HTTP

HTTP:

HTTP stands for Hypertext Transfer Protocol.
Hypertext Transfer Protocol is a set of rule which is used for transferring the files like, audio, video, graphic image, text and other multimedia files on the WWW (World Wide Web).
HTTP is an application-level protocol. The communication usually takes place through TCP/IP sockets, but any reliable transport can also be used.
The standard (default) port for HTTP connection is 80, but other port can also be used.
The first version of HTTP was HTTP/0.9, which was introduced in 1991.
The latest version of HTTP is HTTP/3, which was published in September 2019. It is an alternative to its processor HTTP/2.
This latest version is already in use on the web with the help of UDP (User Datagram Protocol) instead of TCP (Transmission Control Protocol) for the underlying transport protocol.
HTTP is used to make communication between a variety of hosts and clients. It supports a mixture of network configuration.
HTTP is a protocol that is used to transfer the hypertext from the client end to the server end, but HTTP does not have any security.
Whenever a user opens their Web Browser, that means the user indirectly uses HTTP.

Three important things about HTTP
1. Connectionless: HTTP is connectionless. When the HTTP client opens the browser, the browser initiates an HTTP request.
After making the request, the client disconnect from the server and wait for the response.
When the response is ready, the server re-establish the connection again and delivers the response to the client,
after which the client disconnects the connection. So both client and server know about each other during the
current request and response only.

2. Media Independent: HTTP is media independent. HTTP can deliver any sort of data, as long as the two computers can read it.

3. Stateless: The HTTP is stateless. The client and server just know about each other just during the current request.
If the connection is closed, and two computers want to connect again, they need to provide information to each other anew,

and the connection is handled as the very first one.

-- HTTPS:
HTTPS stands for Hypertext Transfer Protocol Secure. HTTPS has a secure transfer.
It was developed by Netscape.
HTTPS is used to encrypt or decrypt user HTTP page or HTTP page requests that are returned by the webserver.
HTTPS is first used in HTTP/1.1 and is defined in RFC 2616.
In HTTPS, the standard port to transfer the information is 443.
Using the HTTPS, sensitive information that we want to transfer from one user to another user can be done securely.
HTTPS protocol uses HTTP on connection encrypted by SSL (Secure Socket Layer) or TLS (Transport Layer Security).
HTTPS protects transmitted data from man-in-the-middle (MITM) attacks and eavesdropping.
It is the default protocol for conduction financial transactions on the web.

-- HTTP Parameters:
These parameters are used in the construction of our request and response message while writing the HTTP program of the client or server
1. HTTP Version : HTTP uses a <major>.<minor> numbering scheme.
The protocol versioning policy allows the sender to indicate the format of a message and its capacity for understanding further HTTP communication.
The first line in the HTTP-Version field indicates the version of the HTTP message.
Example: HTTP / 1.1
2. Entity Tags : Entity tags are used to compare two or more entities from the same requested resource.
An Entity tag must be unique across all the entity versions associated with a particular resource.
Syntax :
entity-tag = [weak] opaque-tag
weak = "W/"
opaque-tag = quoted-string
3. Date/Time Formats
1. 1) Full Date : All the date/time stamps of HTTP MUST be represented in Greenwich Mean Time (GMT).
HTTP application has three different formats for the representation of date/time stamps:
All the date/time stamps of HTTP MUST be represented in Greenwich

Mean Time (GMT).

HTTP application has three different formats for the representation of date/time stamps:

Sun, 06 Nov 1994 08:49:37 GMT ; RFC 822, updated by RFC 1123
Sunday, 06-Nov-94 08:49:37 GMT ; RFC 850, obsoleted by RFC 1036
Sun Nov 6 08:49:37 1994 ; ANSI C's asctime() format

2) Delta Second : Some fields of HTTP header allow a time value, which is specified as an integer number of seconds,

which will be in decimal after the time that the message was received.

Eample: delta-seconds = 1*DIGIT

4. Uniform Resource Identifiers (URI)

URI is simply formatted case insensitive string, which contains the name, location, etc. to identify the website or web server.

Syntax:

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

Here,

--"http" scheme is used to locate network resources through the HTTP protocol.

