x-dvi; q=0.8, text/x-c

This would be interpreted as **text/html** and **text/x-c** are the preferred media types, but if they do not exist, then send the **text/x-dvi** entity, and if that does not exist, send the **text/plain** entity.

Accept-Charset

The *Accept-Charset* request-header field can be used to indicate what character sets are acceptable for the response. Following is the general syntax:

Accept-Charset: character\_set [q=qvalue]

Multiple character sets can be listed separated by commas and the optional qvalue represents an acceptable quality level for nonpreferred character sets on a scale of 0 to 1. Following is an example:

Accept-Charset: iso-8859-5, unicode-1-1; q=0.8

The special value "\*", if present in the **Accept-Charset** field, matches every character set and if no **Accept-Charset** header is present, the default is that any character set is acceptable.

Accept-Encoding

The *Accept-Encoding* request-header field is similar to Accept, but restricts the content-codings that are acceptable in the response. Following is the general syntax:

Accept-Encoding: encoding types

Following are examples:

Accept-Encoding: compress, gzip

Accept-Encoding:

Accept-Encoding: \*

Accept-Encoding: compress;q=0.5, gzip;q=1.0

Accept-Encoding: gzip;q=1.0, identity; q=0.5, \*;q=0

Accept-Language

The *Accept-Language* request-header field is similar to Accept, but restricts the set of natural languages that are preferred as a response to the request. Following is the general syntax:

Accept-Language: language [q=qvalue]

Multiple languages can be listed separated by commas and the optional qvalue represents an acceptable quality level for nonpreferred languages on a scale of 0 to 1. Following is an example:

Accept-Language: da, en-gb;q=0.8, en;q=0.7

Authorization

The *Authorization* request-header field value consists of credentials containing the authentication information of the user agent for the realm of the resource being requested. Following is the general syntax:

Authorization : credentials

The HTTP/1.0 specification defines the BASIC authorization scheme, where the authorization parameter is the string of **username:password** encoded in base 64. Following is an example:

Authorization: BASIC Z3Vlc3Q6Z3Vlc3QxMjM=

The value decodes into is **guest:guest123** where **guest** is user ID and **guest123** is the password.

Cookie

The *Cookie* request-header field value contains a name/value pair of information stored for that URL. Following is the general syntax:

Cookie: name=value

Multiple cookies can be specified separated by semicolons as follows:

Cookie: name1=value1;name2=value2;name3=value3

Expect

The *Expect* request-header field is used to indicate that particular server behaviors are required by the client. Following is the general syntax:

Expect : 100-continue | expectation-extension

If a server receives a request containing an Expect field that includes an expectation-extension that it does not support, it must respond with a 417 (Expectation Failed) status.

From

The *From* request-header field contains an Internet e-mail address for the human user who controls the requesting user agent. Following is a simple example:

From: webmaster@w3.org

This header field may be used for logging purposes and as a means for identifying the source of invalid or unwanted requests.

Host

The *Host* request-header field is used to specify the Internet host and port number of the resource being requested. Following is the general syntax:

Host : "Host" ":" host [ ":" port ] ;

A **host** without any trailing port information implies the default port, which is 80. For example, a request on the origin server for *http://www.w3.org/pub/WWW/* would be:

GET /pub/WWW/ HTTP/1.1

Host: www.w3.org

If-Match

The *If-Match* request-header field is used with a method to make it conditional. This header request the server to perform the requested method only if given value in this tag matches the given entity tags represented by **ETag**. Following is the general syntax:

If-Match : entity-tag

An asterisk (\*) matches any entity, and the transaction continues only if the entity exists. Following are possible examples:

If-Match: "xyzzy"

If-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"

If-Match: \*

If none of the entity tags match, or if "\*" is given and no current entity exists, the server must not perform the requested method, and must return a 412 (Precondition Failed) response.

If-Modified-Since

The *If-Modified-Since* request-header field is used with a method to make it conditional. If the requested URL has not been modified since the time specified in this field, an entity will not be returned from the server; instead, a 304 (not modified) response will be returned without any message-body. Following is the general syntax:

If-Modified-Since : HTTP-date

An example of the field is:

If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT

If none of the entity tags match, or if "\*" is given and no current entity exists, the server must not perform the requested method, and must return a 412 (Precondition Failed) response.

If-None-Match

The *If-None-Match* request-header field is used with a method to make it conditional. This header request the server to perform the requested method only if one of the given value in this tag matches the given entity tags represented by **ETag**. Following is the general syntax:

If-None-Match : entity-tag

An asterisk (\*) matches any entity, and the transaction continues only if the entity does not exist. Following are possible examples:

If-None-Match: "xyzzy"

If-None-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"

If-None-Match: \*

If-Range

The *If-Range* request-header field can be used with a conditional GET to request only the portion of the entity that is missing, if it has not been changed, and the entire entity if it has changed. Following is the general syntax:

If-Range : entity-tag | HTTP-date

Either an entity tag or a date can be used to identify the partial entity already received. For example:

If-Range: Sat, 29 Oct 1994 19:43:31 GMT

Here if the document has not been modified since the given date, the server returns the byte range given by the Range header otherwise, it returns all of the new document.

If-Unmodified-Since

The *If-Unmodified-Since* request-header field is used with a method to make it conditional. Following is the general syntax:

If-Unmodified-Since : HTTP-date

If the requested resource has not been modified since the time specified in this field, the server should perform the requested operation as if the If-Unmodified-Since header were not present. For example:

If-Unmodified-Since: Sat, 29 Oct 1994 19:43:31 GMT

If the request normally would result in anything other than a 2xx or 412 status, the *If-Unmodified-Since* header should be ignored.

Max-Forwards

The *Max-Forwards* request-header field provides a mechanism with the TRACE and OPTIONS methods to limit the number of proxies or gateways that can forward the request to the next inbound server. Following is the general syntax:

Max-Forwards : n

The Max-Forwards value is a decimal integer indicating the remaining number of times this request message may be forwarded. This is useful for debugging with the TRACE method, avoiding infinite loops. For example:

Max-Forwards : 5

The Max-Forwards header field may be ignored for all other methods defined in HTTP specification.

Proxy-Authorization

The *Proxy-Authorization* request-header field allows the client to identify itself (or its user) to a proxy which requires authentication. Following is the general syntax:

Proxy-Authorization : credentials

The Proxy-Authorization field value consists of credentials containing the authentication information of the user agent for the proxy and/or realm of the resource being requested.

Range

The *Range* request-header field specifies the partial range(s) of the content requested from the document. Following is the general syntax:

Range: bytes-unit=first-byte-pos "-" [last-byte-pos]

The first-byte-pos value in a byte-range-spec gives the byte-offset of the first byte in a range. The last-byte-pos value gives the byte-offset of the last byte in the range; that is, the byte positions specified are inclusive. You can specify a byte-unit as bytes Byte offsets start at zero. Following are a simple examples:

- The first 500 bytes

Range: bytes=0-499

- The second 500 bytes

Range: bytes=500-999

- The final 500 bytes

Range: bytes=-500

- The first and last bytes only

Range: bytes=0-0,-1

Multiple ranges can be listed, separated by commas. If the first digit in the comma-separated byte range(s) is missing, the range is assumed to count from the end of the document. If the second digit is missing, the range is byte n to the end of the document.

Referer

The *Referer* request-header field allows the client to specify the address (URI) of the resource from which the URL has been requested. Following is the general syntax:

Referer : absoluteURI | relativeURI

Following is a simple example:

Referer: http://www.tutorialspoint.org/http/index.htm

If the field value is a relative URI, it should be interpreted relative to the *Request-URI*.

TE

The *TE* request-header field indicates what extension *transfer-coding* it is willing to accept in the response and whether or not it is willing to accept trailer fields in a chunked *transfer-coding*. Following is the general syntax:

TE : t-codings

The presence of the keyword "trailers" indicates that the client is willing to accept trailer fields in a chunked transfer-coding and it is specified either of the ways:

TE: deflate

TE:

TE: trailers, deflate;q=0.5

If the TE field-value is empty or if no TE field is present, the only transfer-coding is *chunked*. A message with no transfer-coding is always acceptable.

User-Agent

The *User-Agent* request-header field contains information about the user agent originating the request. Following is the general syntax:

User-Agent : product | comment

Example:

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Server Response Headers

Accept-Ranges

The *Accept-Ranges* response-header field allows the server to indicate its acceptance of range requests for a resource. Following is the general syntax:

Accept-Ranges : range-unit | none

For example a server that accept byte-range requests may send

Accept-Ranges: bytes

Servers that do not accept any kind of range request for a resource may send:

Accept-Ranges: none

This will advise the client not to attempt a range request.

Age

The *Age* response-header field conveys the sender's estimate of the amount of time since the response (or its revalidation) was generated at the origin server. Following is the general syntax:

Age : delta-seconds

Age values are non-negative decimal integers, representing time in seconds. Following is a simple example:

Age: 1030

An HTTP/1.1 server that includes a cache must include an Age header field in every response generated from its own cache.

ETag

The *ETag* response-header field provides the current value of the entity tag for the requested variant. Following is the general syntax:

ETag : entity-tag

Following are simple examples:

ETag: "xyzzy"

ETag: W/"xyzzy"

ETag: ""

Location

The *Location* response-header field is used to redirect the recipient to a location other than the Request-URI for completion. Following is the general syntax:

Location : absoluteURI

Following is a simple example:

Location: http://www.tutorialspoint.org/http/index.htm

The Content-Location header field differs from Location in that the Content-Location identifies the original location of the entity enclosed in the request.

Proxy-Authenticate

The *Proxy-Authenticate* response-header field must be included as part of a 407 (Proxy Authentication Required) response. Following is the general syntax:

Proxy-Authenticate : challenge

Retry-After

The *Retry-After* response-header field can be used with a 503 (Service Unavailable) response to indicate how long the service is expected to be unavailable to the requesting client. Following is the general syntax:

Retry-After : HTTP-date | delta-seconds

Following are two simple examples:

Retry-After: Fri, 31 Dec 1999 23:59:59 GMT

Retry-After: 120

In the latter example, the delay is 2 minutes.

Server

The *Server* response-header field contains information about the software used by the origin server to handle the request. Following is the general syntax:

Server : product | comment

Following is a simple example:

Server: Apache/2.2.14 (Win32)

If the response is being forwarded through a proxy, the proxy application must not modify the Server response-header.

Set-Cookie

The *Set-Cookie* response-header field contains a name/value pair of information to retain for this URL. Following is the general syntax:

Set-Cookie: NAME=VALUE; OPTIONS

Set-Cookie response header comprises the token Set-Cookie:, followed by a comma-separated list of one or more cookies. Here are possible values you can specify as options:

|  |  |
| --- | --- |
| **S.N.** | **Options and Description** |
| 1 | **Comment=comment** This option can be used to specify any comment associated with the cookie. |
| 2 | **Domain=domain** The Domain attribute specifies the domain for which the cookie is valid. |
| 3 | **Expires=Date-time** The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser |
| 4 | **Path=path** The Path attribute specifies the subset of URLs to which this cookie applies. |
| 5 | **Secure** This instructs the user agent to return the cookie only under a secure connection. |

Following is an example of a simple cookie header generated by the server:

Set-Cookie: name1=value1,name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT

Vary

The *Vary* response-header field specifies that the entity has multiple sources and may therefore vary according to specified list of request header(s). Following is the general syntax:

Vary : field-name

You can specify multiple headers separated by commas and a value of asterisk "\*" signals that unspecified parameters not limited to the request-headers. Following is a simple example:

Vary: Accept-Language, Accept-Encoding

Here field names are case-insensitive.

WWW-Authenticate

The *WWW-Authenticate* response-header field must be included in 401 (Unauthorized) response messages. The field value consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable to the Request-URI. Following is the general syntax:

WWW-Authenticate : challenge

WWW- Authenticate field value as it might contain more than one challenge, or if more than one WWW-Authenticate header field is provided, the contents of a challenge itself can contain a comma-separated list of authentication parameters. Following is a simple example:

WWW-Authenticate: BASIC realm="Admin"

Entity Headers

Allow

The *Allow* entity-header field lists the set of methods supported by the resource identified by the Request-URI. Following is the general syntax:

Allow : Method

You can specify multiple method separated by commas. Following is a simple example:

Allow: GET, HEAD, PUT

This field cannot prevent a client from trying other methods.

Content-Encoding

The *Content-Encoding* entity-header field is used as a modifier to the media-type. Following is the general syntax:

Content-Encoding : content-coding

The content-coding is a characteristic of the entity identified by the Request-URI. Following is a simple example:

Content-Encoding: gzip

If the content-coding of an entity in a request message is not acceptable to the origin server, the server should respond with a status code of 415 (Unsupported Media Type).

Content-Language

The *Content-Language* entity-header field describes the natural language(s) of the intended audience for the enclosed entity. Following is the general syntax:

Content-Language : language-tag

Multiple languages may be listed for content that is intended for multiple audiences. Following is a simple example:

Content-Language: mi, en

The primary purpose of Content-Language is to allow a user to identify and differentiate entities according to the user's own preferred language.

Content-Length

The *Content-Length* entity-header field indicates the size of the entity-body, in decimal number of OCTETs, sent to the recipient or, in the case of the HEAD method, the size of the entity-body that would have been sent had the request been a GET. Following is the general syntax:

Content-Length : DIGITS

Following is a simple example:

Content-Length: 3495

Any Content-Length greater than or equal to zero is a valid value.

Content-Location

The *Content-Location* entity-header field may be used to supply the resource location for the entity enclosed in the message when that entity is accessible from a location separate from the requested resource's URI. Following is the general syntax:

Content-Location: absoluteURI | relativeURI

Following is a simple example:

Content-Location: http://www.tutorialspoint.org/http/index.htm

The value of Content-Location also defines the base URI for the entity.

Content-MD5

The *Content-MD5* entity-header field may be used to supply an MD5 digest of the entity, for checking the integrity of the message upon receipt. Following is the general syntax:

Content-MD5 : md5-digest using base64 of 128 bit MD5 digest as per RFC 1864

Following is a simple example:

Content-MD5 : 8c2d46911f3f5a326455f0ed7a8ed3b3

The MD5 digest is computed based on the content of the entity-body, including any content-coding that has been applied, but not including any transfer-encoding applied to the message-body.

