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

* Multiple Requests at Once: HTTP/1.1 makes one request at a time per connection, which can slow down loading times for websites with lots of elements. HTTP/2 can handle multiple requests simultaneously within one connection, making loading faster.
* Smarter Headers: In HTTP/1.1, headers (like cookies and user agents) are sent with every request, even if they're the same for every request. HTTP/2 compresses and sends headers only once, reducing unnecessary data transfer.
* Pushing Resources: With HTTP/2, the server can push resources to the browser without waiting for a request. This speeds up loading by anticipating what the browser needs.
* Priority Loading: HTTP/2 allows websites to prioritize which resources load first, ensuring important elements like text and images load before less important ones, improving overall user experience.
* Binary Format: HTTP/2 uses a binary format to send data, whereas HTTP/1.1 uses text. This binary format is more efficient for computers to understand and process, leading to faster communication between servers and browsers.

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

* Object Basics: In JavaScript, objects are collections of key-value pairs where values can be of any data type, including other objects, arrays, functions, and primitive values like strings or numbers.
* Properties and Methods: Objects have properties (data associated with the object) and methods (functions associated with the object). Properties can be accessed or modified using dot notation (object.property) or bracket notation (object['property']).
* Internal Representation: Internally, JavaScript engines use various techniques to represent objects. One common approach is through hash tables, where keys are hashed to access values quickly. This allows for efficient storage and retrieval of properties and methods.
* Prototypes and Inheritance: JavaScript uses prototypes for inheritance. Each object has a prototype from which it inherits properties and methods. When you try to access a property or method on an object, if it doesn't exist on the object itself, JavaScript looks up the prototype chain until it finds it.
* Memory Management: Objects in JavaScript are allocated memory dynamically and are automatically garbage-collected when they are no longer in use. This means developers don't need to worry about memory management, but they should be mindful of creating unnecessary objects to prevent performance issues.