--If the port is empty, the port is assumed to 80. If abs_path is empty, it is equivalent to anabs_pathof "/".

Characters in the "unsafe" and "reserved" sets are equivalent to their ""%" HEX HEX"encoding.

--Comparisons of host names MUST be case-insensitive.

5. Character Set

The Character set is used to specify the character set that the user prefers. Using comma, multiple character sets can be separated. HTTP character sets are identified using the case ?insensitive tokens.

Example : ISO-8859-1

Or -ASCII

6. Content Codings: Content coding values are used to show an encoding transformation that has been applied to an entity

Example: Accept-encoding: gzip or

Accept-encoding: compress

7. Transfer Codings : The values of transfer coding are used to indicate an encoding transformation that has been applied to an entity to ensure "safe transport" through the network.

Transfer coding is different from content coding.

Transfer coding is the property of the message, and it is not the original entity.

Example: Parameter = attribute "=" value

Attribute = token

Value = token | quoted-string

8. Product Tokens

Product Tokens allow communication applications to identify themselves by the version and name of the software.

Product token should be to the point and short.

Syntax

product = token ["/" product-version]
product-version = token

Example

User-Agent: CERN-LineMode/2.15 libwww/2.17b3
Server: Apache/0.8.4

HTTP Message:

To show how the data is exchanged betweeen the client and the server.

HTTP Client: A program that establishes a connection to a server to send one or more HTTP request messages.

HTTP Server: A program that accepts connections to serve HTTP requests by sending an HTTP response messages.

The HTTP Messages can be classified as follows:

1. Message type
2. Message Headers
3. Message Body
4. Message Length
5. General Header Fields

1. Message type

HTTP message consists of an initial request line and an initial response line.

Format: HTTP-message = Request | Response ; HTTP/1.1 messages

Initial Request Line : The initial line is different for the request and for the response.

A request-line consists of three parts: a method name, requested resource's local path, and the HTTP version being used.

All these parts are separated by spaces.

Syntax: GET /path/to/file/index.html HTTP/1.0

Initial Response Line : The initial Response line is also known as the status line.

It also has three parts: the HTTP version, a response status code that gives the result of the request,

and the English reason phrase describing the status code.

Example: HTTP/1.0 200 OK

HTTP/1.0 404 Not Found

2. Message Headers :

The Message header provides information about the request and response.
It also provides information about the object which is sent in the message body.

Message Headers are of four types:

1. General Header: It has general applicability for both request messages and response messages.
2.Request Header: It has applicability only for the request messages.
3. Response Header: It has applicability only for the response messages.
4. Entity Header: It defines meta-information about the entity-body, and about the resource identified by request.

All the above headers follow the same generic format. Each of the header fields consists of a name followed by a colon and the field values as follows:
message-header = field-name ":" [field-value]

3. Message Body :

The message body of an HTTP message is used to carry the entire body associated with the request and response.
The message-body differs from the entire-body only when a transfer-coding has been applied, as indicated by the Transfer-Encoding header field.

Syntax : message-body = entity-body
| <entity-body encoded as per Transfer-Encoding>

Transfer-Encoding MUST be used to indicate any transfer-codings which is applied by an application to ensure safe and proper transfer of the message.
Transfer-Encoding is a property of the message.

4.Message Length

The transfer-length of a message is the length of the message-body, and it appears in the message.
In a message, when a message body is allowed, and Content-Length is given, its field value MUST exactly match the number of OCTETs in the message-body.
When an invalid length is received and detected, the HTTP/1.1 user agents MUST notify the user.

5. General Header Fields

Some header fields have the applicability for both the request and response messages.
These header fields apply only when the message is transmitted.
Syntax : general-header = Cache-Control

HTTP Request

HTTP Requests are messages which are sent by the client or user to initiate an action on the server.
The first line of the message includes the request message from the client to the server,
the method which is applied to the resource,
identifier of the resource, and
the protocol version.
Request Line:
The Request-Line starts with a method token, which is followed by the Request-URI, the protocol version,
and ending with CRLF. Using the SP characters, the elements are separated.
Syntax: Request-Line = Method SP Request-URI SP HTTP-Version CRLF (carriage returns line feed)

