HTTP Protocol

HTTP stands for Hyper Text Transfer Protocol.WWW is about communication between web clients and servers.Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP ResponsesWorld Wide Web Communication.The World Wide Web is about communication between web clients and web servers. Clients are often browsers (Chrome, Edge, Safari), but they can be any type of program or device. Servers are most often computers in the cloud.

HTTP Request / Response :- Communication between clients and servers is done by requests and responses :-

1. A client (a browser) sends an HTTP request to the web
2. An web server receives the request
3. The server runs an application to process the request
4. The server returns an HTTP response (output) to the browser
5. The client (the browser) receives the response.

A typical HTTP request / response circle :-

1. The browser requests an HTML page. The server returns an HTML file.
2. The browser requests a style sheet. The server returns a CSS file.
3. The browser requests an JPG image. The server returns a JPG file.
4. The browser requests JavaScript code. The server returns a JS file
5. The browser requests data. The server returns data (in XML or JSON).

HTTP means HyperText Transfer Protocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement Web sites that react intelligently to user input. This shortcoming of HTTP is being addressed in a number of new technologies, including ActiveX, Java, JavaScript and cookies.

HTTP Status Codes are Error Messages :- HTTP response status codes. Status codes are issued by a server in response to a client's request made to the server.All HTTP response status codes are separated into five classes or categories. The first digit of the status code defines the class of response, while the last two digits do not have any classifying or categorization role. There are five classes defined by the standard : -

- **1xx informational response :** The request was received, continuing process.

1) 100 Continue :- The server has received the request headers and the client should proceed to send the request body.Sending a large request body to a server after a request has been rejected for inappropriate headers would be inefficient. To have a server check the request's headers, a client must send Expect: 100- continue as a header in its initial request and receive a 100 Continue status code in response before sending the body.

2) 101 Switching Protocols :- The requester has asked the server to switch protocols and the server has agreed to do so.

3) 102 Processing :- A WebDAV request may contain many sub-requests involving file operations, requiring a long time to complete the request. This code indicates that the server has received and is processing the request, but no response is available yet.

4) 103 Early Hints :- Used to return some response headers before final HTTP message.

- **2xx successful** – The request was successfully received, understood, and accepted

1) 200 OK :- Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request, the response will contain anentity describing or containing the result of the action.

2) 201 Created :- The request has been fulfilled, resulting in the creation of a new resource.

3) 202 Accepted :- The request has been accepted for processing, but the processing has not been completed.

4) 203 Non-Authoritative Information :- The server is a transforming proxy that received a 200 OK from its origin, but is returning a modified version of the origin's response.

5) 204 No Content :- The server successfully processed the request, and is not returning any content.

6) 205 Reset Content :- The server successfully processed the request, asks that the requester reset its document view, and is not returning any content.

7) 206 Partial Content :- The server is delivering only part of the resource due to a range header sent by the client. The range header is used by HTTP clients to enable resuming of interrupted downloads, or split a download into multiple simultaneous streams.

8) 207 Multi-Status :- The message body that follows is by default an XML message and can contain a number of separate response codes, depending on how many sub-requests were made.

9) 208 Already Reported :- The members of a DAV binding have already been enumerated in a preceding part of the (multistatus) response, and are not being included again.

10) 226 IM Used :- The server has fulfilled a request for the resource, and the response is a representation of the result of one or more instance-manipulations applied to the current instance.

- **3xx redirection :** Further action needs to be taken in order to complete the request .

1) 300 Multiple Choices :- Indicates multiple options for the resource from which the client may choose.

2) 301 Moved Permanently :- This and all future requests should be directed to the given URI.

3) 302 Found :- Tells the client to look at (browse to) another URL. 302 has been superseded by 303 and 307.

4) 303 See Other :- The response to the request can be found under another URI using the GET method. When received in response to a POST , the client should presume that the server has received the data and should issue a new GET request to the given URI.

5) 304 Not Modified :- Indicates that the resource has not been modified since the version specified by the request headers If-Modified-Since or If-None-Match.

6) 305 Use Proxy :- The requested resource is available only through a proxy, the address for which is provided in the response.

7) 306 Switch Proxy :- No longer used that means subsequent requests should use the specified proxy.

8) 307 Temporary Redirect :- In this case, the request should be repeated with another URI.

9) 308 Permanent Redirect :- The request and all future requests should be repeated using another URI.

- **4xx client error -** the request contains bad syntax or cannot be fulfilled

1) 400 Bad Request :- The server cannot or will not process the request due to an apparent client error.

