Task – 1

**1 . Write a blog on Difference between HTTP1.1 vs HTTP2?**

**Introduction:**

The Hypertext Transfer Protocol (HTTP) is the foundation of data communication on the World Wide Web. As technology evolves, so does HTTP, and two major versions that have significantly impacted web performance are HTTP/1.1 and HTTP/2. In this blog post, we will delve into the key differences between these two versions and understand how HTTP/2 has addressed some of the limitations of its predecessor.

**HTTP/1.1 Overview:**

HTTP/1.1, introduced in 1997, has been the workhorse of the World Wide Web for many years. While it has served the online community well, it has certain limitations that become more pronounced in the modern web landscape.

1. Connection Handling:

HTTP/1.1 uses a request-response model where each request establishes a new connection to the server. This results in latency as each new connection has to undergo the TCP handshake process.

2. Header Overhead:

The protocol includes redundant header information for each request and response, leading to increased data transfer size and longer loading times.

3. Concurrency:

Limited concurrency due to the head-of-line blocking problem. In HTTP/1.1, only one request can be outstanding on a connection at a time, leading to potential inefficiencies.

**HTTP/2 Advancements:**

To overcome the limitations of HTTP/1.1, HTTP/2 was introduced in 2015 with several improvements.

1. Multiplexing:

One of the most significant advancements is multiplexing, which allows multiple requests and responses to be multiplexed over a single connection. This reduces latency and makes better use of available network resources.

2. Header Compression:

HTTP/2 uses header compression to minimize redundant header information, significantly reducing the overhead associated with headers and improving data transfer efficiency.

3. Binary Protocol:

HTTP/2 is a binary protocol as opposed to the text-based protocol of HTTP/1.1. This makes it more efficient to parse, reducing complexity and improving performance.

4. Server Push:

HTTP/2 introduces server push, a feature that allows the server to push resources to the client before they are explicitly requested. This can lead to faster page loads by pre-emptively sending assets that the server anticipates the client will need.

**Conclusion:**

In summary, while HTTP/1.1 has been the backbone of the web for many years, HTTP/2 brings substantial improvements in terms of performance and efficiency. The introduction of multiplexing, header compression, binary protocol, and server push in HTTP/2 collectively contribute to a faster and more responsive web experience. As web technologies continue to evolve, it is crucial for developers and web administrators to stay informed about the latest protocols and best practices to ensure optimal performance for their users.

2. **Read about IP address, port, HTTP methods, MAC address?**

**IP Address (Internet Protocol Address):**

- An IP address is a numerical label assigned to each device participating in a computer network that uses the Internet Protocol for communication. It serves two main purposes: identifying the host or network interface and providing the location of the host in the network.

- There are two types of IP addresses: IPv4 (32-bit address) and IPv6 (128-bit address). IPv4 addresses are more commonly used, but IPv6 is becoming increasingly important as IPv4 addresses are running out.

**Port:**

- In computer networking, a port is a communication endpoint. It is identified by a 16-bit unsigned number, providing a specific address for data to be sent to within a network. Ports allow multiple services or applications to run on a single device, each having a unique port number.

- Ports are categorized into three ranges: well-known ports (0-1023), registered ports (1024-49151), and dynamic or private ports (49152-65535). Well-known ports are assigned to common services like HTTP (port 80) and HTTPS (port 443).

**HTTP Methods (HTTP Verbs):**

- HTTP methods, also known as HTTP verbs, are actions that can be performed on a resource identified by a URI (Uniform Resource Identifier). The common HTTP methods include:

GET: Retrieve data from a specified resource.

POST: Submit data to be processed to a specified resource.

PUT: Update a resource or create a new resource if it does not exist.

DELETE: Delete a specified resource.

PATCH: Apply partial modifications to a resource.

HEAD: Retrieve headers for a specified resource without the body.

- These methods define the operation that the client wants to perform on the specified resource when making an HTTP request.

**MAC Address (Media Access Control Address):**

- A MAC address is a unique identifier assigned to a network interface controller (NIC) for communications at the data link layer of a network segment. It is a hardware address burned into the network card during manufacturing.

- MAC addresses are used for local communication within a network, while IP addresses are used for global communication across networks. MAC addresses are typically written as six groups of two hexadecimal digits, separated by colons or hyphens (e.g., 00:1A:2B:3C:4D: 5E).

These concepts are fundamental to understanding how devices communicate over networks and the protocols that facilitate this communication. They play a crucial role in the functioning of the internet and local networks.

1. **3.** Write a blog about objects and its internal representation in Javascript

**3. Write a blog about objects and its internal representation in JavaScript?**

**Introduction:**

JavaScript, a versatile and widely-used programming language, is known for its object-oriented nature. Objects are a fundamental part of JavaScript, serving as the building blocks for creating complex data structures and organizing code. In this blog post, we will dive into the fascinating world of objects in JavaScript, exploring their internal representation and how they contribute to the language's flexibility and expressiveness.

**Understanding Objects in JavaScript:**

In JavaScript, an object is a collection of key-value pairs, where each key is a string (or Symbol) and each value can be any data type, including other objects. Objects in JavaScript can be created using the object literal notation or through constructor functions.

**Internal Representation of Objects:**

JavaScript engines handle objects in a way that optimizes performance and memory usage. Let's explore some key aspects of the internal representation of objects:

1. Properties and Methods:

Properties in an object are stored as key-value pairs. Methods, which are essentially functions attached to objects, are also stored as properties with function values.

2. Prototypes:

JavaScript follows a prototype-based inheritance model. Objects can inherit properties and methods from other objects through their prototype chain. Each object has an internal link to another object called its prototype, forming a chain that goes all the way up to the “**Object. Prototype”.**

3. Hidden Classes:

JavaScript engines use a concept called hidden classes to optimize property access and assignment. When an object is created, the engine determines its hidden class based on its structure and properties. Objects with the same hidden class share similar memory layouts, improving performance.

4. Shape and Structure:

The shape of an object refers to its property names and the order in which they were added. The structure of an object is the actual memory layout, including the location of properties in memory. JavaScript engines may optimize objects based on their shape and structure to enhance performance.

5. Dynamic Nature:

JavaScript objects are dynamic, meaning you can add or remove properties at runtime. This dynamic nature is possible due to the way objects are internally represented, allowing for flexibility in code.

**Conclusion:**

Objects are at the heart of JavaScript, providing a versatile and dynamic way to structure and organize data. Understanding the internal representation of objects helps developers write efficient and performing code. As you explore the world of JavaScript further, take a closer look at objects and leverage their power to build robust and expressive applications. The internal workings may be hidden, but mastering them empowers you to make the most of JavaScript's object-oriented capabilities.