Content-Range

The *Content-Range* entity-header field is sent with a partial entity-body to specify where in the full entity-body the partial body should be applied. Following is the general syntax:

Content-Range : bytes-unit SP first-byte-pos "-" last-byte-pos

Examples of byte-content-range-spec values, assuming that the entity contains a total of 1234 bytes:

- The first 500 bytes:

Content-Range : bytes 0-499/1234

- The second 500 bytes:

Content-Range : bytes 500-999/1234

- All except for the first 500 bytes:

Content-Range : bytes 500-1233/1234

- The last 500 bytes:

Content-Range : bytes 734-1233/1234

When an HTTP message includes the content of a single range, this content is transmitted with a Content-Range header, and a Content-Length header showing the number of bytes actually transferred. For example,

HTTP/1.1 206 Partial content

Date: Wed, 15 Nov 1995 06:25:24 GMT

Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT

Content-Range: bytes 21010-47021/47022

Content-Length: 26012

Content-Type: image/gif

Content-Type

The *Content-Type* entity-header field indicates the media type of the entity-body sent to the recipient or, in the case of the HEAD method, the media type that would have been sent had the request been a GET. Following is the general syntax:

Content-Type : media-type

Following is an example:

Content-Type: text/html; charset=ISO-8859-4

Expires

The *Expires* entity-header field gives the date/time after which the response is considered stale. Following is the general syntax:

Expires : HTTP-date

Following is an example:

Expires: Thu, 01 Dec 1994 16:00:00 GMT

Last-Modified

The *Last-Modified* entity-header field indicates the date and time at which the origin server believes the variant was last modified. Following is the general syntax:

Last-Modified: HTTP-date

Following is an example:

Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT

HTTP - Caching

HTTP is typically used for distributed information systems, where performance can be improved by the use of response caches. The HTTP/1.1 protocol includes a number of elements intended to make caching work.

The goal of caching in HTTP/1.1 is to eliminate the need to send requests in many cases, and to eliminate the need to send full responses in many other cases.

The basic cache mechanisms in HTTP/1.1 are implicit directives to caches where server-specifies expiration times and validators. We use the **Cache-Control** header for this purpose.

The **Cache-Control** header allows a client or server to transmit a variety of directives in either requests or responses. These directives typically override the default caching algorithms. The caching directives are specified in a comma-separated list. For example:

Cache-control: no-cache

There are following important cache request directives which can be used by the client in its HTTP request:

|  |  |
| --- | --- |
| **S.N.** | **Cache Request Directive and Description** |
| 1 | **no-cache** A cache must not use the response to satisfy a subsequent request without successful revalidation with the origin server. |
| 2 | **no-store** The cache should not store anything about the client request or server response. |
| 3 | **max-age = seconds** Indicates that the client is willing to accept a response whose age is no greater than the specified time in seconds. |
| 4 | **max-stale [ = seconds ]** Indicates that the client is willing to accept a response that has exceeded its expiration time. If seconds are given, it must not be expired by more than that time. |
| 5 | **min-fresh = seconds** Indicates that the client is willing to accept a response whose freshness lifetime is no less than its current age plus the specified time in seconds. |
| 6 | **no-transform** Do not convert the entity-body. |
| 7 | **only-if-cached** Do not retrieve new data. The cache can send a document only if it is in the cache, and should not contact the origin-server to see if a newer copy exists. |

There are following important cache response directives which can be used by the server in its HTTP response:

|  |  |
| --- | --- |
| **S.N.** | **Cache Response Directive and Description** |
| 1 | **public** Indicates that the response may be cached by any cache. |
| 2 | **private** Indicates that all or part of the response message is intended for a single user and must not be cached by a shared cache. |
| 3 | **no-cache** A cache must not use the response to satisfy a subsequent request without successful revalidation with the origin server. |
| 4 | **no-store** The cache should not store anything about the client request or server response. |
| 5 | **no-transform** Do not convert the entity-body. |
| 6 | **must-revalidate** The cache must verify the status of stale documents before using it and expired one should not be used. |
| 7 | **proxy-revalidate** The proxy-revalidate directive has the same meaning as the must- revalidate directive, except that it does not apply to non-shared user agent caches. |
| 8 | **max-age = seconds** Indicates that the client is willing to accept a response whose age is no greater than the specified time in seconds. |
| 9 | **s-maxage = seconds** The maximum age specified by this directive overrides the maximum age specified by either the max-age directive or the Expires header. The s-maxage directive is always ignored by a private cache. |

HTTP - URL Encoding

HTTP URLs can only be sent over the Internet using the ASCII *character-set*, which often contain characters outside the ASCII set. So these unsafe characters must be replaced with a **%** followed by two hexadecimal digits.

Following table shows ASCII symbol of the character and its equal Symbol and finally its replacement which can be used in URL before passing it to the server:

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Symbol** | **Replacement** |
| < 32 |  | Encode with %xx where xx is the hexadecimal representation of the character. |
| 32 | space | + or %20 |
| 33 | ! | %21 |
| 34 | " | %22 |
| 35 | # | %23 |
| 36 | $ | %24 |
| 37 | % | %25 |
| 38 | & | %26 |
| 39 | ' | %27 |
| 40 | ( | %28 |
| 41 | ) | %29 |
| 42 | \* | \* |
| 43 | + | %2B |
| 44 | , | %2C |
| 45 | - | - |
| 46 | . | . |
| 47 | / | %2F |
| 48 | 0 | 0 |
| 49 | 1 | 1 |
| 50 | 2 | 2 |
| 51 | 3 | 3 |
| 52 | 4 | 4 |
| 53 | 5 | 5 |
| 54 | 6 | 6 |
| 55 | 7 | 7 |
| 56 | 8 | 8 |
| 57 | 9 | 9 |
| 58 | : | %3A |
| 59 | ; | %3B |
| 60 | < | %3C |
| 61 | = | %3D |
| 62 | > | %3E |
| 63 | ? | %3F |
| 64 | @ | %40 |
| 65 | A | A |
| 66 | B | B |
| 67 | C | C |
| 68 | D | D |
| 69 | E | E |
| 70 | F | F |
| 71 | G | G |
| 72 | H | H |
| 73 | I | I |
| 74 | J | J |
| 75 | K | K |
| 76 | L | L |
| 77 | M | M |
| 78 | N | N |
| 79 | O | O |
| 80 | P | P |
| 81 | Q | Q |
| 82 | R | R |
| 83 | S | S |
| 84 | T | T |
| 85 | U | U |
| 86 | V | V |
| 87 | W | W |
| 88 | X | X |
| 89 | Y | Y |
| 90 | Z | Z |
| 91 | [ | %5B |
| 92 | \ | %5C |
| 93 | ] | %5D |
| 94 | ^ | %5E |
| 95 | \_ | \_ |
| 96 | ` | %60 |
| 97 | a | a |
| 98 | b | b |
| 99 | c | c |
| 100 | d | d |
| 101 | e | e |
| 102 | f | f |
| 103 | g | g |
| 104 | h | h |
| 105 | i | i |
| 106 | j | j |
| 107 | k | k |
| 108 | l | l |
| 109 | m | m |
| 110 | n | n |
| 111 | o | o |
| 112 | p | p |
| 113 | q | q |
| 114 | r | r |
| 115 | s | s |
| 116 | t | t |
| 117 | u | u |
| 118 | v | v |
| 119 | w | w |
| 120 | x | x |
| 121 | y | y |
| 122 | z | z |
| 123 | { | %7B |
| 124 | | | %7C |
| 125 | } | %7D |
| 126 | ~ | %7E |
| 127 |  | %7F |
| > 127 |  | Encode with %xx where xx is the hexadecimal representation of the character |

HTTP - Security

HTTP is used for a communication over the internet, so application developers, information providers, and users should be aware of the security limitations in HTTP/1.1. This discussion does not include definitive solutions to the problems mentioned here but it does make some suggestions for reducing security risks.

Personal Information leakage

HTTP clients are often privy to large amounts of personal information such as the user's name, location, mail address, passwords, encryption keys, etc. So you should be very careful to prevent unintentional leakage of this information via the HTTP protocol to other sources.

* All the confidential information should be stored at server side in encrypted form.
* Revealing the specific software version of the server might allow the server machine to become more vulnerable to attacks against software that is known to contain security holes.
* Proxies which serve as a portal through a network firewall should take special precautions regarding the transfer of header information that identifies the hosts behind the firewall.
* The information sent in the From field might conflict with the user's privacy interests or their site's security policy, and hence it should not be transmitted without the user being able to disable, enable, and modify the contents of the field.
* Clients should not include a Referer header field in a (non-secure) HTTP request if the referring page was transferred with a secure protocol.
* Authors of services which use the HTTP protocol should not use GET based forms for the submission of sensitive data, because this will cause this data to be encoded in the Request-URI

File and path names based attack

The document should be restricted to the documents returned by HTTP requests to be only those that were intended by the server administrators.

For example, UNIX, Microsoft Windows, and other operating systems use **..** as a path component to indicate a directory level above the current one. On such a system, an HTTP server MUST disallow any such construct in the Request-URI if it would otherwise allow access to a resource outside those intended to be accessible via the HTTP server.

DNS Spoofing

Clients using HTTP rely heavily on the Domain Name Service, and are thus generally prone to security attacks based on the deliberate mis-association of IP addresses and DNS names. So clients need to be cautious in assuming the continuing validity of an IP number/DNS name association.

If HTTP clients cache the results of host name lookups in order to achieve a performance improvement, they must observe the TTL information reported by DNS. If HTTP clients do not observe this rule, they could be spoofed when a previously-accessed server's IP address changes.

Location Headers and Spoofing

If a single server supports multiple organizations that do not trust one another, then it MUST check the values of Location and Content- Location headers in responses that are generated under control of said organizations to make sure that they do not attempt to invalidate resources over which they have no authority.

Authentication Credentials

Existing HTTP clients and user agents typically retain authentication information indefinitely. HTTP/1.1. does not provide a method for a server to direct clients to discard these cached credentials which is a big security risk.

There are a number of work- arounds to parts of this problem, and so its is recommended to make the use of password protection in screen savers, idle time-outs, and other methods which mitigate the security problems inherent in this problem.

Proxies and Caching

HTTP proxies are men-in-the-middle, and represent an opportunity for man-in-the-middle attacks. Proxies have access to security-related information, personal information about individual users and organizations, and proprietary information belonging to users and content providers.

Proxy operators should protect the systems on which proxies run as they would protect any system that contains or transports sensitive information.

Caching proxies provide additional potential vulnerabilities, since the contents of the cache represent an attractive target for malicious exploitation. Therefore, cache contents should be protected as sensitive information.

HTTP - Message Examples

Example 1

HTTP request to fetch **hello.htm** page from the web server running on *tutorialspoint.com*

Client request

GET /hello.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Server response

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

Content-Length: 88

Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

Example 2

HTTP request to fetch **t.html** page which does not exist on the web server running on *tutorialspoint.com*

Client request

GET /**t.html** HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Server response

HTTP/1.1 404 Not Found

Date: Sun, 18 Oct 2012 10:36:20 GMT

Server: Apache/2.2.14 (Win32)

Content-Length: 230

Content-Type: text/html; charset=iso-8859-1

Connection: close

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">

<html>

<head>

<title>404 Not Found</title>

</head>

<body>

<h1>Not Found</h1>

<p>The requested URL /t.html was not found on this server.</p>

</body>

</html>

Example 3

HTTP request to fetch **hello.htm** page from the web server running on *tutorialspoint.com*, but request goes with wrong HTTP version:

Client request

GET /hello.htm **HTTP1**

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Server response

HTTP/1.1 400 Bad Request

Date: Sun, 18 Oct 2012 10:36:20 GMT

Server: Apache/2.2.14 (Win32)

Content-Length: 230

Content-Type: text/html; charset=iso-8859-1

Connection: close

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">

<html>

<head>

<title>400 Bad Request</title>

</head>

<body>

<h1>Bad Request</h1>

<p>Your browser sent a request that this server could not understand.<p>

<p>The request line contained invalid characters following the protocol string.<p>

</body>

</html>

Example 4

HTTP request to post form data to **process.cgi** CGI page on a web server running on *tutorialspoint.com*. Server returns passed name after setting them as cookies:

Client request

POST /cgi-bin/process.cgi HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Content-Type: text/xml; charset=utf-8

Content-Length: 60

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

first=Zara&last=Ali

Server response

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Content-Length: 88

Set-Cookie: first=Zara,last=Ali;domain=tutorialspoint.com;Expires=Mon, 19-

Nov-2010 04:38:14 GMT;Path=/

Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Hello Zara Ali</h1>

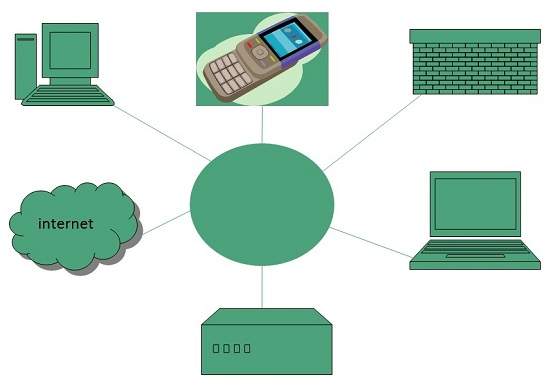
</body>

</html>

Internet

## Internet

* Internet is a world-wide global system of interconnected computer networks.
* Internet uses the standard Internet Protocol (TCP/IP).
* Every computer in internet is identified by a unique IP address.
* IP Address is a unique set of numbers (such as 110.22.33.114) which identifies a computer location.
* A special computer DNS (Domain Name Server) is used to give name to the IP Address so that user can locate a computer by a name.
* For example, a DNS server will resolve a name **http://www.tutorialspoint.com** to a particular IP address to uniquely identify the computer on which this website is hosted.
* Internet is accessible to every user all over the world.



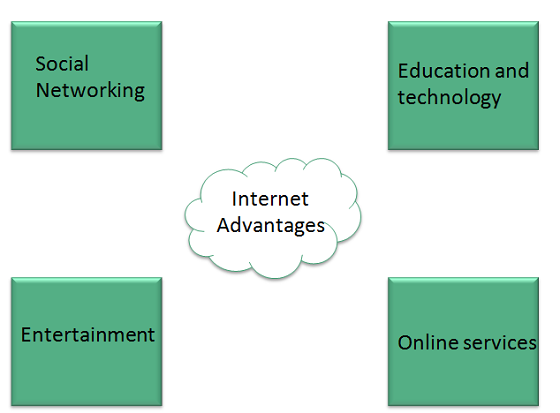
### Internet Evoloution

The concept of Internet was originated in 1969 and has undergone several technological & Infrastructural changes as discussed below:

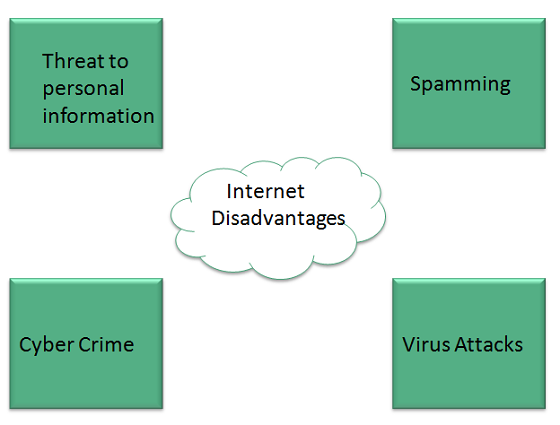
* The origin of Internet devised from the concept of **Advanced Research Project Agency Network (ARPANET).**
* **ARPANET** was developed by United States Department of Defense.
* Basic purpose of ARPANET was to provide communication among the various bodies of government.
* Initially, there were only four nodes, formally called **Hosts.**
* In 1972, the **ARPANET** spread over the globe with 23 nodes located at different countries and thus became known as **Internet.**
* By the time, with invention of new technologies such as TCP/IP protocols, DNS, WWW, browsers, scripting languages etc.,Internet provided a medium to publish and access information over the web.