1) Method
The method token is used to indicate the method which was performed on the resource identified by the Request-URI.
The method is case sensitive.
Syntax :
Method = "OPTIONS"
| "GET"
| "HEAD"
| "POST"
| "PUT"
| "DELETE"
| "TRACE"
| "CONNECT"
| extension-method
extension-method = token

2) Request-URI : The Request-URI is a Uniform Resource Identifier. It is used to identify the resource upon which to apply the request.
Syntax : Request-URI = "*" | absoluteURI | abs_path | authority
On the nature of the request, these four options for Request-URI depend.
a) The asterisk "*" is used to show that the request does not apply to a particular resource,
but it will apply to the server itself. It is allowed only when the method used does not necessarily apply to a resource.
Example: OPTIONS * HTTP/1.1
b) The absoluteURI form is used only when the request is being made to a proxy.

The requested proxy is used to forward the request and return the response.

Example : GET http://www.javatpoint.com/WWW/TheProject.html HTTP/1.1

c) The absolute path can't be empty. If in the original URI, none is present, it must be given as "/".

d) The authority form is only used by the CONNECT method.

The Resource Identified by a Request:
Using the examination of Request-URI and the Host header field, we can determine the exact resource identified by the Internet request.An origin server must use the following rules for determining the requested resource on an HTTP/1.1 request
An origin server must use the following rules for determining the requested resource on an HTTP/1.1 request

1. The host will be part of the Request-URI if Request-URI is an absoluteURI.
2. The host will be determined by the Host header field value if the Request-URI is not an absoluteURI, and the request includes a header field of the host.
3. The response MUST be a 400 (Bad Request) error message if the host as determined by rule 1 or 2 is not a valid host on the server.

