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

**Ans: Multiplexing:**

* HTTP/1.1: Uses separate connections for each resource, leading to inefficiency.
* HTTP/2: Allows multiple requests and responses within a single connection, improving resource utilization and performance.

**Header Compression:**

* HTTP/1.1: Transmits redundant header information with each request and response.
* HTTP/2: Compresses headers to minimize data transfer, reducing overhead and improving efficiency.

**Server Push:**

* HTTP/1.1: Relies solely on client-initiated requests for resource retrieval.
* HTTP/2: Employs server push, proactively sending resources to the client, reducing latency and optimizing performance.

**Binary Protocol:**

* HTTP/1.1: Relies on text-based data formats for communication.
* HTTP/2: Utilizes a binary protocol, simplifying parsing and processing, enhancing performance, and scalability.

**Backward Compatibility:**

* Both HTTP/1.1 and HTTP/2: Maintain backward compatibility.
* Ensures a smooth transition to HTTP/2 without disrupting existing infrastructure or compatibility with older clients.

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

* Objects in JavaScript are collections of key-value pairs, where keys are strings (or Symbols) and values can be any data type, including primitives, functions, or even other objects.
* Each object in JavaScript has a prototype, which serves as a template for properties and methods.
* Objects inherit properties and methods from their prototypes, forming a prototype chain.
* JavaScript engines use different memory allocation techniques to manage objects efficiently, storing them in regions like the stack or heap.
* Garbage collection algorithms reclaim memory occupied by objects that are no longer needed.
* JavaScript engines optimize property lookup using techniques like property caching and inline caching to improve performance. Minimizing property access in critical sections of code can enhance performance.
* Understanding the internal representation of objects can influence coding practices and performance optimizations
* Developers should consider factors like prototype-based inheritance, property access optimization, and memory management for efficient code design.