### Internet Advantages

Internet covers almost every aspect of life, one can think of. Here, we will discuss some of the advantages of Internet:

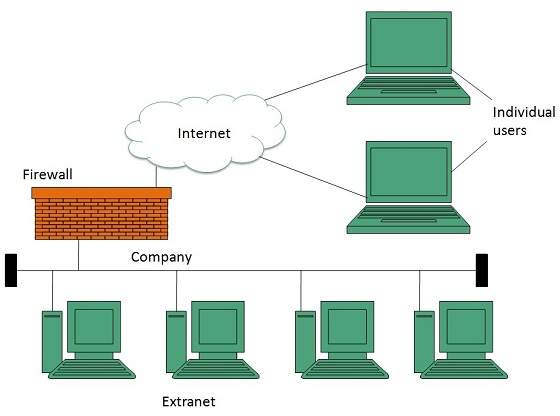


### Internet Disadvantages



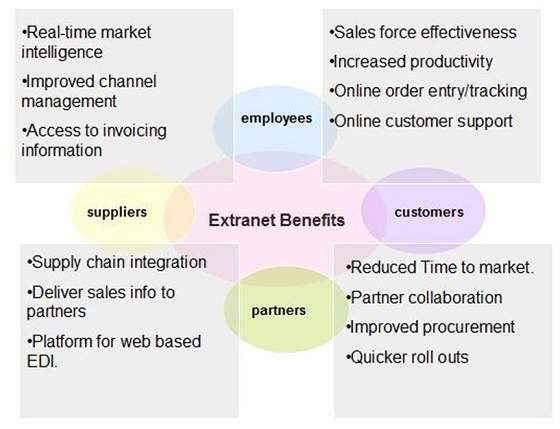
## Extranet

Extranet refers to network within an organization, using internet to connect to the outsiders in controlled manner. It helps to connect businesses with their customers and suppliers and therefore allows working in a collaborative manner.



### Extranet Benefits

Extranet proves to be a successful model for all kind of businesses whether small or big. Here are some of the advantages of extranet for employees, suppliers, business partners, and customers:



### Extranet Issues

Apart for advantages there are also some issues associated with extranet. These issues are discussed below:

### Hosting

Where the extranet pages will be held i.e. who will host the extranet pages. In this context there are two choices:

* Host it on your own server.
* Host it with an Internet Service Provider (ISP) in the same way as web pages.

But hosting extranet pages on your own server requires high bandwidth internet connection which is very costly.

### Security

Additional firewall security is required if you host extranet pages on your own server which result in a complex security mechanism and increase work load.

### Accessing Issues

Information can not be accessed without internet connection. However, information can be accessed in Intranet without internet connection.

### Decreased Interaction

It decreases the face to face interaction in the business which results in lack of communication among customers, business partners and suppliers.

### Extranet vs. Intranet

The following table shows differences between Extranet and Intranet:

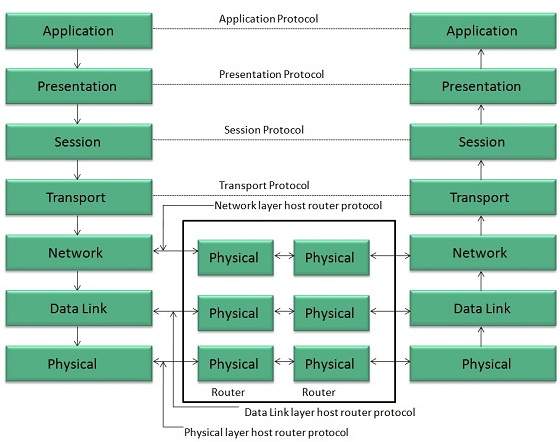
|  |  |
| --- | --- |
| **Extranet** | **Intranet** |
| Internal network that can be accessed externally. | Internal network that can not be accessed externally. |
| Extranet is extension of company's Intranet. | Only limited users of a company. |
| For limited external communication between customers, suppliers and business partners. | Only for communication within a company. |

## Internet Reference Models

### OSI Reference Model

**OSI** is acronym of **Open System Interface**. This model is developed by the **International organization of Standardization (ISO)** and therefore also referred as **ISO-OSI** Model.

The OSI model consists of seven layers as shown in the following diagram. Each layer has a specific function, however each layer provide services to the layer above.



#### **Physical Layer**

The Physical layer is responsible for the following activities:

* Activating, maintaining and deactivating the physical connection.
* Defining voltages and data rates needed for transmission.
* Converting digital bits into electrical signal.
* Deciding whether the connection is simplex, half duplex or full duplex.

#### **Data Link Layer**

The data link layer performs the following functions:

* Performs synchronization and error control for the information which is to be transmitted over the physical link.
* Enables error detection, and adds error detection bits to the data which are to be transmitted.

#### **Network Layer**

Following are the functions of Network Layer:

* To route the signals through various channels to the other end.
* To act as the network controller by deciding which route data should take.
* To divide the outgoing messages into packets and to assemble incoming packets into messages for higher levels.

#### **Transport Layer**

The Transport layer performs the following functions:

* It decides if the data transmission should take place on parallel paths or single path.
* It performs multiplexing, splitting on the data.
* It breaks the data groups into smaller units so that they are handled more efficiently by the network layer.

The Transport Layer guarantees transmission of data from one end to other end.

#### **Session Layer**

The Session layer performs the following functions:

* Manages the messages and synchronizes conversations between two different applications.
* It controls logging on and off, user identification, billing and session management.

#### **Presentation Layer**

The Presentation layer performs the following functions:

<="" ul="" style="box-sizing: border-box; padding: 0px; margin: 0px 0px 20px 45px; font-size: 16px; color: rgb(116, 117, 121); font-family: Nunito, sans-serif; font-style: normal; font-variant-ligatures: normal; font-variant-caps: normal; font-weight: 400; letter-spacing: normal; orphans: 2; text-align: start; text-indent: 0px; text-transform: none; white-space: normal; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px; text-decoration-thickness: initial; text-decoration-style: initial; text-decoration-color: initial;">

* This layer makes it sure that the information is delivered in such a form that the receiving system will understand and use it.

#### **Application Layer**

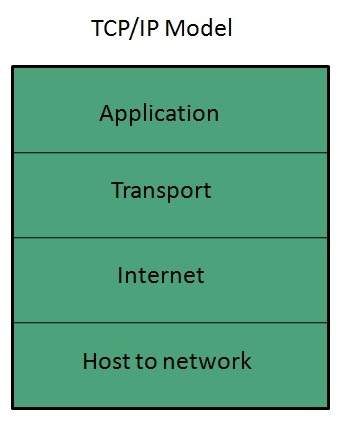
The Application layer performs the following functions:

* It provides different services such as manipulation of information in several ways, retransferring the files of information, distributing the results etc.
* The functions such as LOGIN or password checking are also performed by the application layer.

### TCP/IP Reference Model

**TCP/IP** model is practical model and is used in the Internet. TCP/IP is acronym of Transmission Control Protocol and Internet Protocol.

The **TCP/IP** model combines the two layers (Physical and Data link layer) into one layer i.e. **Host-to-Network** layer. The following diagram shows the various layers of TCP/IP model:



#### **Application Layer**

This layer is same as that of the OSI model and performs the following functions:

* It provides different services such as manipulation of information in several ways, retransferring the files of information, distributing the results etc.
* The functions such as LOGIN or password checking are also performed by the application layer.

**Protocols used: TELNET, FTP, SMTP, DN, HTTP, NNTP** are the protocols employed in this layer.

#### **Transport Layer**

It does the same functions as that of transport layer in OSI model. Here are the key points regarding transport layer:

* It uses **TCP** and **UDP** protocol for end to end transmission.
* TCP is reliable and **connection oriented protocol.**
* TCP also handles flow control.
* The UDP is not reliable and a **connection less protocol** also does not perform flow control.

**Protocols used: TCP/IP** and **UDP** protocols are employed in this layer.

#### **Internet Layer**

The function of this layer is to allow the host to insert packets into network and then make them travel independently to the destination. However, the order of receiving the packet can be different from the sequence they were sent.

**Protocols used: Internet Protocol (IP)** is employed in Internet layer.

#### **Host-to-Network Layer**

This is the lowest layer in TCP/IP model. The host has to connect to network using some protocol, so that it can send IP packets over it. This protocol varies from host to host and network to network.

**Protocols used: ARPANET, SATNET, LAN, packet radio** are the protocols which are used in this layer.

AD

## Domain Name System Architecture

The Domain name system comprises of **Domain Names, Domain Name Space, Name Server** that have been described below:

### Domain Names

Domain Name is a symbolic string associated with an IP address. There are several domain names available; some of them are generic such as **com, edu, gov, net** etc, while some country level domain names such as **au, in, za, us** etc.

The following table shows the **Generic** Top-Level Domain names:

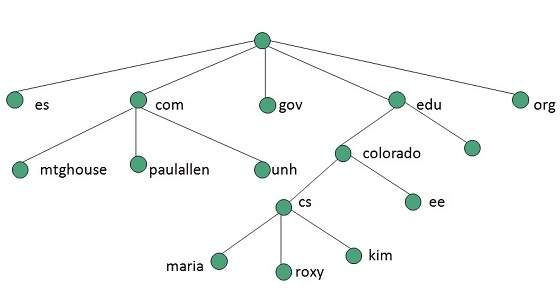
|  |
| --- |
|  |
| **Domain Name** | **Meaning** |
| Com | Commercial business |
| Edu | Education |
| Gov | U.S. government agency |
| Int | International entity |
| Mil | U.S. military |
| Net | Networking organization |
| Org | Non profit organization |

The following table shows the **Country top-level** domain names:

|  |  |
| --- | --- |
| **Domain Name** | **Meaning** |
| au | Australia |
| in | India |
| cl | Chile |
| fr | France |
| us | United States |
| za | South Africa |
| uk | United Kingdom |
| jp | Japan |
| es | Spain |
| de | Germany |
| ca | Canada |
| ee | Estonia |
| hk | Hong Kong |

### Domain Name Space

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy:



In the above diagram each subtree represents a domain. Each domain can be partitioned into sub domains and these can be further partitioned and so on.

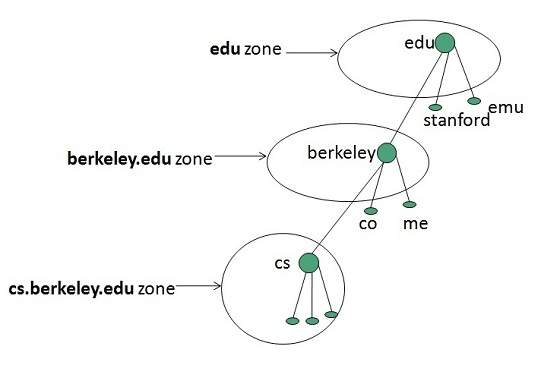
### Name Server

Name server contains the DNS database. This database comprises of various names and their corresponding IP addresses. Since it is not possible for a single server to maintain entire DNS database, therefore, the information is distributed among many DNS servers.

* Hierarchy of server is same as hierarchy of names.
* The entire name space is divided into the zones

### Zones

Zone is collection of nodes (sub domains) under the main domain. The server maintains a database called zone file for every zone.



If the domain is not further divided into sub domains then domain and zone refers to the same thing.

The information about the nodes in the sub domain is stored in the servers at the lower levels however; the original server keeps reference to these lower levels of servers.

#### **Types of Name Servers**

Following are the three categories of Name Servers that manages the entire Domain Name System:

* Root Server
* Primary Server
* Secondary Server

##### Root Server

Root Server is the top level server which consists of the entire DNS tree. It does not contain the information about domains but delegates the authority to the other server

##### Primary Servers

Primary Server stores a file about its zone. It has authority to create, maintain, and update the zone file.

##### Secondary Server

Secondary Server transfers complete information about a zone from another server which may be primary or secondary server. The secondary server does not have authority to create or update a zone file.

## DNS Working

DNS translates the domain name into IP address automatically. Following steps will take you through the steps included in domain resolution process:

* When we type **www.tutorialspoint.com** into the browser, it asks the local DNS Server for its IP address.

Here the local DNS is at ISP end.

* When the local DNS does not find the IP address of requested domain name, it forwards the request to the root DNS server and again enquires about IP address of it.
* The root DNS server replies with delegation that **I do not know the IP address of www.tutorialspoint.com but know the IP address of DNS Server.**
* The local DNS server then asks the com DNS Server the same question.
* The **com** DNS Server replies the same that it does not know the IP address of www.tutorialspont.com but knows the address of tutorialspoint.com.
* Then the local DNS asks the tutorialspoint.com DNS server the same question.
* Then tutorialspoint.com DNS server replies with IP address of www.tutorialspoint.com.
* Now, the local DNS sends the IP address of www.tutorialspoint.com to the computer that sends the request.

## Internet Services

### Communication Services

There are various Communication Services available that offer exchange of information with individuals or groups. The following table gives a brief introduction to these services:

|  |  |
| --- | --- |
| **S.N.** | **Service Description** |
| 1 | **Electronic Mail** Used to send electronic message over the internet. |
| 2 | **Telnet** Used to log on to a remote computer that is attached to internet. |
| 3 | **Newsgroup** Offers a forum for people to discuss topics of common interests. |
| 4 | **Internet Relay Chat (IRC)** Allows the people from all over the world to communicate in real time. |
| 5 | **Mailing Lists** Used to organize group of internet users to share common information through e-mail. |
| 6 | **Internet Telephony (VoIP)** Allows the internet users to talk across internet to any PC equipped to receive the call. |
| 7 | **Instant Messaging** Offers real time chat between individuals and group of people. Eg. Yahoo messenger, MSN messenger. |

### Information Retrieval Services

There exist several Information retrieval services offering easy access to information present on the internet. The following table gives a brief introduction to these services:

|  |  |
| --- | --- |
| **S.N.** | **Service Description** |
| 1 | **File Transfer Protocol (FTP)** Enable the users to transfer files. |
| 2 | **Archie** It’s updated database of public FTP sites and their content. It helps to search a file by its name. |
| 3 | **Gopher** Used to search, retrieve, and display documents on remote sites. |
| 4 | **Very Easy Rodent Oriented Netwide Index to Computer Achieved (VERONICA)** VERONICA is gopher based resource. It allows access to the information resource stored on gopher’s servers. |

### Web Services

Web services allow exchange of information between applications on the web. Using web services, applications can easily interact with each other.

The web services are offered using concept of **Utility Computing.**

### World Wide Web (WWW)

WWW is also known as W3. It offers a way to access documents spread over the several servers over the internet. These documents may contain texts, graphics, audio, video, hyperlinks. The hyperlinks allow the users to navigate between the documents.

