**Task 01:-** Blog write up

**1.difference between HTTP1. 1 Vs HTTP2**

**HTTP1. 1-**

* Keeps all requests and responses in plain text format.
* Pipelining and Head-of-Line Blocking
* Programs like Gzip have long been used to compress the data sent in HTTP messages, especially to decrease the size of CSS and JavaScript files.
* The header component of a message, however, is always sent as plain text. Although each header is quite small, the burden of this uncompressed data weighs heavier and heavier on the connection as more requests are made, particularly penalizing complicated, API-heavy web applications that require many different resources and thus many different resource requests.

**HTTP2-**

* HTTP2 uses the binary framing layer to encapsulate all messages in binary format.
* One of the themes that has come up again and again in HTTP/2 is its ability to use the binary framing layer to exhibit greater control over finer detail.
* The same is true when it comes to header compression. HTTP/2 can split headers from their data, resulting in a header frame and a data frame.
* The HTTP/2-specific compression program [HPACK](https://tools.ietf.org/html/draft-ietf-httpbis-header-compression-12) can then compress this header frame. This algorithm can encode the header metadata using Huffman coding, thereby greatly decreasing its size.

**2.http version history-**

## 1)HTTP/0.9 — The One-line Protocol (Year of release 1991)

* Initial version of HTTP — a simple client-server, request-response, telenet-friendly protocol
* Request nature: single-line (method + path for requested document)
* Methods supported: GET only
* Response type: hypertext only
* Connection nature: terminated immediately after the response
* No HTTP headers (cannot transfer other content type files), No status/error codes, No URLs, No versioning

## 2)HTTP/1.0 — Building extensibility( Year of release is 1996)

* Browser-friendly protocol
* Provided header fields including rich metadata about both request and response (HTTP version number, status code, content type)
* Response: not limited to hypertext (Content type header provided ability to transmit files other than plain HTML files — e.g. scripts, style sheets, media)
* Methods supported: **GET HEAD POST**
* Connection nature: terminated immediately after the response

# 3)HTTP/1.1 — The Standardize Protocol(Year of release is 1997)

* HTTP/1.1 was published in early 1997, only a few months after HTTP/1.0.
* A connection can be reused, saving the time to reopen it numerous times to display the resources embedded into the single original document retrieved.
* Pipelining has been added, allowing to send a second request before the answer for the first one is fully transmitted, lowering the latency of the communication.
* Additional cache control mechanisms have been introduced.
* Content negotiation, including language, encoding, or type, has been introduced, and allows a client and a server to agree on the most adequate content to exchange.

4)HTTP/2- A protocol for Grater performance

* In the first half of the 2010s, Google demonstrated an alternative way of exchanging data between client and server, by implementing an experimental protocol SPDY. This amassed interest from developers working on both browsers and servers. Defining an increase in responsiveness, and solving the problem of duplication of data transmitted, SPDY served as the foundations of the HTTP/2 protocol.
* t is a multiplexed protocol. Parallel requests can be handled over the same connection, removing the order and blocking constraints of the HTTP/1.x protocol.
* It compresses headers. As these are often similar among a set of requests, this removes duplication and overhead of data transmitted.
* It allows a server to populate data in a client cache, in advance of it being required, through a mechanism called the server push.

**3.List 5 difference between Browser Js console Vs Node Js-**

* In browser “window” is a predefined global object which has functions and attributes, where as NodeJS doesn’t have it.
* In browser “location” is another predefined object, where as NodeJS doesn’t have it.
* In browser “require” is not predefined object, where as NodeJS has it.
* In browser module is not required, where as in NodeJS you have to keep your code inside the module.
* In browser “document” is a predefined object, where as NodeJS doesn’t have it.

**4.what happens when you type a URL in the address bar in the browser.**

* You enter a URL into a web browser
* The browser looks up the IP address for the domain name via DNS
* The browser sends a HTTP *request* to the server
* The server sends back a HTTP *response*
* The browser begins rendering the HTML
* The browser sends requests for additional objects embedded in HTML (images, CSS, JavaScript) and repeats steps 3-5.
* Once the page is loaded, the browser sends further Async requests as needed.