TASK 1;

1. Difference between HTTP1.1 vs HTTP2

2. http version history

3. List 5 difference between Browser JS(console) vs Nodejs

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

1.

\*Binary Framing Layer-

All requests and responses in the HTTP 1.1 version are kept in plain text format, while HTTP 2 uses binary framing layer to encapsulate all messages in binary format.

The conversion of messages into binary allows HTTP 2 try new approaches of data delivery not available in HTTP 1.1

\*Delivery modes-

HTTP 1.1 uses persistent TCP connections to retrieve the additional resources requested by the user, but if one request fails, all the subsequent requests are blocked. There is also a limit on the number of parallel TCP connections that can be established.

In HTTP 2, using Binary Framing, requests are sent in a single connection, as streams of data, containing frames. These frames can be re assembled at the other end. Hence, there is no problem of waiting for one request to be completed. Also, multiple requests can be sent at once. This is called Multiplexing

\*Stream prioritization-

The binary framing layer organizes messages into parallel streams of data. When a client sends concurrent requests to a server, it can prioritize the responses it is requesting by assigning a weight between 1 and 256 to each stream. The higher number indicates higher priority.

\*Compression-

Programs like gzip are used to compress data (CSS and Js files) in HTTP but the header of the message is sent in plain text format, which can be a burden on the connection.

HTTP 2can split the headers from their data, resulting in header frame and data frame. 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.

Invented by Tim Berners-Lee, HTTP (Hypertext Transfer Protocol) is the underlying communication protocol of World Wide Web. **HTTP functions as a request–response protocol in the client–server computing model.** HTTP standards are developed by the [Internet Engineering Task Force](https://en.wikipedia.org/wiki/Internet_Engineering_Task_Force) (IETF) and the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C).

HTTP has four versions — HTTP/0.9, HTTP/1.0, HTTP/1.1, and HTTP/2.0. Today the version in common use is HTTP/1.1 and the future will be HTTP/2.0.

HTTP 0.9: The One-Line Protocol

The request consists of a single line: GET method and the path of the requested document. The response is a single hypertext document—no headers or any other metadata, just the HTML. Important features-

* Client request is a single ASCII character string.
* Server response is an ASCII character stream.
* Server response is a hypertext markup language (HTML).
* Connection is terminated after the document transfer is complete.

# HTTP/1.0: Rapid Growth

# As the demands of the world wide web increased, the requirements of the protocols increased from retrieving basic html files to much more functionality. Hence, a new version of http had to be formulated to handle these limitations. Important features-

* Request may consist of multiple newline separated header fields.
* Response object is prefixed with a response status line.
* Response object has its own set of newline separated header fields.
* Response object is not limited to hypertext.
* The connection between server and client is closed after every request.

Both the request and response headers were kept as ASCII encoded, but the response object itself could be of any type: an HTML file, a plain text file, an image, or any other content type.

# HTTP/1.1: Internet Standard

The HTTP/1.1 standard resolved a lot of the protocol ambiguities found in earlier versions and introduced a number of critical performance optimizations: keepalive connections, chunked encoding transfers, byte-range requests, additional caching mechanisms, transfer encodings, and request pipelining. Important features-

* Request for HTML file, with encoding, charset, and cookie metadata
* Number of octets in the chunk expressed as an ASCII hexadecimal number (256 bytes)
* End of chunked stream response
* Request for an icon file made on same TCP connection
* Inform server that the connection will not be reused
* Icon response, followed by connection close
* Methods supported: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS

 The main difference is that we have two object requests, one for an HTML page and one for an image, both delivered over a single connection. This is connection keepalive in action, which allows us to reuse the existing TCP connection for multiple requests to the same host and deliver a much faster end-user experience.

To terminate the persistent connection, the second client request sends an explicit close token to the server via the Connection header. Similarly, the server can notify the client of the intent to close the current TCP connection once the response is transferred. Technically, either side can terminate the TCP connection without such signal at any point, but clients and servers should provide it whenever possible to enable better connection reuse.

# HTTP/2: Improving Transport Performance

The primary focus of HTTP/2 is on improving transport performance and enabling both lower latency and higher throughput. None of the high-level protocol semantics are affected: all HTTP headers, values, and use cases are the same.

Any existing website or application can and will be delivered over HTTP/2 without modification: there is no need to modify our application markup to take advantage of HTTP/2. The HTTP servers will have to speak HTTP/2, but that should be a transparent upgrade for the majority of users.

3. Differences between browser console and NodeJS –

* In the browser, most of the time we interact with the [DOM](https://flaviocopes.com/dom/), or other [Web Platform APIs](https://flaviocopes.com/web-platform/) like Cookies. Those do not exist in Node. There is no document, window and all the other objects that are provided by the browser.
* Another big difference is that in Node.js one can control the environment. Unless we are building an open source application that anyone can deploy anywhere, we know which version of Node to run the application on. Compared to the browser environment, where there is no luxury to choose what browser the visitors will use.
* The browser, doesn’t provide all the APIs that Node.js provides through its modules, like the filesystem access functionality.
* The object “require” is predefined in Node which is used to include modules in the app. In Node everything is a module. Code must be kept inside a module. But moduling is not mandatory in client side JavaScript, i.e. in browsers.
* The object “location” is related to a particular url; that means it is page specific. So, node doesn’t require that. Itis another predefined object in browsers, that has all the information about the url we have loaded.

4. When we type a URL into a browser the first thing that happens is a Domain Name Server (DNS) matches the given URL to an IP address. Then the browser sends an HTTP request to the server and the server sends back an HTTP response. The browser begins rendering the HTML on the page while also requesting any additional resources such as CSS, JavaScript, images, etc.

The HTML is rendered with the help of DOM parser where a tree is constructed of the HTML elements. The same is done by the CSS parser. Together they form the layout tree. The JavaScript engine compiles the JS code.

Each subsequent request completes a request/response cycle and is rendered in turn by the browser. Then once the page is loaded some sites will make further asynchronous requests. Most websites use the TCP for communicating (request-response).