### Video Conferencing

Video conferencing or Video teleconferencing is a method of communicating by two-way video and audio transmission with help of telecommunication technologies.

#### **Modes of Video Conferencing**

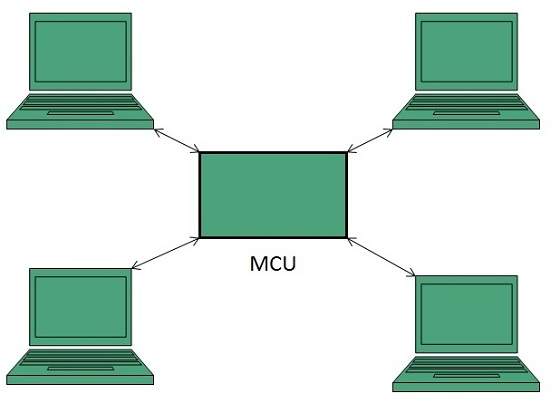
##### Point-to-Point

This mode of conferencing connects two locations only.



##### Multi-point

This mode of conferencing connects more than two locations through **Multi-point Control Unit (MCU).**



## Internet Protocols

### Transmission Control Protocol (TCP)

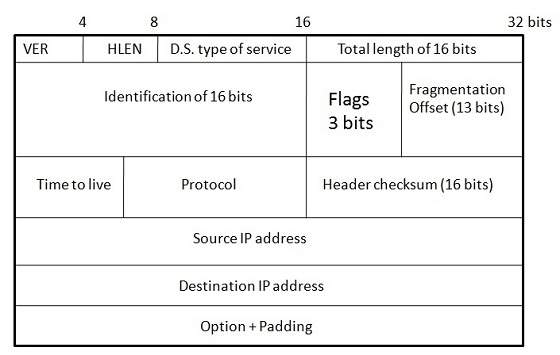
* Transmission Control Protocol (TCP) corresponds to the Transport Layer of OSI Model.
* TCP is a reliable and connection oriented protocol.
* TCP offers:
  + Stream Data Transfer.
  + Reliability.
  + Efficient Flow Control
  + Full-duplex operation.
  + Multiplexing.
* TCP offers connection oriented end-to-end packet delivery.
* TCP ensures reliability by sequencing bytes with a forwarding acknowledgement number that indicates to the destination the next byte the source expect to receive.
* It retransmits the bytes not acknowledged with in specified time period.

### Internet Protocol (IP)

Internet Protocol is **connectionless** and **unreliable** protocol. It ensures no guarantee of successfully transmission of data.

In order to make it reliable, it must be paired with reliable protocol such as TCP at the transport layer.

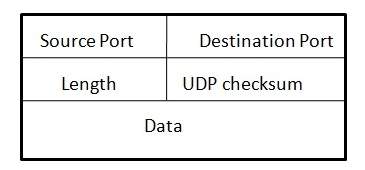
Internet protocol transmits the data in form of a datagram as shown in the following diagram:



### User Datagram Protocol (UDP)

Like IP, UDP is connectionless and unreliable protocol. It doesn’t require making a connection with the host to exchange data. Since UDP is unreliable protocol, there is no mechanism for ensuring that data sent is received.

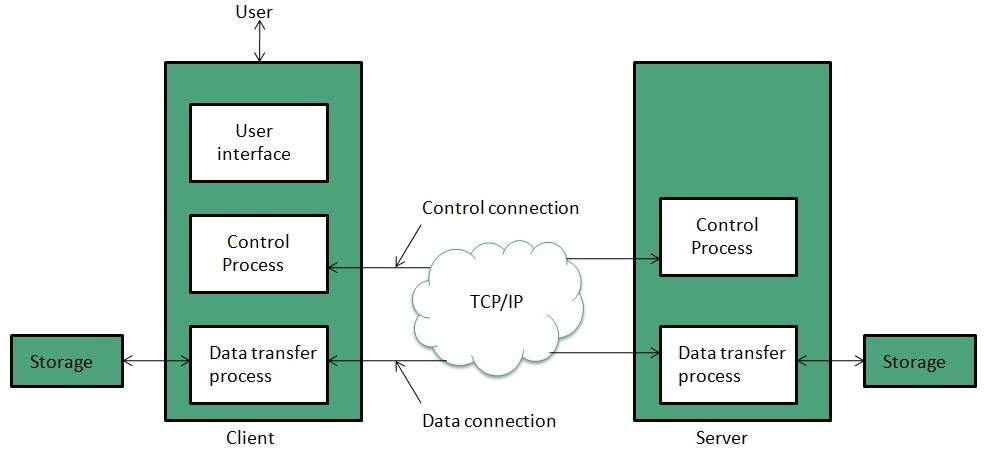
UDP transmits the data in form of a datagram. The UDP datagram consists of five parts as shown in the following diagram:



### File Transfer Protocol (FTP)

FTP is used to copy files from one host to another. FTP offers the mechanism for the same in following manner:

* FTP creates two processes such as Control Process and Data Transfer Process at both ends i.e. at client as well as at server.
* FTP establishes two different connections: one is for data transfer and other is for control information.
* **Control connection** is made between **control processes** while **Data Connection** is made between<="" b="" style="box-sizing: border-box;">
* FTP uses **port 21** for the control connection and **Port 20** for the data connection.



### Trivial File Transfer Protocol (TFTP)

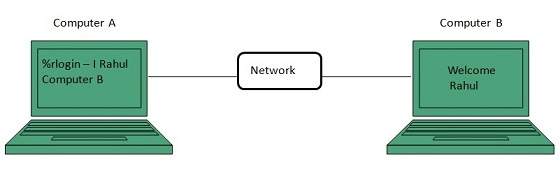
**Trivial File Transfer Protocol** is also used to transfer the files but it transfers the files without authentication. Unlike FTP, TFTP does not separate control and data information. Since there is no authentication exists, TFTP lacks in security features therefore it is not recommended to use TFTP.

**Key points**

* TFTP makes use of UDP for data transport. Each TFTP message is carried in separate UDP datagram.
* The first two bytes of a TFTP message specify the type of message.
* The TFTP session is initiated when a TFTP client sends a request to upload or download a file.
* The request is sent from an ephemeral UDP port to the **UDP port 69**of an TFTP server.

### Telnet

Telnet is a protocol used to log in to remote computer on the internet. There are a number of Telnet clients having user friendly user interface. The following diagram shows a person is logged in to computer A, and from there, he remote logged into computer B.



### Hyper Text Transfer Protocol (HTTP)

HTTP is a communication protocol. It defines mechanism for communication between browser and the web server. It is also called request and response protocol because the communication between browser and server takes place in request and response pairs.

#### **HTTP Request**

HTTP request comprises of lines which contains:

* Request line
* Header Fields
* Message body

**Key Points**

* The first line i.e. the **Request line** specifies the request method i.e. **Get** or **Post.**
* The second line specifies the header which indicates the domain name of the server from where index.htm is retrieved.

#### **HTTP Response**

Like HTTP request, HTTP response also has certain structure. HTTP response contains:

* Status line
* Headers
* Message body

## Email

Email is a service which allows us to send the message in electronic mode over the internet. It offers an efficient, inexpensive and real time mean of distributing information among people.

### E-mail Protocols

#### **SMPTP**

**SMTP** stands for **Simple Mail Transfer Protocol**. It was first proposed in 1982. It is a standard protocol used for sending e-mail efficiently and reliably over the internet.

**Key Points:**

* SMTP is application level protocol.
* SMTP is connection oriented protocol.
* SMTP is text based protocol.
* It handles exchange of messages between e-mail servers over TCP/IP network.
* Apart from transferring e-mail, SMPT also provides notification regarding incoming mail.
* When you send e-mail, your e-mail client sends it to your e-mail server which further contacts the recipient mail server using SMTP client.
* These SMTP commands specify the sender’s and receiver’s e-mail address, along with the message to be send.
* The exchange of commands between servers is carried out without intervention of any user.
* In case, message cannot be delivered, an error report is sent to the sender which makes SMTP a reliable protocol.

### IMAP

**IMAP** stands for **Internet Message Access Protocol.** It was first proposed in 1986. There exist five versions of IMAP as follows:

* Original IMAP
* IMAP2
* IMAP3
* IMAP2bis
* IMAP4

**Key Points:**

* IMAP allows the client program to manipulate the e-mail message on the server without downloading them on the local computer.
* The e-mail is hold and maintained by the remote server.
* It enables us to take any action such as downloading, delete the mail without reading the mail.It enables us to create, manipulate and delete remote message folders called mail boxes.
* IMAP enables the users to search the e-mails.
* It allows concurrent access to multiple mailboxes on multiple mail servers.

### POP

POP stands for Post Office Protocol. It is generally used to support a single client. There are several versions of POP but the POP 3 is the current standard.

**Key Points**

* POP is an application layer internet standard protocol.
* Since POP supports offline access to the messages, thus requires less internet usage time.
* POP does not allow search facility.
* In order to access the messaged, it is necessary to download them.
* It allows only one mailbox to be created on server.
* It is not suitable for accessing non mail data.
* POP commands are generally abbreviated into codes of three or four letters. Eg. STAT.

### E-mail Working

Email working follows the client server approach. In this client is the mailer i.e. the mail application or mail program and server is a device that manages emails.

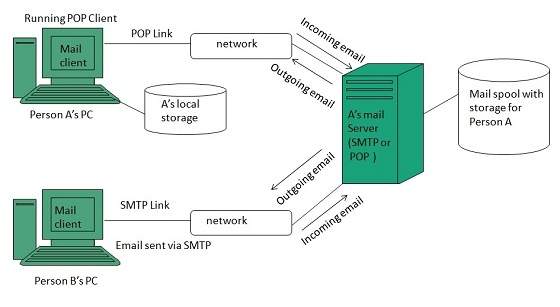
Following example will take you through the basic steps involved in sending and receiving emails and will give you a better understanding of working of email system:

* Suppose person A wants to send an email message to person B.
* Person A composes the messages using a mailer program i.e. mail client and then select Send option.
* The message is routed to **Simple Mail Transfer Protocol** to person B’s mail server.
* The mail server stores the email message on disk in an area designated for person B.

The disk space area on mail server is called mail spool.

* Now, suppose person B is running a POP client and knows how to communicate with B’s mail server.
* It will periodically poll the POP server to check if any new email has arrived for B.As in this case, person B has sent an email for person B, so email is forwarded over the network to B’s PC. This is message is now stored on person B’s PC.

The following diagram gives pictorial representation of the steps discussed above:

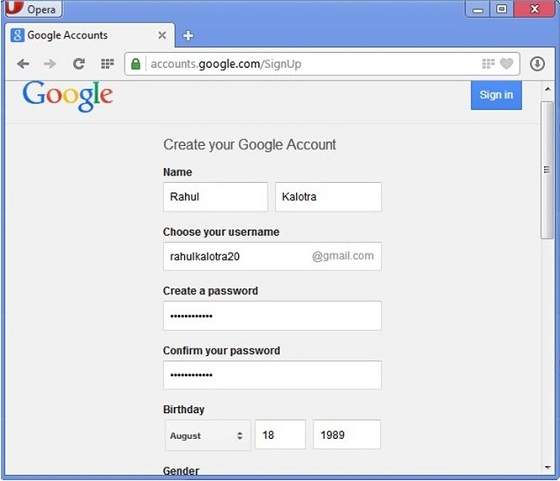


### E-mail Operations

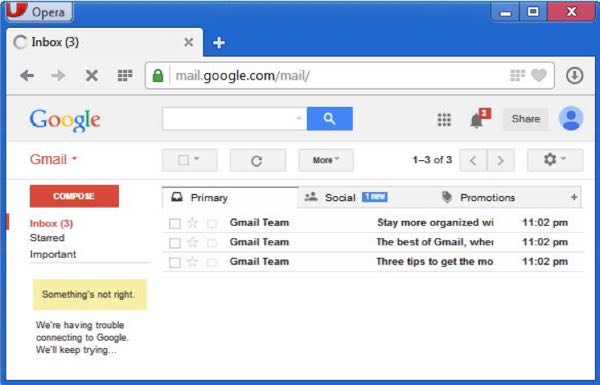
#### **Creating Email Account**

There are various email service provider available such as **Gmail, hotmail, ymail, rediff mail** etc. Here we will learn how to create an account using Gmail.

* Open gmail.com and click **create an account.**
* Now a form will appear. Fill your details here and click **Next Step.**



* This step allows you to add your picture. If you don’t want to upload now, you can do it later. Click **Next Step.**
* Now a welcome window appears. Click **Continue to Gmail.**
* Wow!! You are done with creating your email account with Gmail. It’s that easy. Isn’t it?
* Now you will see your Gmail account as shown in the following image:



**Key Points:**

* Gmail manages the mail into three categories namely **Primary, Social** and **Promotions.**
* **Compose** option is given at the right to compose an email message.
* **Inbox, Starred, Sent mail, Drafts** options are available on the left pane which allows you to keep track of your emails.

#### **Composing and Sending Email**

Before sending an email, we need to compose a message. When we are composing an email message, we specify the following things:

* Sender’s address in To field
* Cc (if required)
* Bcc (if required)
* Subject of email message
* Text
* Signature

You should specify the correct email address; otherwise it will send an error back to the sender.

Once you have specified all the above parameters, It’s time to send the email. The mailer program provides a Send button to send email, when you click Send, it is sent to the mail server and a message **mail sent successfully** is shown at the above.

#### **Reading Email**

Every email program offers you an interface to access email messages. Like in Gmail, emails are stored under different tabs such as primary, social, and promotion. When you click one of tab, it displays a list of emails under that tab.

In order to read an email, you just have to click on that email. Once you click a particular email, it gets opened.

The opened email may have some file attached with it. The attachments are shown at the bottom of the opened email with an option called **download attachment.**

#### **Replying Email**

After reading an email, you may have to reply that email. To reply an email, click **Reply** option shown at the bottom of the opened email.

Once you click on Reply, it will automatically copy the sender’s address in to the To field. Below the To field, there is a text box where you can type the message.

Once you are done with entering message, click Send button. It’s that easy. Your email is sent.

#### **Forwarding Email**

It is also possible to send a copy of the message that you have received along with your own comments if you want. This can be done using **forward** button available in mail client software.

The difference between replying and forwarding an email is that when you reply a message to a person who has send the mail but while forwarding you can send it to anyone.

When you receive a forwarded message, the message is marked with a > character in front of each line and **Subject:** field is prefixed with **Fw.**

#### **Deleting Email**

If you don’t want to keep email into your inbox, you can delete it by simply selecting the message from the message list and clicking **delete** or pressing the appropriate command.

Some mail clients offers the deleted mails to be stored in a folder called deleted items or trash from where you can recover a deleted email.

### E-mail Security

#### **E-mail Hacking**

Email hacking can be done in any of the following ways:

* Spam
* Virus
* Phishing

#### **Spam**

E-mail spamming is an act of sending **Unsolicited Bulk E-mails (UBI)** which one has not asked for. Email spams are the junk mails sent by commercial companies as an advertisement of their products and services.

