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

In today's digital age, the speed and efficiency of communication over the internet are of utmost importance. The protocol that governs this communication is HTTP (Hypertext Transfer Protocol), which defines how data is transferred between servers and clients. HTTP has evolved over the years, with the most recent versions being HTTP/1.1 and HTTP/2. In this blog, we will explore the differences between these two protocols.

Connection Management

HTTP/1.1 creates a new connection for every request/response, which means that each request/response pair must negotiate a separate connection. This can lead to latency and delays, especially when the number of requests increases. On the other hand, HTTP/2 uses a single connection for all requests/responses, which eliminates the need to negotiate a new connection for each pair. This improves performance by reducing latency and overhead.

Multiplexing

HTTP/1.1 only allows one request at a time per connection, which means that requests have to wait for previous responses before they can be sent. This can lead to a situation known as head-of-line blocking, where a slow request blocks faster requests from being processed. In contrast, HTTP/2 supports multiplexing, which allows multiple requests to be sent at the same time over a single connection. This eliminates head-of-line blocking and improves performance by enabling parallel processing of requests.

Server Push

HTTP/1.1 relies on the client to request all resources needed to render a web page. This means that the server has to wait for requests from the client before sending resources, which can lead to delays in page loading times. HTTP/2 introduces server push, which allows the server to proactively send resources to the client before they are requested. This improves page load times by reducing the number of round trips between the client and server.

Binary vs. Textual

HTTP/1.1 uses textual headers, which means that each header is transmitted as plain text. This can lead to overhead and slower performance, especially when dealing with large headers. HTTP/2 uses binary headers, which means that headers are encoded in binary format. This reduces overhead and improves performance, especially for large headers.

Security

Both HTTP/1.1 and HTTP/2 support HTTPS, which encrypts the communication between the client and server. However, HTTP/2 requires the use of TLS (Transport Layer Security) encryption, which provides improved security and protection against attacks like man-in-the-middle attacks.

In conclusion, HTTP/2 offers significant improvements over HTTP/1.1 in terms of performance, efficiency, and security. However, not all servers and clients support HTTP/2, and upgrading to HTTP/2 may require changes to existing applications and infrastructure. Nonetheless, as the internet continues to evolve and demands for faster and more efficient communication increase, it is likely that HTTP/2 will become the standard protocol for web communication.

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

In JavaScript, everything is an object, whether it is a primitive value like a number or a string, or a more complex data structure like an array or an object. Understanding how objects are represented internally in JavaScript is crucial to writing efficient and effective code. In this blog, we will explore the internal representation of objects in JavaScript.

Objects in JavaScript are represented as key-value pairs. The key is a string or a symbol, and the value can be any data type, including other objects. The keys are unique within an object, and each key corresponds to a value.

When we create an object in JavaScript, memory is allocated for the object in the heap, which is a region of memory used for dynamic memory allocation. The object is represented as a collection of properties, where each property is a key-value pair. Each property in an object is stored in a separate memory location, and the memory location is referred to as a property descriptor.

The property descriptor contains the following information:

Value: This is the actual value of the property.

Writable: This indicates whether the value of the property can be changed.

Enumerable: This indicates whether the property can be accessed in a loop.

Configurable: This indicates whether the property can be deleted or modified.

In addition to the property descriptors, each object also has a hidden class, which is a template that defines the structure of the object. The hidden class is used by the JavaScript engine to optimize access to properties in the object. When we add a new property to an object, the JavaScript engine updates the hidden class of the object to include the new property.

JavaScript engines use a technique called inline caching to optimize property access. When a property is accessed, the engine checks the hidden class of the object to determine the location of the property in memory. The location is then cached so that subsequent accesses to the property can be performed more quickly.

In summary, objects in JavaScript are represented as collections of key-value pairs, with each property stored in a separate memory location. The property descriptors contain information about the value, writability, enumerability, and configurability of the property. The hidden class is a template that defines the structure of the object and is used by the JavaScript engine to optimize property access. By understanding how objects are represented internally in JavaScript, we can write more efficient and effective code.