Request Header Fields:
The request-header fields are used to allow the client to pass additional information to the server like the request and the client itself.
The request header fields act as request modifiers, with semantics equivalent to the parameters on a programming language method invocation.
Syntax:

```
request-header = Accept
                | 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

The name of the request-header field can be extended reliably only in combination with a change in the version of the protocol.

HTTP Response
HTTP Response sent by a server to the client.
The response is used to provide the client with the resource it requested.
It is also used to inform the client that the action requested has been carried out.
It can also inform the client that an error occurred in processing its request.
A HTTP response contains the following things:

1. Status Line
2. Response Header Fields or a series of HTTP headers
3. Message Body

1. Status Line: In the response message, the status line is the first line. The status line contains three items:

a) HTTP Version Number : It is used to show the HTTP specification to which the server has tried to make the message comply.

Example: HTTP-Version = HTTP/1.1

b) Status Code : It is a three-digit number that indicates the result of the request.

The first digit defines the class of the response.
The last two digits do not have any categorization role.
There are five values for the first digit, which are as follows:
1xx: Information : It shows that the request was received and continuing the process.
2xx: Success : It shows that the action was received successfully, understood, and accepted.
3xx: Redirection : It shows that further action must be taken to complete the request.
4xx: Client Error : It shows that the request contains incorrect syntax, or it cannot be fulfilled.
5xx: Server Error : It shows that the server failed to fulfil a valid request.

c) Reason Phrase : It is also known as the status text. It is a human-readable text that summarizes the meaning of the status code.

Example : HTTP/1.1 200 OK

2. Response Header Fields

The HTTP Headers for the response of the server contain the information that a client can use to find out more about the response,
and about the server that sent it.
This information is used to assist the client with displaying the response to a user, with storing the response for the use of future,
and with making further requests to the server now or in the future.

response-header = Accept-Ranges
| Age
| ETag
| Location
| Proxy-Authenticate
| Retry-After
| Server
| Vary
| WWW-Authenticate
The name of the Response-header field can be extended reliably only in combination with a change in the version of the protocol.

3. Message Body
The body of the message is used for most responses.
The exceptions are where a server is using certain status codes and where the server is responding to a client request,
which asks for the headers but not the response body.
For a response to a successful request, the body of the message contains either some information about the status of the action
which is requested by the client or the resource which is requested by the client.
For the response to an unsuccessful request, the body of the message might provide further information about
some action the client needs to take to complete the request successfully or about the reason for the error.

HTTP Entity
The Entity is transferred by the HTTP request and response message if not otherwise restricted by the request method or response status code.
An entity is used to consist of entity-header fields and entity-body, although some responses will only include the entity-headers.

-- Entity Header Fields
Entity-header fields are used to define the metainformation about the entity-body.
If body isnot present, the entity-header field defines metainformation about the

resource identified by request.
Some of this metainformation is OPTIONAL. Some might be REQUIRED by the portion of this specification.
syntax:
entity-header = Allow
| Content-Encoding
| Content-Language
| Content-Length
| Content-Location
| Content-MD5
| Content-Range
| Content-Type
| Expires
| Last-Modified
| extension-header
extension-header = message-header
The extension-header mechanism is used to allow the additional entity-header fields, which is defined without the protocol change,
but these fields cannot be assumed to be recognizable by the recipient.
Recipient ignores the unrecognized header fields and MUST be forwarded by transparent proxies.

-- Entity Body
If there is any entity-body, it will send with an HTTP request or an HTTP response in the format as described below.
The entity-header fields define the encoding.
syntax: entity-body = *OCTET
When a message-body is present, an entity-body will only present in a message.
The entity-body is obtained from the message-body using decoding any Transfer-Encoding that might have been applied
to ensure the proper and safe transfer of the message.

-- Type
When an entity-body is included with a message, the header fields Content-Type and Content-Encoding determines the data type of that body.
These are used to define a two-layer, ordered encoding model:
syntax: entity-body := Content-Encoding(Content-Type(data))

Content-Type is used to specify the media type of the underlying data.
Content-Encoding indicates any additional content coding which is applied to the data, usually for the purpose
of data compression that is a requested resource property.

Any message of HTTP/1.1 containing an entity-body SHOULD include a Content-

Type header field that defines the media type of that body.
If and only if a Content-Type field is not given the media type, the recipient MAY attempt to guess the media type
using the inspection of its content or the name extension of the URI, which is used to identify the resource.
The recipient SHOULD treat it as "application/octet-stream" type if the media type remains unknown.

-- Entity Length
The message of an entity length is the length of the message body before applying any transfer-coding.

HTTP Methods
For HTTP/1.1, the set of common methods are defined below.
The name of these methods is case sensitive, and they must be used in uppercase.
Method and Description
i) GET
This method retrieves information from the given server using a given URI. GET request can retrieve the data. It can not apply other effects on the data.
ii) HEAD
This method is the same as the GET method. It is used to transfer the status line and header section only.
iii) POST
The POST request sends the data to the server. For example, file upload, customer information, etc. using the HTML forms.
iv) PUT
The PUT method is used to replace all the current representations of the target resource with the uploaded content.
v) DELETE
The DELETE method is used to remove all the current representations of the target resource, which is given by URI.
vi) CONNECT
This method establishes a tunnel to the server, which is identified by a given URI.
vii) OPTIONS
This method describes the options of communication for the target resource.

GET Method
This method is used to retrieve data from a web server using the specifying parameters in the URL portion of the request.
This is the main method that is used for document retrieval. The use of the GET method to fetch first.htm is as follows:

```
GET /first.htm HTTP/1.1
User-Agent: Mozilla/69.0 (compatible; MSIE5.01; Windows 10)
Host: www.javatpoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

The following are the server response against the above GET request:

```
HTTP/1.1 200 OK
Date: Wed, 4 Dec 2019 5:15:40 GMT
Server: Apache/2.4. 41 (Win32)
Last-Modified: Mon, 2 Dec 2019 15:40:30 GMT
ETag: "34aa387-d-1568eb00"
Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 55
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1> First Program</h1>
</body>
</html>
```

-- HEAD Method

This method is the same as the GET method. But in the HEAD method, the server replies with a response line and headers without entity-body.

