**1. Difference between http1 and http2:­­**

Multiplexing: HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, 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/2 does this by splitting data into binary-code messages and numbering these messages so that the client knows which stream each binary message belongs to.

Server push: Typically, a server only serves content to a client device if the client asks for it. However, this approach is not always practical for modern webpages, which often involve several dozen separate resources that the client must request. HTTP/2 solves this problem by allowing a server to "push" content to a client before the client asks for it. The server also sends a message letting the client know what pushed content to expect – like if Bob had sent Alice a Table of Contents of his novel before sending the whole thing. Header compression: Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. However, HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes from every HTTP packet. Given the volume of HTTP packets involved in loading even a single webpage, those bytes add up quickly, resulting in faster loading.

**2. Blog about objects and its internal representation in Javascript:**

In JavaScript, objects are collections of key-value pairs where the keys are strings (or symbols) and the values can be any type of data, including other objects and functions. This allows objects to model real-world entities with properties (data) and behaviors (methods). Internally, JavaScript engines like V8 (used in Chrome and Node.js) have complex mechanisms to manage and optimize objects. Understanding these internal details can help developers write more efficient code. JavaScript engines use different strategies to store object properties efficiently. Initially, objects are often represented using a simple structure called a "hidden class" (or "shape"). Hidden classes describe the layout of an object, including its properties and their types. In cases where objects have many properties, the engine might switch to a dictionary mode to handle dynamic property changes efficiently. JavaScript engines use garbage collection to automatically manage memory. When objects are no longer needed, the garbage collector reclaims the memory they occupy. Modern garbage collectors, like the one in V8, employ advanced algorithms to minimize pauses and ensure smooth performance.