**Day-1 Task**

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

The evolution of the internet has brought about significant changes in the way data is transmitted between clients and servers. Among these changes, the transition from HTTP/1.1 to HTTP/2 has been a crucial step forward in enhancing web performance. In this blog post, we will delve into the differences between HTTP/1.1 and HTTP/2, exploring how the latter has addressed various limitations and ushered in a new era of efficient web communic ation.

HTTP/1.1:

HTTP/1.1, the foundation of the World Wide Web for many years, served its purpose admirably. However, as web content grew more complex, the limitations of HTTP/1.1 became increasingly evident. One of the primary challenges was its tendency to create a bottleneck when loading multiple resources.

Key limitations of HTTP/1.1:

Head-of-line Blocking: This occurs when one slow-loading resource on a page delays the loading of subsequent resources, impacting overall page load times.

High Latency: The protocol requires multiple connections to load resources in parallel, resulting in increased latency and slower page rendering.

Redundant Header Information: Each HTTP request includes redundant header information, leading to unnecessary overhead and increased data transfer times.

HTTP/2:

In response to the limitations of HTTP/1.1, HTTP/2 was introduced to address performance bottlenecks and improve the efficiency of data transmission. Some key features of HTTP/2 include:

Multiplexing: HTTP/2 enables multiple requests and responses to be sent in parallel over a single connection. This eliminates head-of-line blocking, allowing for more efficient resource loading and faster page rendering.

Header Compression: Unlike HTTP/1.1, which sends redundant header information with each request, HTTP/2 compresses headers, reducing the amount of data transmitted and speeding up communication between clients and servers.

Server Push: HTTP/2 introduces server push, allowing servers to proactively send resources to clients before they are explicitly requested. This feature reduces the need for additional round trips, further enhancing performance.

Binary Protocol: While HTTP/1.1 relies on a textual format, HTTP/2 uses a binary protocol, which is more efficient for both machines and humans. The binary format simplifies parsing and reduces errors, contributing to overall performance improvements.

The transition from HTTP/1.1 to HTTP/2 represents a significant leap forward in optimizing web communication. HTTP/2's multiplexing, header compression, server push, and binary protocol contribute to faster page load times and a more responsive web experience. As technology continues to evolve, it is essential for web developers and administrators to stay informed about these advancements to ensure the continued improvement of web performance. Embracing HTTP/2 can lead to a more efficient, faster, and ultimately more satisfying online experience for users worldwide.

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

JavaScript Objects:

In JavaScript, objects are dynamic, mutable collections of key-value pairs, where each key is a string (or Symbol) and each value can be any data type. Objects provide a way to structure and organize data, making them a cornerstone in JavaScript programming.

Internal Representation of Objects:

Property Bag:

At its core, an object in JavaScript is essentially a property bag—a collection of properties, where each property has a key and a corresponding value. These properties can be dynamically added, modified, or deleted at runtime.

Memory Allocation:

JavaScript engines allocate memory for objects dynamically. The internal representation involves memory storage for the object's properties and methods. Unlike some statically-typed languages, JavaScript doesn't require defining a class before creating an object, allowing for a more fluid and adaptable development process.

Properties and Methods:

Properties can hold primitive values (numbers, strings, booleans) or references to other objects, creating a hierarchy of interconnected data structures. Additionally, objects can have methods—functions attached to them—which can manipulate the object's internal state.

Prototypes and Inheritance:

JavaScript uses prototypal inheritance, allowing objects to inherit properties and methods from other objects. This mechanism contributes to the language's flexibility and enables the creation of complex object hierarchies without the need for traditional class-based inheritance.

Dynamic Nature:

One of the key features of JavaScript objects is their dynamic nature. Properties can be added or removed on the fly, providing a high degree of flexibility during runtime. This dynamic behavior aligns with the language's commitment to adaptability and ease of use.

JavaScript objects are more than just data containers; they are dynamic entities that empower developers to model and manipulate data in a highly flexible and expressive manner. Understanding the internal representation of objects sheds light on their role as the backbone of JavaScript programming, contributing to the language's adaptability and versatility. As you navigate the world of JavaScript development, embrace the power of objects and leverage their dynamic nature to build robust and scalable applications.