# Task 1

**1.Difference between http1.1 vs http2**

1.Key Features

HTTP1.1: It supports connection reuse i.e., for every TCP connection there could be multiple requests and responses, and pipelining where the client can request several resources from the server at once.:

HTTP 2: Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. It is done using streams which can be prioritized, can have dependencies and individual flow control. It also provides a feature called server push that allows the server to send data that the client will need but has not yet requested.

2.Web Traffic:

HTTP 1.1: HTTP/1.1 provides faster delivery of web pages and reduces web traffic as compared to HTTP/1.0. However, TCP starts slowly and with domain sharding (resources can be downloaded simultaneously by using multiple domains), connection reuse and pipelining, there is an increased risk of network congestion.

HTTP 2: HTTP/2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays.

3.Protocol type:

HTTP 1.1: Text based protocol that is in the readable form.

HTTP 2: It is a binary protocol (HTTP requests are sent in the form of 0s and 1s). Needs to be converted back from binary in order to read it.

4.Predicting Resource Requests:

HTTP 1.1: In HTTP/1.1, if the developer knows in advance which additional resources the client machine will need to render the page, they can use a technique called resource in-lining to include the required resource directly within the HTML document that the server sends in response to the initial GET request. For example, if a client needs a specific CSS file to render a page, in-lining that CSS file will provide the client with the needed resource before it asks for it, reducing the total number of requests that the client must send.

HTTP 2: Since HTTP/2 enables multiple concurrent responses to a client’s initial GET request, a server can send a resource to a client along with the requested HTML page, providing the resource before the client asks for it. This process is called server push. In this way, an HTTP/2 connection can accomplish the same goal of resource in-lining while maintaining the separation between the pushed resource and the document. This means that the client can decide to cache or decline the pushed resource separate from the main HTML document, fixing the major drawback of resource in-lining.

5.Compression:

HTTP 1.1: 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. Additionally, the use of cookies can sometimes make headers much larger, increasing the need for some kind of compression.

HTTP 2: 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 can then compress this header frame. This algorithm can encode the header metadata using Huffman coding, thereby greatly decreasing its size. Additionally, HPACK can keep track of previously conveyed metadata fields and further compress them according to a dynamically altered index shared between the client and the server.

**Q-2. http version history**

The different versions of http are:

1.http 0.9 – 1991

2.http 1.0 – 1996

3.http 1.1 – 1997

4.http 2.0 – 2015

5.http 3.0 (Draft version) – 2020.

**Q-3 Difference between NodeJS and BrowserJS**

1. JavaScript is a programming language that is used for writing scripts on the website. NodeJS is a JavaScript runtime environment.

2. JavaScript can only be run in the browsers.

NodeJS code can be run outside the browser.

3. JavaScript is basically used on the client-side. It is mostly used on the server-side.

4. JavaScript is capable enough to add HTML and play with the DOM.

Nodejs does not have capability to add HTML tags.

5. JavaScript can run in any browser engine as like JS core in safari and Spidermonkey in Firefox.

Nodejs can only run in V8 engine of google chrome.

6. JavaScript is used in frontend development.

Nodejs is used in server-side development.

7. Some of the JavaScript frameworks are RamdaJS, TypedJS, etc.

Some of the Nodejs modules are Lodash, express etc. These modules are to be imported from npm.

8. JavaScript is the upgraded version of ECMA script that uses Chrome’s V8 engine written in C++. Nodejs is written in C, C++ and JavaScript.

**Q-4 What happens when you type a URL in the address bar in the browser?**

When a URL is entered into the browser, the browser looks up the IP address for the domain name via DNS, then the browser sends a HTTP request to the server. The server sends back a HTTP response. After this, the browser begins rendering the HTML and sends requests for additional objects embedded in HTML (images, CSS, JavaScript) and repeats the above steps. Once the page is loaded, the browser sends further async requests as needed.