**Task 01**

**Question 1:**

Difference between **HTTP/1.1 vs HTTP/2**

* **HTTP** - **Hypertext Transfer Protocol**
* HTTP is the entire backbone of the world wide web.
* HTTP is a TCP/IP based communication protocol.
* Define how files are to be transferred between clients and servers on the world wide web.

|  |  |
| --- | --- |
| **HTTP/1.1** | **HTTP/2** |
| 1. HTTP/1.1 loads the website slowly. | 1. HTTP/2 loads the website fast. |
| 1. HTTP/1.1, the first standardizes version of HTTP, was introduced in 1997. | 1. Google introduced HTTP/2 in 2015 which is based on the SPDY protocol. |
| 1. HTTP/1.1 it requires performance optimization to load the website effectively. | 1. HTTP/2 does not require any additional optimization. |
| 1. It present significant performance optimizations (over HTTP/0.9 and HTTP/1.0) | 1. It supported multiplexing (multiple request/response sent and received asynchronously over a single TCP connection) |
| 1. HTTP/1.1 loads resources one after the other, so if one resource cannot be load, it blocks all the other resources behind it. | 1. HTTP/2 is able to use a single TCP connection to send multiple streams of data at once, so that no one resource blocks any other resource HTTP/1. |
| 1. HTTP/1.1 introduced chunked transfer encoding to allow content on persistent connection to be streamed rather than buffered. | 1. HTTP/2 no longer supports HTTP/1.1’s chunked transfer encoding mechanism, as it provide its own, more efficient, mechanism for data streaming. |

* HTTP/1 is less secured when compared with HTTP/2 and HTTP/2 is more secure when compared with HTTP/1.1
* HTTP/2 is binary, whereas HTTP1 is textual.
* HTTP/2 is fully multiplexed, instead of ordered and blocking.
* HTTP/2 can, therefore, use one connection for parallelism.
* HTTP/2 uses header compression to reduce overhead.
* HTTP/2 allows servers to “push” response proactively into client caches.
* HTTP/2 is secured by default.

**Question 2:**

**HTTP version History**

Invented by **Tim Berners Lee** at CERN in the years 1989-1991.

HTTP functions as a request-response protocol in the client-server computing model.

HTTP has 4 versions

|  |  |
| --- | --- |
| **HTTP Version** | **Years** |
| HTTP/0.9 | 1991 |
| HTTP/1.0 | 1996 |
| HTTP/1.1 | 1997 |
| HTTP/2.0 | 2015 |
| 3.0 | Draft (2020) |

**HTTP/0.9 - The one-line protocol**

* Initial version of HTTP- a simple client-server, request-response, telenet-friendly protocol.
* It’s extremely simple request consists of a single line and start with the only GET followed by path to the resource.
* There were no status or error massage.

**HTTP/1.0 - Building extensibility**

* Browser – friendly protocol.
* Version information is now sent within each request.
* A status code is also sent at the beginning of the response.
* The HTTP header has been introduced.
* The transfer of the other documents than plain HTML filles has been added.
* Methods supported: GET, HEAD, POST
* Connection nature: terminated immediately after the response.

**HTTP/1.1 – The standardized protocol**

* A connection can be reused.
* Pipelining has been added, allowing to send a second request before the answer the first one is fully transmitted.
* Cache control mechanism have been introduced.
* Introduced critical performance optimizations and feature enhancements-persistent and pipelined connections, chunked transfer compression/decompression, faster response and great bandwidth savings by adding cache support.
* Methods supported: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS
* Connection nature: long-lived

**HTTP/2 – Improving transport Performance**

* HTTP/2 is binary, instead of textual.
* It is a multiplexed protocol.
* Parallel requests can be handled over the same connection, removing the ordered and blocking issues if the HTTP/1.1 protocol.
* It compresses headers. As these are often similar among a set of requests, this removes duplication and overhead of data transmitted.
* HTTP/2 allows servers to “push” response proactively into client caches

**Question 3:**

**List 5 difference between Browser JS (console) vs NodeJS**

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

**QUESTION 4:**

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

* **URL** stands for **Uniform Resource Locator.**
* URL is the address of the website which you can find in the address bar of your web browser.
* It is a reference to a resource on the internet, be it images, hypertext pages, audio/video files, etc.

**Example:**

https://www.example.com/

**What is DNS:**

* **DNS** is short for **Domain Name System**.
* Like a phonebook, DNS maintains and maps the name of the website, i.e. URL, and particular IP address it links to.
* Every URL on the internet has a unique IP address which is of the computer which hosts the server of the website requested.

**Steps for what happens when we enter a URL:**

1. Browser checks cache for DNS entry to find the corresponding IP address of website.  
   It looks for following cache. If not found in one, then continues checking to the next until found.



**Example Picture 1.0**

* + **Cache**
  + **Check Browser**
  + **Operating Systems Cache**
  + **Router Cache**
  + **ISP Cache**

1. If not found in cache, ISP’s (Internet Service Provider) DNS server initiates a DNS query to find IP address of server that hosts the domain name.  
   So, the domain name which you entered got converted into a DNS number. Suppose the above entered domain name **example.com** has an IP address of 93.184.216.34.
2. Browser initiates a TCP (Transfer Control Protocol) connection with the server by using

**STEP 1**: Synchronize (SYN)

**STEP 2**: Synchronize (SYN) **+** Acknowledge (ACK)

**STEP 3**: Acknowledge (ACK)

1. Browser sends an HTTP request to the web server. GET or POST request.
2. Server on the host computer handles that request and sends back a response. It assembles a response in some format like JSON, XML and HTML.
3. Server sends out an HTTP response along with the status of response.
4. Browser displays HTML content.
5. Finally, **Done**.