The use of HEAD method to fetch header information about first.htm is as follows:

```
HEAD /first.htm HTTP/1.1
User-Agent: Mozilla/69.0 (compatible; MSIE5.01; Windows 10)
Host: www.javatpoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

The following are the server response against the above HEAD request:

```
HTTP/1.1 200 OK
Date: Wed, 4 Dec 2019 5:15:40 GMT
Server: Apache/2.4. 41 (Win32)
Last-Modified: Mon, 2 Dec 2019 15:40:30 GMT
ETag: "34aa387-d-1568eb00"
```

Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 55
Content-Type: text/html
Connection: Closed
Here, we can see that the server does not send any data after the header.

-- POST Method
This method is used to send some data to the server, for example, update files from data, etc.
The use of POST method to send a form data to the server is as follows:

POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/69.0 (compatible; MSIE5.01; Windows 10)
Host: www.jsvatpoint.com
Content-Type: text/xml; charset=utf-5
Content-Length: 55
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

<"xml version="1.0" encoding="utf-5">
<string xmlns=" https://www.javatpoint.com/">string</string>

The script of server side process.cgi processes the passed data and sends the response which is as follows:
HTTP/1.1 200 OK
Date: Wed, 4 Dec 2019 5:15:40 GMT
Server: Apache/2.4. 41 (Win32)
Last-Modified: Mon, 2 Dec 2019 15:40:30 GMT
ETag: "34aa387-d-1568eb00"
Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 55
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1> Request Processed Successfully</h1>
</body>
</html>

-- PUT Method

This method requests the server to store the included entity-body at a location that is specified by the given URL.
The below example requests the server to save the given entity-body in first.htm at the root of the server.

```
PUT /first.htm HTTP/1.1
User-Agent: Mozilla/69.0 (compatible; MSIE5.01; Windows 10)
Host: www.javatpoint.com
Accept-Language: en-us
Connection: Keep-Alive
Content-type: text/html
Content-Length: 150

<html>
<body>
<h1> First Program</h1>
</body>
</html>
```

In first.htm file, the server will store the given entity-body, and it will also send the following response back to the client:

```
HTTP/1.1 201 Created
Date: Wed, 4 Dec 2019 5:15:40 GMT
Server: Apache/2.4. 41 (Win32)
Content-type: text/html
Content-length: 30
Connection: Closed

<html>
<body>
<h1> The file was created.</h1>
</body>
</html>
```

-- DELETE Method
This method requests the server to delete a file at a location that is specified by the given URL.
The below example requests the server to delete the first.htm file at the root of the server:

```
DELETE /first.htm HTTP/1.1
User-Agent: Mozilla/69.0 (compatible; MSIE5.01; Windows 10)
Host: www.javatpoint.com
Accept-Language: en-us
Connection: Keep-Alive
```

After the above example, the server will delete the first.htm file, and it will also send the response back to the client, which is as follows:

```
HTTP/1.1 200 OK
Date: Wed, 4 Dec 2019 5:15:40 GMT
Server: Apache/2.4. 41 (Win32)
Content-type: text/html
Content-length: 30
Connection: Closed

<html>
<body>
<h1>URL deleted</h1>
</body>
</html>
```

-- CONNECT Method

This method is used by the client. It establishes a network connection to a web server over HTTP.

The below example requests a connection with a web server which is running on the host javatpoint.com:

```
CONNECT www.javatpoint.com HTTP/1.1
User-Agent: Mozilla/69.0 (compatible; MSIE5.01; Windows 10)
```

The following example shows that the connection is established with the server, and the response is sent back to the client:

```
HTTP/1.1 200 Connection established
Date: Wed, 4 Dec 2019 5:15:40 GMT
Server: Apache/2.4. 41 (Win32)
```

-- OPTIONS Method

This method is used by the client. It is used to find out the HTTP methods and other options that are supported by a web server.

The below example requests a list of methods which is supported by a web server which is running on javatpoint.com:

```
OPTIONS * HTTP/1.1
User-Agent: Mozilla/69.0 (compatible; MSIE5.01; Windows 10)
```

In the below example, the server will send information which is based on the current configuration of the server:

```
HTTP/1.1 200 OK
Date: Wed, 4 Dec 2019 5:15:40 GMT
Server: Apache/2.4. 41 (Win32)
Allow: GET,HEAD,POST,OPTIONS,TRACE
```

Content-Type: httpd/unix-directory

---

dhilli Notes:
--3 ways to send data from client to Server
1. In request parameters (url)
2. Request body
3. Request Headers

-- Http Methods (GET,PUT
-- Cookies
-- What is HTTP Request and HTTP Response?