DAY-1

**QUESTION-1**

HTTP/1.1 – APPLICATION LAYER STANDARDIZED PROTOCOL

This came after the release of HTTP/1.0. HTTP/1.1 introduced huge improvements.

Persistent Connections:

TCP connection is reused. It no longer needed to be opened multiple times to display resources (images, texts, audio) in single document. Latency on subsequent requests is reduced since there is no time spent in TCP's connection opening handshake. It uses keep-alive header.

Pipelining:

Pipelining was added. This allowed a second request to be sent before the answer to the first one was fully transmitted. Parallel requests are possible but responses returned are in the order of requests, so there will be waiting time for the responses which are ready to render. This is Head-Of-Line blocking.

Streaming with Chunked Transfer:

Chunked responses were also supported. They are sent independently through a single persistent connection. The first chunk specifies the type in Transfer-Encoding header. Chunked encoding isn't exclusive to video streaming or file downloads. It is used to transmit any large amount of data to a client.

Low Latency:

Reduced latency due to the head-of-line blocking and lack of multiplexing.

HTTP/2

Binary Protocol:

It's a binary protocol rather than a text protocol. It uses binary format for data transmission. It can’t be read and created manually. Despite this hurdle, it allows for the implementation of improved optimization techniques.

Multiplexed Protocol:

Multiplexing using stream ID over single TCP connection. Each HTTP request is tagged with stream ID. It removes the constraints of the HTTP/1.1 protocol. It no longer need to wait for responses of each request as there is no Head-Of-Line blocking. It uses single TCP connection in an efficient way.

Compression:

This compresses headers as we have stream ID as identifier of request. As these are often similar among a set of requests, this removes the duplication and overhead of data transmitted.

Server Push:

This allows a server to populate data in a client cache. Both client and server should allow HTTP/2 to make server push work else it can be abused. It lets the server pre-emptively PUSH website assets to the client without the user having explicitly asked for them.

Secure:

This is a secure design by default. It supports Transport Layer Security. This allows secure data transmission. It no longer connects on port 80. It will now connect on port 443.

Protocol Negotiation during TLS:

TLS extensions can be used to inject some own functions while TLS is executing. It upgrades both client and server HTTP versions while doing their encryption.

**QUESTION-2**

Object and its internal representation in JavaScript

Ways to implement objects

Key-Value Pairs:

JavaScript objects are collections of key-value pairs. The keys are typically strings or symbols, and the values can be any data type, including other objects, functions, arrays, and primitive values.

Properties and Methods:

Properties are the values associated with an object, and methods are functions that are properties of an object. You can access properties and methods of an object using dot notation.

Prototype-based Inheritance:

JavaScript objects can inherit properties and methods from other objects through prototype-based inheritance. Each object has a prototype, which is another object. When you try to access a property or method on an object and it's not found, JavaScript looks for it in the object's prototype chain.

Object Creation:

There are several ways to create objects in JavaScript:

Object literals: const obj = { key: value };

Constructor functions: function MyClass() { this.key = value; }

ES6 Classes: class MyClass { constructor() { this.key = value; } }

Object.create(): const obj = Object.create(proto);

Object Representation in Memory:

Objects in JavaScript are typically implemented using hash tables or similar data structures internally. When you create an object, memory is allocated to store its properties and methods. Each property is stored along with its associated key in the hash table.