#### **Virus**

Some emails may incorporate with files containing malicious script which when run on your computer may lead to destroy your important data.

#### **Phishing**

Email phishing is an activity of sending emails to a user claiming to be a legitimate enterprise. Its main purpose is to steal sensitive information such as usernames, passwords, and credit card details.

Such emails contains link to websites that are infected with malware and direct the user to enter details at a fake website whose look and feels are same to legitimate one.

#### **E-mail Spamming and Junk Mails**

Email spamming is an act of sending Unsolicited Bulk E-mails (UBI) which one has not asked for. Email spams are the junk mails sent by commercial companies as an advertisement of their products and services.

Spams may cause the following problems:

* It floods your e-mail account with unwanted e-mails, which may result in loss of important e-mails if inbox is full.
* Time and energy is wasted in reviewing and deleting junk emails or spams.
* It consumes the bandwidth that slows the speed with which mails are delivered.
* Some unsolicited email may contain virus that can cause harm to your computer.

##### Blocking Spams

Following ways will help you to reduce spams:

* While posting letters to newsgroups or mailing list, use a separate e-mail address than the one you used for your personal e-mails.
* Don’t give your email address on the websites as it can easily be spammed.
* Avoid replying to emails which you have received from unknown persons.
* Never buy anything in response to a spam that advertises a product.

#### **E-mail Cleanup and Archiving**

In order to have light weighted Inbox, it’s good to archive your inbox from time to time. Here I will discuss the steps to clean up and archive your Outlook inbox.

* Select **File** tab on the mail pane.
* Select **Cleanup Tools** button on account information screen.
* Select **Archive** from cleanup tools drop down menu.
* Select **Archive this folder and all subfolders** option and then click on the folder that you want to archive. Select the date from the **Archive items older than:** list. Click **Browse** to create new **.pst** file name and location. Click **OK.**

### E-mail Providers

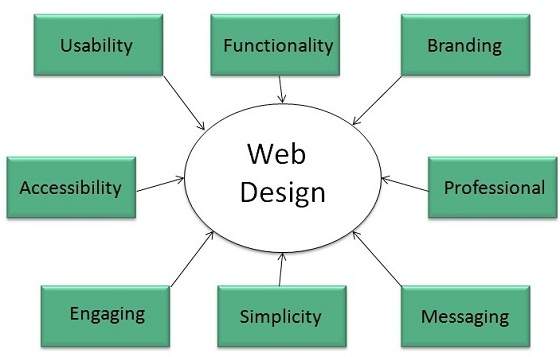
There are several email service providers available in the market with their enabled features such as sending, receiving, drafting, storing an email and much more.

The following table shows the popular email service providers:

|  |  |
| --- | --- |
| **S.N.** | **Service and Description** |
| 1. | **Gmail** Gmail is an email service that allows users to collect all the messages. It also offers approx 7 GB of free storage. |
| 2. | **Hotmail** Hotmail offers free email and practically unlimited storage accessible on web. |
| 3. | **Yahoo Mail** Yahoo Mail offers unlimited storage, SMS texting, social networking and instant messaging to boot. |
| 4. | **iCloud Mail** iCloud Mail offers ample storage, IMAP access, and an elegantly functional web application. |
| 5. | **ATM Mail** ATM Mail is a free email service with good spam protection. |
| 6. | **Mail.com** and **GMX Mail** Mail.com and GMX Mail offers reliable mail service with unlimited online storage. |
| 7. | **Shortmail** Shortmail offers easy and fast email service but with limited 500 characters per message. |
| 8. | **Inbox.com** Inbox.com offers 5 GB of free online storage. IMAP is not supported by Inbox.com |
| 9. | **Facebook Messages** Facebook Messages includes the message conversation. |
| 10. | **My Way Mail** My Way Mail offers clean and fast free email service but lacks in secure messaging. |

## Website Designing

Web designing has direct link to visual aspect of a web site. Effective web design is necessary to communicate ideas effectively.



Web desinging is subset of web development. However these terms are used interchangeably.

Key Points

Design Plan should include the following:

* Details about information architecture.
* Planned structure of site.
* A site map of pages

### Wireframe

**Wireframe** refers to a visual guide to appearance of web pages. It helps to define structre of web site, linking between web pages and layout of visual elements.

Following things are included in a wireframe:

* Boxes of primary graphical elements
* Placement of headlines and sub headings
* Simple layout structure
* Calls to action
* Text blocks

Wireframe can be created using program like Visio but you can also use a pen and paper.

### Web Designing Tools

Here is the list of tools that can be used to make effective web designs:

* Photoshop CC
* Illustrator CC
* Coda 2
* OmniGraffle
* Sublime Text
* GitHub
* Pen and Parer
* Vim
* Imageoptim
* Sketch 3
* Heroku
* Axure
* Hype 2
* Slicy
* Framer.js
* Image Alpha
* Emmet LiveStyle
* Hammer
* Icon Slate
* JPEGmini Lite
* BugHerd

### Web Page Anatomy

A web site includes the following components:

### Containing Block

**Container** can be in the form of page’s body tag, an all containing div tag. Without container there would be no place to put the contents of a web page.

### Logo

**Logo** refers to the identity of a website and is used across a company’s various forms of marketing such as business cards, letterhead, brouchers and so on.

### Naviagation

The site’s **navigation system** should be easy to find and use. Oftenly the anvigation is placed rigth at the top of the page.

### Content

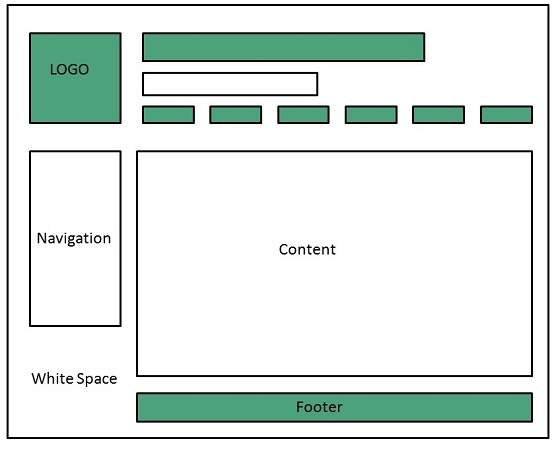
The content on a web site should be relevant to the purpose of the web site.

### Footer

**Footer** is located at the bottom of the page. It usually contains copyright, contract and legal information as well as few links to the main sections of the site.

### Whitespace

It is also called as **negative space** and refers to any area of page that is not covered by type or illustrations.



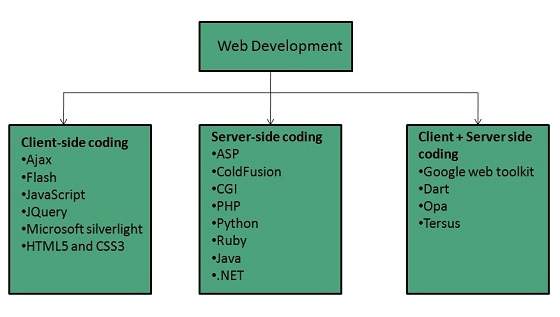
### Web design Mistakes

One should be aware of the following common mistakes should always keep in mind:

* Website not working in any other browser other internet explorer.
* Using cutting edge technology for no good reason
* Sound or video that starts automatically
* Hidden or disguised navigation
* 100% flash content.

## Web development

**Web development** refers to building website and deploying on the web. Web development requires use of scripting languages both at the server end as well as at client end.

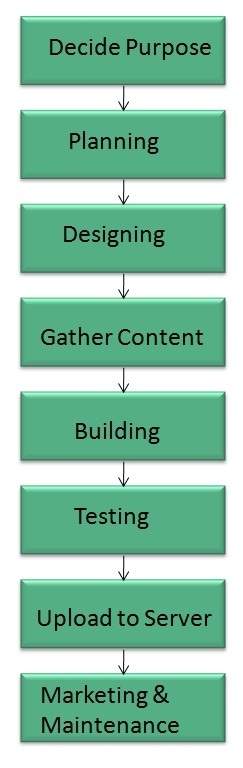


Before developing a web site once should keep several aspects in mind like:

* What to put on the web site?
* Who will host it?
* How to make it interactive?
* How to code it?
* How to create search engine friendly web site?
* How to secure the source code frequently?
* Will the web site design display well in different browsers?
* Will the navigation menus be easy to use?
* Will the web site loads quickly?
* How easily will the site pages print?
* How easily will visitors find important details specific to the web site?
* How effectively the style sheets be used on your web sites?

### Web Development Process

Web development process includes all the steps that are good to take to build an attractive, effective and responsive website. These steps are shown in the following diagram:



### Web development tools

Web development tools helps the developer to test and debug the web sites. Now a days the web development tooll come with the web browsers as add-ons. All web browsers have built in tools for this purpose.

Thsese tools allow the web developer to use HTML, CSS and JavaScript etc.. These are accessed by hovering over an item on a web page and selecting the “Inspect Element” from the context menu.

#### **Featues**

Following are the common featuers that every web development tool exhibits:

##### HTML and the DOM

HTML and DOM viewer allows you to see the DOM as it was rendered. It also allows to make changes to HTML and DOM and see the changes reflected in the page after the change is made.

##### Web Page Assests, Resources, and Network Information

Web development tools also helps to inspect the resources that are loaded and available on the web page.

##### Profiing and Auditing

**Profiling** refers to get information about the performance of a web page or web application and **Auditing** provides developers suggestions, after analyzing a page, for optimizations to decerease page load time and increase responsiveness.

### Skills Required

For being a successful web developer, one should possess the following skills:

* Understanding of client and server side scripting.
* Creating, editing and modifying templates for a CMS or web development framework.
* Testing cross browser inconsistencies.
* Conducting observational user testing.
* Testing for compliance to specified standards such as accessibility standards in the client region.
* Programming interaction with javaScript, PHP, and Jquery etc.

## Websites Hosting

**Web hosting** is a service of providing online space for storage of web pages. These web pages are made available via **World Wide Web.** The companies which offer website hosting are known as **Web hosts.**

The servers on which web site is hosted remain switched on 24 x7. These servers are run by web hosting companies. Each server has its own IP address. Since IP addresses are difficult to remember therefore, webmaster points their domain name to the IP address of the server their website is stored on.

It is not possible to host your website on your local computer, to do so you would have to leave your computer on 24 hours a day. This is not practical and cheaper as well. This is where web hosting companies comes in.

### Types of Hosting

The following table describes different types of hosting that can be availed as per the need:

|  |  |
| --- | --- |
| **S.N.** | **Hosting Description** |
| 1. | **Shared Hosting** In shared hosting, the hosting company puts thousand of website on the same physical server. Each customer has their own allocation of physical web space and a set of bandwidth limit. As all websites share same physical memory, MYSQL server and Apache server, one website on the server experiencing high traffic load will affect performance of all websites on the server. |
| 2. | **Virtual Private Server (VPS)** It is also known as Virtual Dedicated Server. It is a server which is partitioned into smaller servers. In this customer is given their own partition, which is installed with its own operating system. Unlike shared hosting, VPS doesn’t share memory or processor time rather it allocates certain amount of memory and CPU to use which means that any problem on a VPS partition on the same drive will not affect other VPS customers. |
| 3. | **Dedicated Server** In this kind of hosting, single dedicated server is setup for just one customer. It is commonly used by the businesses that need the power, control and security that a dedicated server offers. |
| 4. | **Reseller Hosting** A reseller acts as a middle man and sells hosting space of someone else’s server. |
| 5. | **Grid Hosting** Instead of utilizing one server, Grid Hosting spreads resources over a large number of servers. It is quite stable and flexible. The servers can be added or taken away from the grid without crashing the system. |

### Web Hosting Companies

Following are the several companies offering web hosting service:

|  |  |
| --- | --- |
| **S.N.** | **Hosting Company** |
| 1. | Blue Host |
| 2. | Go Daddy |
| 3. | Host Gator |
| 4. | just Host |
| 5. | Laughing Squid |
| 6. | Hivelocity |
| 7. | liquid Web |
| 8. | Media TempleServInt |
| 9. | Wired Tree |
| 10. | Wild West Domains |
| 11. | Wix |
| 12. | WIPL |
| 13. | Big Rock |

## Website Security Considerations

Websites are always to prone to security risks. **Cyber crime** impacts your business by hacking your website. Your website is then used for hacking assaults that install malicious software or malware on your visitor’s computer.

Hackers may also steal important customer data such as credit card information, destroy your business and propagate illegal content to your users.

### Security Considerations

#### **Updated Software**

It is mandatory to keep you software updated. It plays vital role in keeping your website secure.

#### **SQL Injection**

It is an attempt by the hackers to manipulate your database. It is easy to insert rogue code into your query that can be used to manipulate your database such as change tables, get information or delete data.

#### **Cross Site Scripting (XSS)**

It allows the attackers to inject client side script into web pages. Therefore, while creating a form It is good to endure that you check the data being submitted and encode or strip out any HTML.

#### **Error Messages**

You need to be careful about how much information to be given in the error messages. For example, if the user fails to log in the error message should not let the user know which field is incorrect: username or password.

#### **Validation of Data**

The validation should be performed on both server side and client side.

#### **Passwords**

It is good to enforce password requirements such as of minimum of eight characters, including upper case, lower case and special character. It will help to protect user’s information in long run.

#### **Upload files**

The file uploaded by the user may contain a script that when executed on the server opens up your website.

#### **SSL**

It is good practice to use SSL protocol while passing personal information between website and web server or database.

## World Wide Web

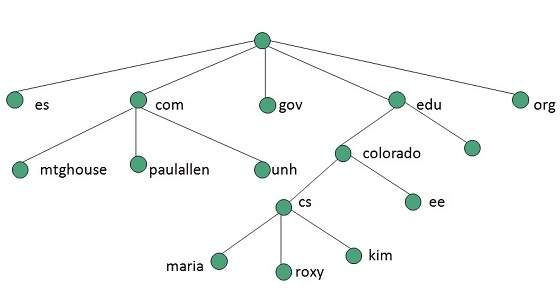
A technical definition of the World Wide Web is : all the resources and users on the Internet that are using the Hypertext Transfer Protocol (HTTP).

A broader definition comes from the organization that Web inventor **Tim Berners-Lee** helped found, the **World Wide Web Consortium (W3C).**

The World Wide Web is the universe of network-accessible information, an embodiment of human knowledge.

In simple terms, The World Wide Web is a way of exchanging information between computers on the Internet, tying them together into a vast collection of interactive multimedia resources.

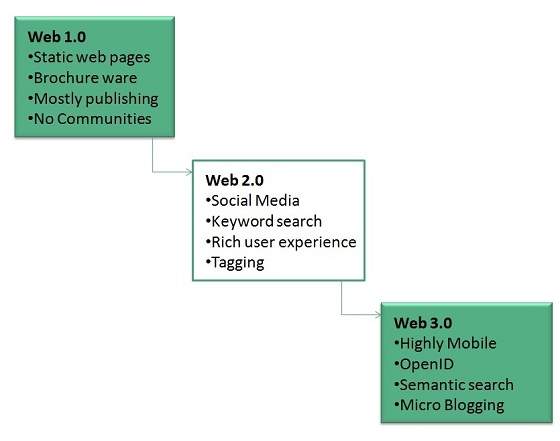
**Internet** and **Web** is not the same thing: Web uses internet to pass over the information.



