Write a blog on Difference between HTTP1.1 vs HTTP2

HTTP1.1

Since HTTP/1.1 was standardized in 1999, it has been the most widely used protocol for web communication. It has been a reliable server for the internet for more than 20 years, but in the age of sophisticated web apps and rich media, its performance issues are starting to show.

Its intrinsic slowness in managing several requests at once is one of HTTP/1.1's main shortcomings. Because of the sequential processing of each request under this protocol, "head-of-line blocking" occurs. The performance of the web page as a whole is impacted if one request takes a long time to finish, as this causes delays in future requests.

HTTP2

As an updated protocol created to meet the changing needs of the internet, HTTP/2 was established as a response to the drawbacks of HTTP/1.1. Multiplexing, which enables several requests and responses to be sent and received simultaneously over a single TCP connection, is one of HTTP/2's most important improvements.

HTTP/2 reduces the complexity of creating and maintaining connections by doing away with the requirement for numerous TCP connections to handle simultaneous requests. This leads to quicker page loads and enhanced overall performance. To further maximize the efficiency of data transmission, HTTP/2 also uses header compression techniques to minimize the amount of request and response headers.

Write a blog about objects and its internal representation in Javascript

In JavaScript, objects serve as the cornerstone of the language, offering a flexible and dynamic way to organize and manipulate data. Internally, JavaScript engines implement objects using hash tables, where each key-value pair is stored in a hash bucket for efficient access. Property descriptors provide metadata about each property, defining attributes such as writability and enumerability, while prototypes facilitate inheritance and code reuse through a hierarchical chain. Memory management is crucial, and modern JavaScript engines employ sophisticated garbage collection algorithms to reclaim memory occupied by unused objects.

Understanding the internal representation of objects in JavaScript provides valuable insights into how the language works under the hood. By grasping concepts such as hash tables, property descriptors, and prototypes, developers can write more efficient and maintainable code. Memory management is also essential, and awareness of garbage collection mechanisms helps prevent memory leaks and optimize performance. With this understanding, developers can harness the full power of JavaScript objects to build robust and scalable applications.