**HTTP FUNDAMENTALS**

Hypertext Transfer Protocol was developed by W3C (World Wide Web Consortium and IETF (Internet Engineering Task Force). It is an application layer communications protocol that is used to access resources and is the standard way of communicating through applications. It runs on top of TCP/IP, using port 80 as default or 443 for HTTPS.

HTTP 0.9 came out in 1991 and is the first documented version. On May 1996, RFC 1945 first standardized HTTP 1.0. HTTP 1.1 was officially released in January 1997 as defined in RFC 2068 and updates were released in June 1999 under RFC 2616. In June 2014, RFC 230 until RFC 7235 was released.

HTTP is based on a client-server architecture. Clients also known as “user agents” are applications that communicate with the HTTP protocol and is a program or user that sends requests to the server. They can be web browsers, web crawlers/spiders and other end user tools and applications. Servers on the other hand, are the ones providing the services and fulfills requests from client programs.

Some types of servers:

* Origin server – resources in an origin server are physically there
* Proxy server – checks authentication, forwards the requests and gives back requests.

HTTP has a secure version called the HTTPS. HTTPS is encrypted and needs a digital or a self-signed certification to set it up. It uses SSL (Secure Sockets Layer) or TLS (Transport Layer Security) to encrypt communications. These protocols takes HTTP traffic and transmits it in an encrypted form.

**Features:**

**HTTP uses a request-response protocol**: The server processes the request and replies with an HTTP response message. In a pull protocol, a client sends an HTTP request message to a server. In a push protocol, the server volunteers new information via notifications. In HTTP 2, servers can push resources to clients without the clients requesting.

**HTTP is a stateless protocol:** The server does not keep information about clients in between requests. Both the server and the client are aware of each other only during a request.

**Other functionalities:**

* Content media type (MIME: Multipurpose Internet Mail Extension) specification
* Language and character set specification
* Content negotiation – talk to tell what the recipient can handle
* Selecting the best representation for a given response when there are multiple representations available
* Client-server protocol negotiation – asking the server if it can handle higher versions. If the server can handle higher versions then it will proceed to upgrade
* Persistent connections – telling the server to not close the connection for further requests
* In HTTP 1.0, connection is terminated after a resource has been retrieved from responding to a request
* Request pipelining – sending requests one after another
* In HTTP 1.0, only one resource can be transferred per connection
* Authentication/ authorization

**General Header Fields (client & server)**

* Cache-control – used for controlling cache and telling whether a resource needs to be cached or not
* Connection – Control whether connection is persistent or not and allows the sender to specify options that are desired for that particular connection
* Date – represents the date and time at which the request was generated
* Pragma – from older version, generic directive, indicates that something is cacheable
* Used to include implementation specific directives that might apply to any recipient along the request/response chain.
* Trailer – indicates that a header is present at the end of the data to supply metadata that might be dynamically generated while the message body is sent.
* Transfer-encoding – indicates what type of transformation has been applied to the message body to safely transfer it between the sender and the recipient.
* Upgrade – used for protocol negotiation \*
* Allows the client to upgrade or change to a different protocol on the same connection.
* Via – indicate where the request passed through
* Used for tracking message forwards, avoiding request loops, and identifying the protocol capabilities of senders along the request/response chain.
* Warning – carry additional information about the status or transformation of a message error in the message \*

**Request Header Fields**

* Accept – specifying acceptable file type; server responds with negotiation, accept if other type can be delivered.
* Accept-Charset – indicates which character sets are acceptable for the response
* Accept-Encoding –restricts the content-codings that are acceptable in the response.
* Accept-Language – restricts the set of natural languages that are preferred as a response to the request.
* Authorization
* Ensure protection from unauthorized access
* When proper credentials are entered, another request will be sent with authorization
* Without authorization, www-authenticate challenge will be sent and a pop-up authorization will appear
* Proxy Authorization – allows the client to identify itself (or its user) to a proxy which requires authentication
* Consists of credentials containing the authentication information of the user agent for the proxy and/or realm of the resource being requested.
* Expect – for two-face connection
* Indicates that particular server behaviors are required by the client.
* If any of the expectations cannot be met the server must respond with a 417 (Expectation Failed).
* From – contains contact credentials of sender – an internet e-mail address
* May be used for logging purposes and as a means for identifying the source of invalid or unwanted requests.
* Host – required in HTTP 1.1
* Specifies the internet host and port number of the resource being requested. This enables the origin server to distinguish among resources while servicing requests for multiple host names on a single IP address
* If-Match – comparing the file in the cache and server to check if file is unmodified to avoid state entity.
* If-None-Match – requests the server to perform the requested method only if one of the given value in a tag matches the given entity tags represented by the Etag
* Used to update caches or to prevent to upload a new resource when one is already existing
* If-Range – “is what I have a portion of what you have”
* Can be used with a conditional GET to request only the portion of the entity that is missing
* Allows the client to “short circuit” the second request
* If-Unmodified-Since – used with a method to make it conditional
* Expects the entity to be transmitted only if it has not been modified after the given date
* Ranges – Specifying portions of the resource the client wants to receive
* Max-Forwards – for tracing and limiting a trace.
* Limit the number of proxies or gateways that can forward the request to the next inbound server
* Contains a decimal integer indicating the remaining number of times a request message may be forwarded
* Referer – specify the address (URI) where the request was originated; to check where traffic is coming from
* TE – trailer encoding
* Specifies the transfer encodings the user agent is willing to accepts
* User-Agent – contains additional information about the clients’ identity

References:

<http://searchsoftwarequality.techtarget.com/definition/HTTPS>

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec12.html>

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

https://tools.ietf.org/html/rfc7230#section-6.7