### WWW Evolution

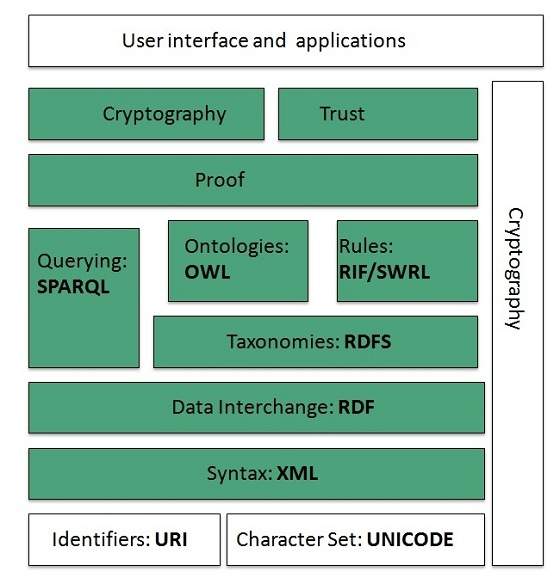
**World Wide Web** was created by **Timothy Berners Lee** in 1989 at **CERN** in **Geneva.** World Wide Web came into existence as a proposal by him, to allow researchers to work together effectively and efficiently at **CERN.** Eventually it became **World Wide Web.**

The following diagram briefly defines evolution of World Wide Web:



### WWW Architecture

WWW architecture is divided into several layers as shown in the following diagram:



#### **Identifiers and Character Set**

**Uniform Resource Identifier (URI)** is used to uniquely identify resources on the web and **UNICODE** makes it possible to built web pages that can be read and write in human languages.

#### **Syntax**

**XML (Extensible Markup Language)** helps to define common syntax in semantic web.

#### **Data Interchange**

**Resource Description Framework (RDF)** framework helps in defining core representation of data for web. RDF represents data about resource in graph form.

#### **Taxonomies**

**RDF Schema (RDFS)** allows more standardized description of **taxonomies** and other **ontological** constructs.

#### **Ontologies**

**Web Ontology Language (OWL)** offers more constructs over RDFS. It comes in following three versions:

* OWL Lite for taxonomies and simple constraints.
* OWL DL for full description logic support.
* OWL for more syntactic freedom of RDF

#### **Rules**

**RIF** and **SWRL** offers rules beyond the constructs that are available from **RDFs** and **OWL.** Simple Protocol and **RDF Query Language (SPARQL)** is SQL like language used for querying RDF data and OWL Ontologies.

#### **Proof**

All semantic and rules that are executed at layers below Proof and their result will be used to prove deductions.

#### **Cryptography**

**Cryptography** means such as digital signature for verification of the origin of sources is used.

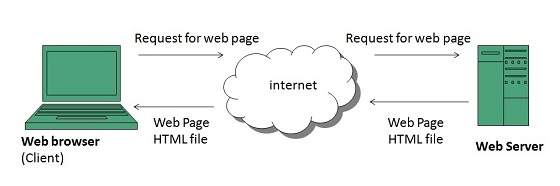
#### **User Interface and Applications**

On the top of layer **User interface and Applications** layer is built for user interaction.

### WWW Operation

**WWW** works on client- server approach. Following steps explains how the web works:

1. User enters the URL (say, **http://www.tutorialspoint.com**) of the web page in the address bar of web browser.
2. Then browser requests the Domain Name Server for the IP address corresponding to www.tutorialspoint.com.
3. After receiving IP address, browser sends the request for web page to the web server using HTTP protocol which specifies the way the browser and web server communicates.
4. Then web server receives request using HTTP protocol and checks its search for the requested web page. If found it returns it back to the web browser and close the HTTP connection.
5. Now the web browser receives the web page, It interprets it and display the contents of web page in web browser’s window.



### WWW Future

There had been a rapid development in field of web. It has its impact in almost every area such as education, research, technology, commerce, marketing etc. So the future of web is almost unpredictable.

Apart from huge development in field of WWW, there are also some technical issues that W3 consortium has to cope up with.

#### **User Interface**

Work on higher quality presentation of 3-D information is under deveopment. The W3 Consortium is also looking forward to enhance the web to full fill requirements of global communities which would include all regional languages and writing systems.

#### **Technology**

Work on privacy and security is under way. This would include hiding information, accounting, access control, integrity and risk management.

#### **Architecture**

There has been huge growth in field of web which may lead to overload the internet and degrade its performance. Hence more better protocol are required to be developed.

## Web Browsers

**web Browser** is an application software that allows us to view and explore information on the web. User can request for any web page by just entering a URL into address bar.

Web browser can show text, audio, video, animation and more. It is the responsibility of a web browser to interpret text and commands contained in the web page.

Earlier the web browsers were text-based while now a days graphical-based or voice-based web browsers are also available. Following are the most common web browser available today:

|  |  |
| --- | --- |
| **Browser** | **Vendor** |
| Internet Explorer | Microsoft |
| Google Chrome | Google |
| Mozilla Firefox | Mozilla |
| Netscape Navigator | Netscape Communications Corp. |
| Opera | Opera Software |
| Safari | Apple |
| Sea Monkey | Mozilla Foundation |
| K-meleon | K-meleon |

### Architecture

There are a lot of web browser available in the market. All of them interpret and display information on the screen however their capabilities and structure varies depending upon implementation. But the most basic component that all web browser must exhibit are listed below:

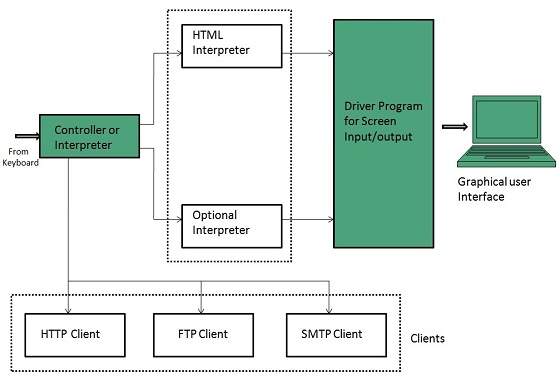
* Controller/Dispatcher
* Interpreter
* Client Programs

**Controller** works as a control unit in CPU. It takes input from the keyboard or mouse, interpret it and make other services to work on the basis of input it receives.

**Interpreter** receives the information from the controller and execute the instruction line by line. Some interpreter are mandatory while some are optional For example, HTML interpreter program is mandatory and java interpreter is optional.

**Client Program** describes the specific protocol that will be used to access a particular service. Following are the client programs tat are commonly used:

* HTTP
* SMTP
* FTP
* NNTP
* POP



## Web Server

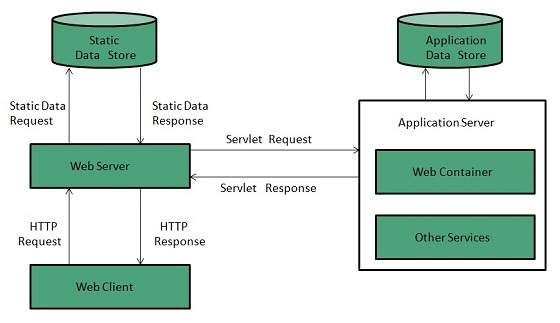
**Web server** is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

Web site is collection of web pages whileweb server is a software that respond to the request for web resources.

### Web Server Working

Web server respond to the client request in either of the following two ways:

* Sending the file to the client associated with the requested URL.
* Generating response by invoking a script and communicating with database



**Key Points**

* When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.
* If the requested web page is not found, web server will the send an **HTTP response:Error 404 Not found.**
* If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

### Architecture

Web Server Architecture follows the following two approaches:

* Concurrent Approach
* Single-Process-Event-Driven Approach.

#### **Concurrent Approach**

Concurrent approach allows the web server to handle multiple client requests at the same time. It can be achieved by following methods:

* Multi-process
* Multi-threaded
* Hybrid method.

#### **Multi-processing**

In this a single process (parent process) initiates several single-threaded child processes and distribute incoming requests to these child processes. Each of the child processes are responsible for handling single request.

It is the responsibility of parent process to monitor the load and decide if processes should be killed or forked.

#### **Multi-threaded**

Unlike Multi-process, it creates multiple single-threaded process.

#### **Hybrid**

It is combination of above two approaches. In this approach multiple process are created and each process initiates multiple threads. Each of the threads handles one connection. Using multiple threads in single process results in less load on system resources.

### Examples

Following table describes the most leading web servers available today:

|  |  |
| --- | --- |
| **S.N.** | **Web Server Descriptino** |
| 1 | **Apache HTTP Server** This is the most popular web server in the world developed by the Apache Software Foundation. Apache web server is an open source software and can be installed on almost all operating systems including Linux, UNIX, Windows, FreeBSD, Mac OS X and more. About 60% of the web server machines run the Apache Web Server. |
| 2. | **Internet Information Services (IIS)** The Internet Information Server (IIS) is a high performance Web Server from Microsoft. This web server runs on Windows NT/2000 and 2003 platforms (and may be on upcoming new Windows version also). IIS comes bundled with Windows NT/2000 and 2003; Because IIS is tightly integrated with the operating system so it is relatively easy to administer it. |
| 3. | **Lighttpd** The lighttpd, pronounced lighty is also a free web server that is distributed with the FreeBSD operating system. This open source web server is fast, secure and consumes much less CPU power. Lighttpd can also run on Windows, Mac OS X, Linux and Solaris operating systems. |
| 4. | **Sun Java System Web Server** This web server from Sun Microsystems is suited for medium and large web sites. Though the server is free it is not open source. It however, runs on Windows, Linux and UNIX platforms. The Sun Java System web server supports various languages, scripts and technologies required for Web 2.0 such as JSP, Java Servlets, PHP, Perl, Python, and Ruby on Rails, ASP and Coldfusion etc. |
| 5. | **Jigsaw Server** Jigsaw (W3C's Server) comes from the World Wide Web Consortium. It is open source and free and can run on various platforms like Linux, UNIX, Windows, and Mac OS X Free BSD etc. Jigsaw has been written in Java and can run CGI scripts and PHP programs. |

## Proxy Server

**Proxy server** is an intermediary server between client and the internet. Proxy servers offers the following basic functionalities:

* Firewall and network data filtering.
* Network connection sharing
* Data caching

Proxy servers allow to hide, conceal and make your network id anonymous by hiding your IP address.

### Purpose of Proxy Servers

Following are the reasons to use proxy servers:

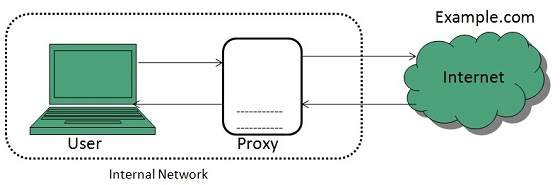
* Monitoring and Filtering
* Improving performance
* Translation
* Accessing services anonymously
* Security

### Type of Proxies

Following table briefly describes the type of proxies:

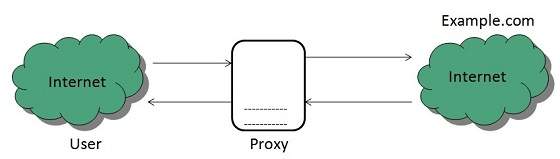
#### **Forward Proxies**

In this the client requests its internal network server to forward to the internet.



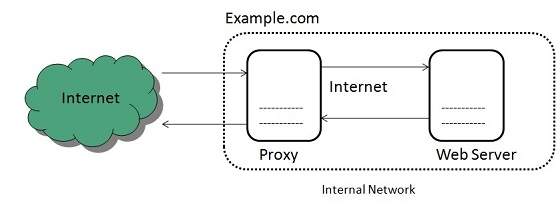
#### **Open Proxies**

Open Proxies helps the clients to conceal their IP address while browsing the web.



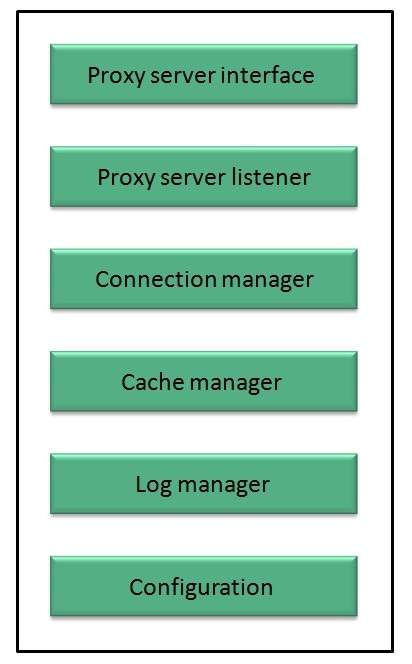
#### **Reverse Proxies**

In this the requests are forwarded to one or more proxy servers and the response from the proxy server is retrieved as if it came directly from the original Server.



### Architecture

The proxy server architecture is divided into several modules as shown in the following diagram:



#### **Proxy user interface**

This module controls and manages the user interface and provides an easy to use graphical interface, window and a menu to the end user. This menu offers the following functionalities:

* Start proxy
* Stop proxy
* Exit
* Blocking URL
* Blocking client
* Manage log
* Manage cache
* Modify configuration

#### **Proxy server listener**

It is the port where new request from the client browser is listened. This module also performs blocking of clients from the list given by the user.

#### **Connection Manager**

It contains the main functionality of the proxy server. It performs the following functions:

* It contains the main functionality of the proxy server. It performs the following functions:
* Read request from header of the client.
* Parse the URL and determine whether the URL is blocked or not.
* Generate connection to the web server.
* Read the reply from the web server.
* If no copy of page is found in the cache then download the page from web server else will check its last modified date from the reply header and accordingly will read from the cache or server from the web.
* Then it will also check whether caching is allowed or not and accordingly will cache the page.

#### **Cache Manager**

This module is responsible for storing, deleting, clearing and searching of web pages in the cache.

#### **Log Manager**

This module is responsible for viewing, clearing and updating the logs.

#### **Configuration**

This module helps to create configuration settings which in turn let other modules to perform desired configurations such as caching.

## Search Engines

**Search Engine** refers to a huge database of internet resources such as web pages, newsgroups, programs, images etc. It helps to locate information on World Wide Web.

User can search for any information by passing query in form of keywords or phrase. It then searches for relevant information in its database and return to the user.



### Search Engine Components

Generally there are three basic components of a search engine as listed below:

1. Web Crawler
2. Database
3. Search Interfaces

#### **Web crawler**

It is also known as **spider** or **bots.** It is a software component that traverses the web to gather information.

#### **Database**

All the information on the web is stored in database. It consists of huge web resources.

#### **Search Interfaces**

This component is an interface between user and the database. It helps the user to search through the database.

