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

HTTP/1.1: The Old Workhorse

HTTP/1.1, the predecessor to HTTP/2, has been the backbone of the web for many years. While it served its purpose well in its time, it had certain limitations that became increasingly evident as the web grew more complex.

1. **Serial Processing**: One of the most significant limitations of HTTP/1.1 is its serial processing of requests and responses. In this version, each request had to wait for the previous one to complete before it could be sent or received, leading to a phenomenon known as the "head-of-line blocking." This slowed down web page loading times, as all resources (HTML, CSS, JavaScript, images, etc.) had to be requested and delivered sequentially.
2. **Overhead**: HTTP/1.1 carried significant overhead due to its textual nature. Headers, which contain metadata about the request or response, were verbose, leading to unnecessary data transfer and latency.
3. **Multiple Connections**: To overcome the limitations of serial processing, browsers resorted to opening multiple connections to the same server. While this improved performance to some extent, it was not an efficient solution, as it increased the load on both servers and clients.

HTTP/2: A Quantum Leap in Web Performance

To address the shortcomings of HTTP/1.1, the Internet Engineering Task Force (IETF) introduced HTTP/2 in 2015. HTTP/2 retains the same semantics as its predecessor but introduces several fundamental changes that significantly enhance web performance.

1. **Multiplexing**: One of the standout features of HTTP/2 is multiplexing. Instead of the serial processing in HTTP/1.1, HTTP/2 allows multiple requests and responses to be sent and received in parallel over a single connection. This eliminates the head-of-line blocking problem, resulting in faster page loading times.
2. **Header Compression**: HTTP/2 employs header compression techniques, reducing the overhead associated with transmitting headers. The use of binary framing also minimizes unnecessary whitespace and makes headers more compact.
3. **Server Push**: Another groundbreaking feature of HTTP/2 is server push. With server push, a server can proactively send resources to the client before they are requested. This reduces the need for additional round trips, further improving page load times.
4. **Prioritization**: HTTP/2 allows for the prioritization of resources, ensuring that critical assets like HTML and CSS are fetched before less crucial ones like images and scripts. This enhances the perceived speed of web pages.
5. **Connection Multiplexing**: Unlike HTTP/1.1, which required multiple connections to load various resources, HTTP/2 uses a single connection for all requests to a domain. This reduces the overhead associated with setting up and maintaining multiple connections.

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

In JavaScript, objects are complex data types that allow developers to represent and structure data in a flexible and dynamic way. They are collections of key-value pairs, where keys are also known as properties, and values can be of various data types, including other objects.

Objects can be created using the object literal notation or via constructors like **Object()**, and they can contain a mix of properties, which can be primitive values (e.g., strings, numbers, booleans), functions, or even other objects. This versatility makes objects a core feature of the language.

**Internal Representation of Objects**

Understanding how JavaScript represents objects internally is crucial for efficient coding and debugging. At a high level, objects are typically implemented as dictionaries or associative arrays, where each property is stored with its corresponding value. However, JavaScript engines, like V8 in Chrome or SpiderMonkey in Firefox, optimize object representation to ensure performance.

**Property Storage**

JavaScript engines use various techniques to store properties efficiently. One common approach is using hash tables or similar data structures to map property names (keys) to their associated values. These data structures allow for quick retrieval and addition of properties, making object access efficient.

**Hidden Classes and Shapes**

JavaScript engines employ techniques like hidden classes (V8) or shapes (SpiderMonkey) to optimize object property access. When an object is created, the engine assigns it a hidden class or shape based on its properties. Subsequent objects with the same structure (i.e., the same properties in the same order) share the same hidden class/shape. This optimization reduces the overhead of property access.

**Property Descriptors**

In JavaScript, each property has an associated property descriptor, which defines its characteristics (e.g., whether it's writable, enumerable, or configurable). Property descriptors are used internally by JavaScript engines to manage and control object properties.Top of Form