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

**HTTP1.1**

* 1. **Multiplexing:**
     1. HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it.
  2. **Server push:**
     1. 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.
  3. **Header compression:**
     1. Small files load more quickly than large ones.
     2. To speed up web performance, both HTTP/1.1 compress HTTP messages to make them smaller.
     3. 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.
  4. **Pipelining and Head-of-Line Blocking:**
     1. The first response that a client receives on an HTTP GET request is often not the fully rendered page.
     2. The client discovers that the full rendering of the page requires these additional resources from the server only after it downloads the page.
     3. HTTP/1.1 takes care of this problem by introducing persistent connections and pipelining. With persistent connections, HTTP/1.1 assumes that a TCP connection should be kept open unless directly told to close.
     4. This is known as head-of-line (HOL) blocking, and is a significant problem with optimizing connection efficiency in HTTP/1.1.
     5. Adding separate, parallel TCP connections could alleviate this issue, but there are limits to the number of concurrent TCP connections possible between a client and server, and each new connection requires significant resources.
  5. **Resource Inlining:**
     1. If multiple pages require the resource, each new HTML document will have the same resource inlined in its code, leading to larger HTML documents and longer load times than if the resource were simply cached in the beginning.

**HTTP2**

* 1. **Multiplexing:**
     1. 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.
     2. 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.
  2. **Server push:** 
     1. HTTP/2 solves this problem by allowing a server to "push" content to a client before the client asks for it.
     2. 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.
  3. **Header compression:**
     1. Small files load more quickly than large ones.
     2. To speed up web performance, both HTTP/2 compress HTTP messages to make them smaller.
     3. HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets.
  4. **Advantages of the Binary Framing Layer:**
     1. HTTP/2, the binary framing layer encodes requests/responses and cuts them up into smaller packets of information, greatly increasing the flexibility of data transfer.
     2. HTTP/2 specifications do not make it mandatory to use the TLS layer, many major browsers only support HTTP/2 with HTTPS.
  5. **Stream Prioritization:**
     1. Stream prioritization not only solves the possible issue of requests competing for the same resource, but also allows developers to customize the relative weight of requests to better optimize application performance.

1. **Write a blog about objects and its internal representation in Javascript**
2. objects are quite different from JavaScript’s primitive data-types(Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data-types all store a single value each (depending on their types).
3. Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.
4. An object, is a reference data type. Variables that are assigned a reference value are given a reference or a pointer to that value.
5. That reference or pointer points to the location in memory where the object is stored. The variables don’t actually store the value.

**Creating Objects In JavaScript :**

* 1. **Create JavaScript Object with Object Literal**
     1. One of easiest way to create a javascript object is object literal, simply define the property and values inside curly braces
     2. Example : let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};
  2. **Create JavaScript Object with Constructor**
     1. Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects
     2. Using the JavaScript Keyword newthe following example also creates a new JavaScript object with four properties:
     3. Example:

var person = new Object();  
person.firstName = “John”;  
person.lastName = “Doe”;  
person.age = 50;  
person.eyeColor = “blue”;

* 1. **Using the Object.create method**
     1. Objects can also be created using the **Object.create()**  method.