### Search Engine Working

Web crawler, database and the search interface are the major component of a search engine that actually makes search engine to work. Search engines make use of Boolean expression AND, OR, NOT to restrict and widen the results of a search. Following are the steps that are performed by the search engine:

* The search engine looks for the keyword in the index for predefined database instead of going directly to the web to search for the keyword.
* It then uses software to search for the information in the database. This software component is known as web crawler.
* Once web crawler finds the pages, the search engine then shows the relevant web pages as a result. These retrieved web pages generally include title of page, size of text portion, first several sentences etc.

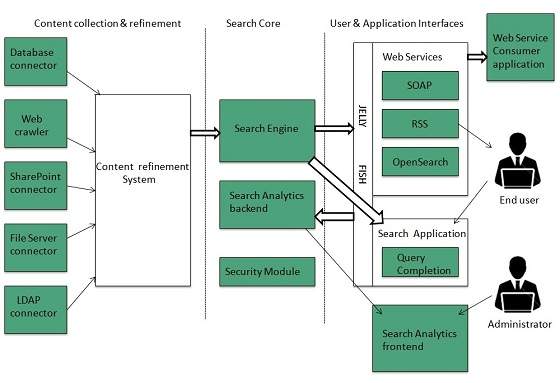
These search criteria may vary from one search engine to the other. The retrieved information is ranked according to various factors such as frequency of keywords, relevancy of information, links etc.

* User can click on any of the search results to open it.

### Architecture

The search engine architecture comprises of the three basic layers listed below:

* Content collection and refinement.
* Search core
* User and application interfaces



## Internet Collaboration

### Online Chatting

**Online chatting** is a text-based communication between two or more people over the network. In this, the text message is delivered in real time and people get immediate response.

**Talkomatic** was the world first online chat system. It was developed by **Doug Brown** and **David R. Woolley** in 1973.

#### **Chat Etiquette**

**Chat etiquette** defines rules that are supposed to be followed while online chatting:

* Avoid chat slang
* Try to spell all words correctly.
* Don’t write all the words in capital.
* Don’t send other chat users private messages without asking them.
* Abide by the rules created by those running the chat.
* Use emoticons to let other person know your feelings and expressions.

#### **Web Based Chat Services**

Following web sites offers browser based chat services:

|  |  |
| --- | --- |
| **Website** | **Description** |
| Facebook | It was founded by **Mark Zuckerberg** with his college roommates at Harvard university. Facebook lets the user to create personal profile, post status & photos, and receive notifications. |
| eBuddy | It is an instant messaging service. It supports multiprotocol instant messaging clients. |
| Convore | It offers real time web based chat system. |
| MeBeam | It offers video based chat between the clients to create video conferencing rooms for up to 16 people. |
| Yahoo! Messenger | It offers PC-PC, PC-phone, Phone-to-PC, file transfer, webcam hosting, text messaging service etc. |
| WhatsApp | It is an instant messaging service application available on smart phones. |
| Gmail | It offers instant chatting, sending and receiving mails, and video calling services. |

### Instant Messaging (IM)

**Instant messaging** is a software utility that allows IM users to communicate by sending text messages, files, and images. Some of the IMs also support voice and video calls.

|  |  |
| --- | --- |
| **Application** | **Description** |
| Nimbuzz | It is native iPhone app. It supports voice and video chats, file sharing, and group chats with panache. |
| eBuddy | eBuddy IM helps to have all your buddies from multiple IM accounts in one single list. |
| Imo.in | It has capability to link all your IM accounts together. You can log on to all of your IM accounts by just logging into imo.in. |
| MeBeam | It offers video based chat between the clients to create video conferencing rooms for up to 16 people. |
| Yahoo! Messenger | It offers PC-PC, PC-phone, Phone-to-PC, file transfer, webcam hosting, text messaging service etc. |
| GoogleTalk | It is an IM by Google and one of the most widely used. |
| Lync | Lync is an IM developed by Microsoft. It is widely used in corporate sector for internal and external communication as well. |

### Internet Relay Chat (IRC)

**Internet Relay Chat** is a protocol developed by Oikarinen in August 1988. It defines set of rules for communication between client and server by some communication mechanism such as chat rooms, over the internet.

IRC consist of separate networks of IRC servers and machines. These allow IRC clients to connect to IRC. IRC client runs a program client to connect to a server on one of the IRC nets. After connecting to IRC server on IRC network, user can join with one or more channels and converse over there.

### Video Conferencing

**Video conferencing** or **Video teleconferencing** is a method of communicating by two-way video and audio transmission with help of telecommunication technologies.

#### **Modes of Video Conferencing**

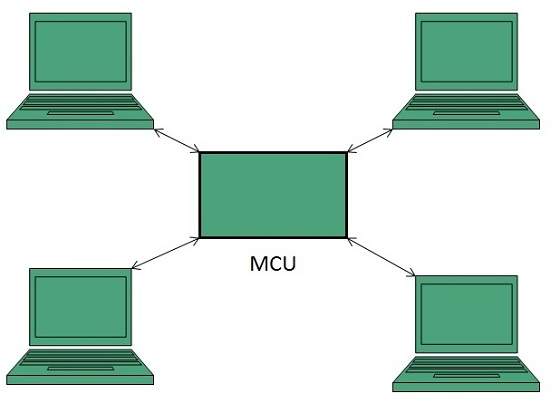
##### Point-to-Point

This mode of conferencing connects two locations only.



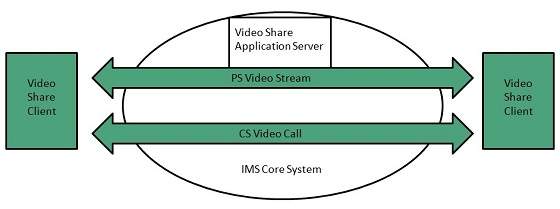
##### Multi-point

This mode of conferencing connects more than two locations through **Multi-point Control Unit (MCU).**



### Video Sharing

**Video sharing** is an **IP Multimedia System (IMS)** service that allows user to switch voice calls to unidirectional video streaming session. The video streaming session can be initiated by any of the parties. Moreover, the video source can be the camera or the pre-recorded video clip.



## Mailing List

In order to send same email to a group of people, an electron list is created which is know as Mailing List. It is the list server which receives and distributes postings and automatically manages subscriptions.

Mailing list offers a forum, where users from all over the globe can answer questions and have them answered by others with shared interests.

### Types of Mailing List

Following are the various types of mailing lists:

#### **Response List**

It contains the group of people who have responsed to an offer in some way. These people are the customers who have shown interest in specific product or service.

#### **Compiled List**

The compiled list is prepared by collecting information from various sources such as surveys, telemarketing etc.

#### **Announcements**

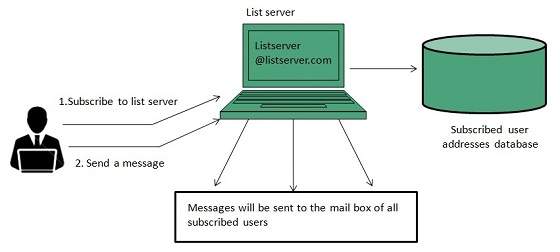
These lists are created for sending out coupans , new product announcements and other offers to the customers.

#### **Discussion List**

This list is created for sharing views on a specific topic suchas computer, environment , healt, education etc.

### How does mailing list work?

Before joining a mailing list, it is mandatory to subscribe to it. Once you are subscribed, your message will be sent to all the persons who have subscribed to the list. Similarly if any subscriber posts a message, then it will be received by all subscribers of the list.



### Finding Mailing List

There are a number of websites are available to maintain database of publically accessible mailing list. Some of these are:

* http://tile.net./lists
* http://lists.com
* http://topica.com
* http://isoft.com/lists/list-q.html

**Mailing lists** can also be found using Google website. In Google, move to directory and the follow: **Computers > Internet >Mailing List > Directories.**

### Subscribing to Mailing List

To subscribe to a list, you need to send an email message to the administrative address mailing list containing one or more commands. For example, if you want to subscribe to Harry Potter list in gurus.com where name of the list server us Majordomo, then you have to send email to majordom@gurus.com containing the text, Subscribe harry potter in its body.

After sending the email, you will receive a confirmation email for your subscription. This email will include list of commands that will help you to perform various operations such as unsubscribing, receiving acknowledgement, and find out what list you are subscribed to.

There are many list servers available, each having its own commands for subscribing to the list. Some of them are described in the following table:

|  |  |  |
| --- | --- | --- |
| **List Server** | **Command** | **Example** |
| LISTSERV | subscribe listname yourname | subscribe commdhtml rahul |
| Majordomo | subscribe listname | subscribe commdhtml |
| ListProc | subscribe listname yourname | subscribe commdhtml rahul |

## Usenet (USEr NETwork)

Like mailing lists Usenet is also a way of sharing information. It was started by Tom Truscott and Jim Ellis in 1979. Initially it was limited to two sites but today there are thousands of Usenet sites involving millions of people.

**Usenet** is a kind of discussion group where people can share views on topic of their interest. The article posted to a newsgroup becomes available to all readers of the newsgroup.

### Mailing list vs. Newsgroup

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Mailing List** | **Newsgroup** |
| 1. | Messages are delivered to individual mailboxes of subscribed member of group. | Messages are not posted to individual mailboxes but can be viewed by anyone who has subscribed to that newsgroup. |
| 2. | Working with mailing list is easier than newsgroup. It is easy to compose and receive emails. | Working with a particular newsgroup requires proper knowledge of that newsgroup. |
| 3. | In order to send or receive mails, you required an email program. | It requires a newsgroup reader. |
| 4. | Messages are delivered to certain group of people. | Messages are available to public. |
| 5. | Mailing list does not support threaded discussion. | Newsgroup supports threaded discussion. |
| 6. | Messages delivered to listed subscribers can not be cancelled. | Article posted on a newsgroup can be cancelled. |

## Online Education

There are several forms of online education available as discussed below:

### Online Training

Online Training is a form of distance learning in which educational information is delivered through internet. There are many online applications. These applications vary from simple downloadable content to structured programs.

### Online certification

It is also possible to do online certification on specialized courses which add value to your qualification. Many companies offer online certification on a number of technologies.

There are three types of online certification as listed below:

* Corporate
* Product-specific
* Profession-wide

**Corporate certifications** are made by small organizations for internal purposes.

**Product-specific certifications** target at developing and recognizing adeptness with regard to particular product.

**Profession wide certification** aims at recognizing expertise in particular profession.

## Online Seminar

**Online seminar** is the one which is conducted over the internet. It is a live seminar and allows the attendees to ask questions via Q&A panel onscreen.

Online seminar just requires a computer with internet connection, headphones, speakers, and authorization to attend it.

### Webinar

**Webinar** is a web based seminar or workshop in which presentation is delivered over the web using conferencing software. The audio part of webinar is delivered through teleconferencing.

### Conferencing

**Online conferencing** is also a kind of online seminar in which two or more people are involved. It is also performed over the internet. It allows the business persons to do meeting online.

## Social Networking

Social Networking refers to grouping of individuals and organizations together via some medium, in order to share thoughts, interests, and activities.

There are several web based social network services are available such as facebook, twitter, linkedin, Google+ etc. which offer easy to use and interactive interface to connect with people with in the country an overseas as well. There are also several mobile based social networking services in for of apps such as Whatsapp, hike, Line etc.

### Available Social networking Services

The following table describes some of the famous social networking services provided over web and mobile:

|  |  |
| --- | --- |
| **S.N.** | **Service Description** |
| 1. | **Facebook** Allows to share text, photos, video etc. It also offers interesting online games. |
| 2. | **Google+** It is pronounced as Google Plus. It is owned and operated by Google. |
| 3. | **Twitter** Twitter allows the user to send and reply messages in form of tweets. These tweets are the small messages, generally include 140+ characters. |
| 4. | **Faceparty** Faceparty is a UK based social networking site. It allows the users to create profiles and interact with each other using forums messages. |
| 5. | **Linkedin** Linkedin is a business and professional networking site. |
| 6. | **Flickr** Flickr offers image hosting and video hosting. |
| 7. | **Ibibo** Ibibo is a talent based social networking site. It allows the users to promote one’s self and also discover new talent. |
| 8. | **Whatsapp** It is a mobile based messaging app. It allows to send text, video, and audio messages |
| 9. | **Line** It is same as whatsapp. Allows to make free calls and messages. |
| 10. | **Hike** It is also mobile based messager allows to send messages and exciting emoticons. |

## Internet Security

Internet security refers to securing communication over the internet. It includes specific security protocols such as:

* Internet Security Protocol (IPSec)
* Secure Socket Layer (SSL)

### Threats

Internet security threats impact the network, data security and other internet connected systems. Cyber criminals have evolved several techniques to threat privacy and integrity of bank accounts, businesses, and organizations.

Following are some of the internet security threats:

* Mobile worms
* Malware
* PC and Mobile ransomware
* Large scale attacks like Stuxnet that attempts to destroy infrastructure.
* Hacking as a Service
* Spam
* Phishing

### Email Phishing

**Email phishing** is an activity of sending emails to a user claiming to be a legitimate enterprise. Its main purpose is to steal sensitive information such as usernames, passwords, and credit card details.

Such emails contains link to websites that are infected with malware and direct the user to enter details at a fake website whose look and feels are same to legitimate one.

#### **What a phishing email may contain?**

Following are the symptoms of a phishing email:

##### Spelling and bad grammar

Most often such emails contain grammatically incorrect text. Ignore such emails, since it can be a spam.

##### Beware of links in email

Don’t click on any links in suspicious emails.

##### Threats

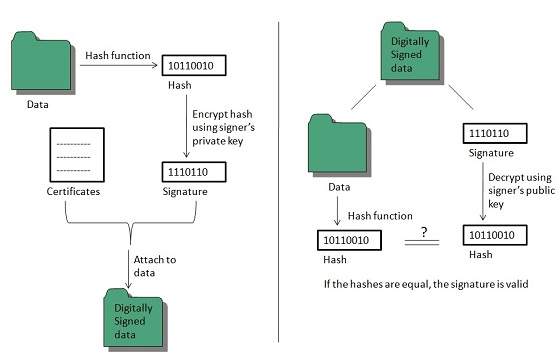
Such emails contain threat like “your account will be closed if you didn’t respond to an email message”.

##### Spoofing popular websites or companies

These emails contain graphics that appear to be connected to legitimate website but they actually are connected to fake websites.

## Digital Signature

**Digital signatures** allow us to verify the author, date and time of signatures, authenticate the message contents. It also includes authentication function for additional capabilities.



A digital signature should not only be tied to the signing user, but also to the message.

### Applications

There are several reasons to implement digital signatures to communications:

#### **Authentication**

Digital signatures help to authenticate the sources of messages. For example, if a bank’s branch office sends a message to central office, requesting for change in balance of an account. If the central office could not authenticate that message is sent from an authorized source, acting of such request could be a grave mistake.

#### **Integrity**

Once the message is signed, any change in the message would invalidate the signature.