2) 401 Unauthorized :- Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.

3) 402 Payment Required :- Reserved for future use. The original intention was that this code might be used as part of some form of digital cash or micropayment scheme, as proposed.

4) 403 Forbidden :- The request contained valid data and was understood by the server, but the server is refusing action. This may be due to the user not having the necessary permissions for a resource or needing an account of some sort, or attempting a prohibited action.

5) 404 Not Found :- The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.

6) 405 Method Not Allowed :- A request method is not supported for the requested resource; for example, a GET request on a form that requires data to be presented via POST, or a PUT request on a read-only resource.

7) 406 Not Acceptable :- The requested resource is capable of generating only content not acceptable according to the Accept headers sent in the request.

8) 407 Proxy Authentication Required :- The client must first authenticate itself with the proxy.

9) 408 Request Timeout :- The server timed out waiting for the request. According to HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait. The client may repeat the request without modifications at any later time."

10) 409 Conflict :- Indicates that the request could not be processed because of conflict in the current state of the resource.

11) 410 Gone :- Indicates that the resource requested is no longer available and will not be available again. This should be used when a resource has been intentionally removed and the resource should be purged.

12) 411 Length Required :- The request did not specify the length of its content, which is required by the requested resource.

13) 412 Precondition Failed :- The server does not meet one of the preconditions that the requester put on the request header fields.

14) 413 Payload Too Large :- The request is larger than the server is willing or able to process. Previously called "Request Entity Too Large".

15) 414 URI Too Long :- The URI provided was too long for the server to process. Often the result of too much data being encoded as a query-string of a GET request, in which case it should be converted to a POST request.

16) 415 Unsupported Media Type :- The request entity has a media type which the server or resource does not support. For example, the client uploads an image as image/svg+xml, but the server requires that images use a different format.

17) 416 Range Not Satisfiable :- The client has asked for a portion of the file , but the server cannot supply that portion.

18) 417 Expectation Failed :- The server cannot meet the requirements of the Expect request-header field.

19) 421 Misdirected Request :- The request was directed at a server that is not able to produce a response.

20) 422 Unprocessable Entity :- The request was well-formed but was unable to be followed due to semantic errors.

21) 423 Locked :- The resource that is being accessed is locked.

22) 424 Failed Dependency :- The request failed because it depended on another request and that request failed.

23) 425 Too Early :- Indicates that the server is unwilling to risk processing a request that might be replayed.

24) 426 Upgrade Required :- The client should switch to a different protocol.

25) 428 Precondition Required :- The origin server requires the request to be conditional.

26) 429 Too Many Requests :- The user has sent too many requests in a given amount of time. Intended for use with rate-limiting schemes.

27) 431 Request Header Fields Too Large :- The server is unwilling to process the request because either an individual header field or all the header fields collectively are too large.

28) 451 Unavailable For Legal Reasons :- A server operator has received a legal demand to deny access to a resource or to a set of resources that includes the requested resource.

- **5xx server error :** The server failed to fulfil an apparently valid request.

1) 500 Internal Server Error :- A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.

2) 501 Not Implemented :- The server either does not recognize the request method, or it lacks the ability to fulfil the request. Usually this implies future availability.

3) 502 Bad Gateway :- The server was acting as a gateway or proxy and received an invalid response from the upstream server.

4) 503 Service Unavailable :- The server cannot handle the request. Generally, this is a temporary state.

5) 504 Gateway Timeout :- The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.

6) 505 HTTP Version Not Supported :- The server does not support the HTTP protocol version used in the request.

7) 506 Variant Also Negotiates :- Transparent content negotiation for the request results in a circular reference.

8) 507 Insufficient Storage :- The server is unable to store the representation needed to complete the request.

9) 508 Loop Detected :- The server detected an infinite loop while processing the request.

10) 510 Not Extended :- Further extensions to the request are required for the server to fulfil it.

11) 511 Network Authentication Required :- The client needs to authenticate to gain network access.

Method in server :-

Some of the method of server are :-

- **POST :-** A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
- **GET Method :-** 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.
- **Head :-** Same as GET, but transfers the status line and header section only.
- **PUT :-** Replaces all current representations of the target resource with the uploaded content.

- **DELETE :-** Removes all current representations of the target resource given by a URI.
- **CONNECT :-** Establishes a tunnel to the server identified by a given URI.
- **OPTIONS :-** Describes the communication options for the target resource.
- **TRACE :-** Performs a message loop-back test along the path to the target resource.