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

HTTP/1.1 and HTTP/2 are both versions of the Hypertext Transfer Protocol, which is the protocol used for communication between web browsers and web servers. Here are some key differences between HTTP/1.1 and HTTP/2:

1. Multiplexing:
   * HTTP/1.1**:** In HTTP/1.1, only one request can be outstanding on a connection at a time. This means that if a browser needs multiple resources (e.g., images, stylesheets, scripts) from a server, it has to open multiple connections, leading to increased latency.
   * HTTP/2: HTTP/2 supports multiplexing, which allows multiple requests and responses to be sent and received simultaneously on the same connection. This can significantly improve page load times by reducing latency.
2. Header Compression:
   * HTTP/1.1: Each request and response in HTTP/1.1 includes headers that contain metadata about the request or response. These headers are often repetitive, leading to unnecessary overhead, especially for mobile users with limited bandwidth.
   * HTTP/2: HTTP/2 uses header compression, which reduces the overhead associated with sending headers. This can result in faster page loads, particularly for websites with many small resources.
3. Binary Protocol:
   * HTTP/1.1: Uses a text-based protocol, which is human-readable but not very efficient in terms of parsing and transmission.
   * HTTP/2: Adopts a binary protocol, which is more compact and allows for more efficient parsing by machines. This results in faster and more reliable communication between browsers and servers.
4. Server Push:
   * HTTP/1.1: A server can only respond to a client's request. If the server identifies additional resources that the client will need, it has to wait for the client to request them.
   * HTTP/2: Introduces server push, which allows the server to push additional resources to the client before the client requests them. This can be beneficial for improving page load times by proactively sending resources that the server anticipates the client will need.
5. Connection Management:
   * HTTP/1.1: Requires multiple connections for parallelism, and each connection has its own overhead.
   * HTTP/2: Uses a single, multiplexed connection, reducing the need for multiple connections and their associated overhead.

In summary, HTTP/2 is designed to address the limitations of HTTP/1.1 and improve the performance of web communication by introducing features like multiplexing, header compression, a binary protocol, and server push. These improvements contribute to faster and more efficient web browsing experiences.

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

In JavaScript, objects are a fundamental data type, and they are used to represent and store collections of key-value pairs. Objects in JavaScript are dynamic, meaning you can add, modify, or remove properties at runtime. The internal representation of objects in JavaScript can be understood in terms of properties, methods, and the prototype chain.

1. Properties:
   * Objects in JavaScript consist of properties, where each property is a key-value pair. The key is a string (or a Symbol), and the value can be any valid JavaScript value, including other objects.
   * Properties can be accessed using dot notation (object.property) or bracket notation (object['property']).
   * Objects can have their own properties, and they can also inherit properties from a prototype.
2. Methods:
   * Functions that are associated with an object are called methods. Methods are essentially properties in an object where the value is a function.
   * Methods can be called on an object using the dot notation, for example: object.method().
3. Prototype Chain:
   * JavaScript objects are linked to a prototype object through what is known as the prototype chain.
   * Each object has an internal reference to its prototype. When you try to access a property or method on an object, JavaScript looks for that property or method in the object itself. If it doesn't find it, it looks in the object's prototype, and so on, forming a chain until it reaches the end of the prototype chain.
   * The prototype chain allows for property and method inheritance.