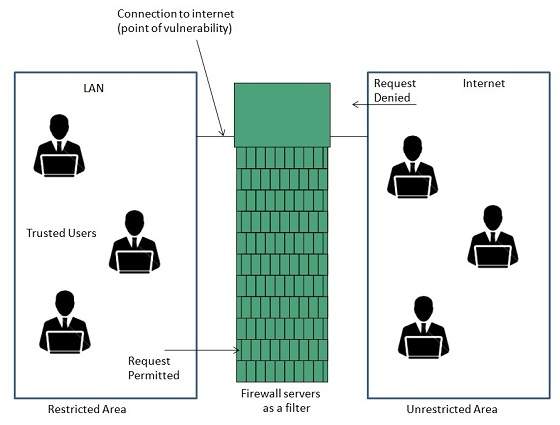
#### **Non-repudiation**

By this property, any entity that has signed some information cannot at a later time deny having signed it.

## Firewall Security

**Firewall** is a barrier between Local Area Network (LAN) and the Internet. It allows keeping private resources confidential and minimizes the security risks. It controls network traffic, in both directions.

The following diagram depicts a sample firewall between LAN and the internet. The connection between the two is the point of vulnerability. Both hardware and the software can be used at this point to filter network traffic.



There are two types of Firewall system: One works by using filters at the network layer and the other works by using proxy servers at the user, application, or network layer.

**Key Points**

* Firewall management must be addressed by both system managers and the network managers.
* The amount of filtering a firewall varies. For the same firewall, the amount of filtering may be different in different directions.

## HTML

**HTML** stands for **Hyper Text Markup Language.** It is a formatting language used to define the appearance and contents of a web page. It allows us to organize text, graphics, audio, and video on a web page.

**Key Points:**

* The word Hypertext refers to the text which acts as a link.
* The word markup refers to the symbols that are used to define structure of the text. The markup symbols tells the browser how to display the text and are often called tags.
* The word Language refers to the syntax that is similar to any other language.

HTML was created by **Tim Berners-Lee** at **CERN.**

### HTML Versions

The following table shows the various versions of HTML:

|  |  |
| --- | --- |
| **Version** | **Year** |
| HTML 1.0 | 1991 |
| HTML 2.0 | 1995 |
| HTML 3.2 | 1997 |
| HTML 4.0 | 1999 |
| XHTML | 2000 |
| HTML5 | 2012 |

### HTML Tags

**Tag** is a command that tells the web browser how to display the text, audio, graphics or video on a web page.

**Key Points:**

* Tags are indicated with pair of angle brackets.
* They start with a less than (<) character and end with a greater than (>) character.
* The tag name is specified between the angle brackets.
* Most of the tags usually occur in pair: the start tag and the closing tag.
* The start tag is simply the tag name is enclosed in angle bracket whereas the closing tag is specified including a forward slash (/).
* Some tags are the empty i.e. they don’t have the closing tag.
* Tags are not case sensitive.
* The starting and closing tag name must be the same. For example <b> hello </i> is invalid as both are different.
* If you don’t specify the angle brackets (<>) for a tag, the browser will treat the tag name as a simple text.
* The tag can also have attributes to provide additional information about the tag to the browser.

### Basic tags

The following table shows the Basic HTML tags that define the basic web page:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <html> </html> | Specifies the document as a web page. |
| <head> </head> | Specifies the descriptive information about the web documents. |
| <title> </title> | Specifies the title of the web page. |
| <body> </body> | Specifies the body of a web document. |

The following code shows how to use basic tags.

<html>

<head> Heading goes here…</head>

<title> Title goes here…</title>

<body> Body goes here…</body>

</html>

### Formatting Tags

The following table shows the HTML tags used for formatting the text:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <b> </b> | Specifies the text as bold. Eg. **this is bold text** |
| <em> </em> | It is a phrase text. It specifies the emphasized text. Eg. Emphasized text |
| <strong> </strong> | It is a phrase tag. It specifies an important text. Eg. **this is strong text** |
| <i> </i> | The content of italic tag is displayed in italic. Eg. *Italic text* |
| <sub> </sub> | Specifies the subscripted text. Eg. X1 |
| <sup> </sup> | Defines the superscripted text. Eg. X2 |
| <ins> </ins> | Specifies the inserted text. Eg. The price of pen is now 15. |
| <del> </del> | Specifies the deleted text. Eg. The price of pen is now 15. |
| <mark> </mark> | Specifies the marked text. Eg. It is raining |

### Table Tags

Following table describe the commonaly used table tags:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <table> </table> | Specifies a table. |
| <tr> </tr> | Specifies a row in the table. |
| <th> </th> | Specifies header cell in the table. |
| <td> </td> | Specifies the data in an cell of the table. |
| <caption> </caption> | Specifies the table caption. |
| <colgroup> </colgroup> | Specifies a group of columns in a table for formatting. |

### List tags

Following table describe the commonaly used list tags:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <ul> </ul> | Specifies an unordered list. |
| <ol> </ol> | Specifies an ordered list. |
| <li> </li> | Specifies a list item. |
| <dl> </dl> | Specifies a description list. |
| <dt> </dt> | Specifies the term in a description list. |
| <dd> </dd> | Specifies description of term in a description list. |

### Frames

**Frames** help us to divide the browser’s window into multiple rectangular regions. Each region contains separate html web page and each of them work independently.

A set of frames in the entire browser is known as frameset. It tells the browser how to divide browser window into frames and the web pages that each has to load.

The following table describes the various tags used for creating frames:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <frameset> </frameset> | It is replacement of the <body> tag. It doesn’t contain the tags that are normally used in <body> element; instead it contains the <frame> element used to add each frame. |
| <frame> </frame> | Specifies the content of different frames in a web page. |
| <base> </base> | It is used to set the default target frame in any page that contains links whose contents are displayed in another frame. |

### Forms

**Forms** are used to input the values. These values are sent to the server for processing. Forms uses input elements such as text fields, check boxes, radio buttons, lists, submit buttons etc. to enter the data into it.

The following table describes the commonly used tags while creating a form:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <form> </form> | It is used to create HTML form. |
| <input> </input> | Specifies the input field. |
| <textarea> </textarea> | Specifies a text area control that allows to enter multi-line text. |
| <label> </label> | Specifies the label for an input element. |

## CSS

**CSS** is acronym of **Cascading Style Sheets.** It helps to define the presentation of HTML elements as a separate file known as CSS file having **.css** extension.

CSS helps to change formatting of any HTML element by just making changes at one place. All changes made would be reflected automatically to all of the web pages of the website in which that element appeared.

### Embedding CSS into HTML

Following are the four methods to add CSS to HTML documents.

* Inline Style Sheets
* Embedded Style Sheets
* External Style Sheets
* Imported Style Sheets

#### **Inline Style Sheets**

**Inline Style Sheets** are included with HTML element i.e. they are placed inline with the element. To add inline CSS, we have to declare style attribute which can contain any CSS property.

**Syntax:**

<Tagname STYLE = “ Declaration1 ; Declaration2 “> …. </Tagname>

Let’s consider the following example using Inline Style Sheets:

<p style="color: blue; text-align: left; font-size: 15pt">

Inline Style Sheets are included with HTML element i.e. they are placed inline with the element.

To add inline CSS, we have to declare style attribute which can contain any CSS property.

</p>

#### **Embedded Style Sheets**

**Embedded Style** Sheets are used to apply same appearance to all occurrence of a specific element. These are defined in element by using the **<style>** element.

The **<style>** element must include **type** attribute. The value of **type** attribute specifies what type of syntax it includes when rendered by the browser.

**Syntax**

<head> <title> …. </title>

<style type =”text/css”>

…….CSS Rules/Styles….

</head>

Let’s consider the following example using Embedded Style Sheets:

<style type="text/css">

p {color:green; text-align: left; font-size: 10pt}

h1 { color: red; font-weight: bold}

</style>

#### **External Style Sheets**

**External Style Sheets** are the separate **.css** files that contain the CSS rules. These files can be linked to any HTML documents using <link> tag with rel attribute.

**Syntax:**

<head> <link rel= “stylesheet” type=”text/css” href= “url of css file”>

</head>

In order to create external css and link it to HTML document, follow the following steps:

* First of all create a CSS file and define all CSS rules for several HTML elements. Let’s name this file as external.css.

p {

Color: orange; text-align: left; font-size: 10pt;

}

h1 {

Color: orange; font-weight: bold;

}

* Now create HTML document and name it as **externaldemo.html.**

<html>

<head>

<title> External Style Sheets Demo </title>

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

</head>

<body>

<h1> External Style Sheets</h1>

<p>External Style Sheets are the separate .css files that contain the CSS rules.</p>

</body>

</html>

#### **Imported Style Sheets**

**Imported Style Sheets** allow us to import style rules from other style sheets. To import CSS rules we have to use @import before all the rules in a style sheet.

**Syntax:**

<head><title> Title Information </title>

<style type=”text/css”>

@import URL (cssfilepath)

… CSS rules…

</style>

</head>

Let’s consider the following example using Inline Style Sheets:

<html>

<head>

<title> External Style Sheets Demo </title>

<style>

@import url(external.css);

</style>

</head>

<body>

<h1> External Style Sheets</h1>

<p>External Style Sheets are the separate .css files that contain the CSS rules.</p>

</body>

</html>

## javaScript

**JavaScript** is a lightweight, interpreted programming language with object-oriented capabilities that allows you to build interactivity into otherwise static HTML pages.

JavaScript code is not compiled but translated by the translator. This translator is embedded into the browser and is responsible for translating javascript code.

JavaScript is:

* Lightweight, interpreted programming language.
* Designed for creating network-centric applications.
* Complementary to and integrated with Java.
* Complementary to and integrated with HTML
* Open and cross-platform

### JavaScript Statements

JavaScript statements are the commands to tell the browser to what action to perform. Statements are separated by semicolon (;).

JavaScript statement constitutes the JavaScript code which is translated by the browser line by line.

Example of JavaScript statement:

document.getElementById("demo").innerHTML = "Welcome";

Following table shows the various JavaScript Statements:

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Statement** | **Description** |
| 1. | switch case | A block of statements in which execution of code depends upon different cases. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a **default** condition will be used. |
| 2. | If else | The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally. |
| 3. | While | The purpose of a while loop is to execute a statement or code block repeatedly as long as expression is true. Once expression becomes false, the loop will be exited. |
| 4. | do while | Block of statements that are executed at least once and continues to be executed while condition is true. |
| 5. | for | Same as while but initialization, condition and increment/decrement is done in the same line. |
| 6. | for in | This loop is used to loop through an object's properties. |
| 7. | continue | The continue statement tells the interpreter to immediately start the next iteration of the loop and skip remaining code block. |
| 8. | break | The break statement is used to exit a loop early, breaking out of the enclosing curly braces. |
| 9. | function | A function is a group of reusable code which can be called anywhere in your programme. The keyword function is used to declare a function. |
| 10. | return | Return statement is used to return a value from a function. |
| 11. | var | Used to declare a variable. |
| 12. | try | A block of statements on which error handling is implemented. |
| 13. | catch | A block of statements that are executed when an error occur. |
| 14. | throw | Used to throw an error. |

### JavaScript Comments

JavaScript supports both C-style and C++-style comments, thus:

* Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
* Any text between the characters /\* and \*/ is treated as a comment. This may span multiple lines.
* JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.-->
* The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

Example:

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

<!--

// this is a comment. It is similar to comments in C++

/\*

\* This is a multiline comment in JavaScript

\* It is very similar to comments in C Programming

\*/

//-->

<script>

### JavaScript variable

Variables are referred as named containers for storing information. We can place data into these containers and then refer to the data simply by naming the container.

Rules to declare variable in JavaScript

* In JavaScript variable names are case sensitive i.e. a is different from A.
* Variable name can only be started with a underscore ( \_ ) or a letter (from a to z or A to Z), or dollar ( $ ) sign.
* Numbers (0 to 9) can only be used after a letter.
* No other special character is allowed in variable name.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword as follows:

<script type="text/javascript">

<!--

var money;

var name, age;

//-->

</script>

Variables can be initialized at time of declaration or after declaration as follows:

<script type="text/javascript">

<!--

var name = "Ali";

var money;

money = 2000.50;

//-->

</script>

### Javascript Data Type

There are two kinds of data types as mentioned below:

* Primitive Data Type
* Non Primitive Data Type

Primitive Data Types are shown in the following table:

|  |  |
| --- | --- |
| **S.N.** | **Datatype Description** |
| 1. | **String** Can contain groups of character as single value. It is represented in double quotes.E.g. var x= “tutorial”. |
| 2. | **Numbers** Contains the numbers with or without decimal. E.g. var x=44, y=44.56; |
| 3. | **Booleans** Contain only two values either true or false. E.g. var x=true, y= false. |
| 4. | **Undefined** Variable with no value is called Undefined. E.g. var x; |
| 5. | **Null** If we assign null to a variable, it becomes empty. E.g. var x=null; |

Following table contains Non primitive Data Types:

|  |  |
| --- | --- |
| **S.N.** | **Datatype Description** |
| 1. | **Array** Can contain groups of values of same type. E.g. var x={1,2,3,55}; |
| 2. | **Objects** Objects are stored in property and value pair. E.g. var rectangle = { length: 5, breadth: 3}; |

### JavaScript Functions

Function is a group of reusable statements (Code) that can be called any where in a program. In javascript function keyword is used to declare or define a function.

**Key Points:**

* To define a function use function keyword followed by functionname, followed by parentheses ().
* In parenthesis, we define parameters or attributes.
* The group of reusabe statements (code) is enclosed in curly braces {}. This code is executed whenever function is called.

**Syntax:**

function functionname (p1, p2) {

function coding…

}

### JavaScript Operators

Operators are used to perform operation on one, two or more operands. Operator is represented by a symbol such as +, =, \*, % etc. Following are the operators supported by javascript:

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators
* Arithmetic Operators

### Control Structure

Control structure actually controls the flow of execution of a program. Following are the several control structure supported by javascript.

1. if … else
2. switch case
3. do while loop
4. while loop
5. for loop

## PHP

**PHP** is acronym of **Hypertext Preprocessor (PHP)** is a programming language that allows web developers to create dynamic content that interacts with databases.PHP is basically used for developing web based software applications.

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

**Key Points**

* PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
* PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
* It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
* PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
* PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

### Uses of PHP

* PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
* PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
* You add, delete, modify elements within your database through PHP.
* Access cookies variables and set cookies.
* Using PHP, you can restrict users to access some pages of your website.
* It can encrypt data.

### Characteristics

Five important characteristics make PHP's practical nature possible:

* Simplicity
* Efficiency
* Security
* Flexibility
* Familiarity

### "Hello World" Script in PHP:

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this:

<html>

<head>

<title>Hello World</title>

<body>

<?php echo "Hello, World!";?>

</body>

</html>

It will produce following result:

Hello, World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ate are recognised by the PHP Parser.

<?php PHP code goes here ?>

<?php PHP code goes here ?>

<script language="php"> PHP code goes here </script>