{1}: **A BLOG ON DIFFERENCE BETWEEN HTTP1.1 AND HTTP2.**

The Hypertext Transfer Protocol (HTTP) is the foundation of communication on the World Wide Web. HTTP/1.1 has been the predominant version of this protocol for more than 20 years. However, in 2015, the Internet Engineering Task Force (IETF) published HTTP/2 as the successor to HTTP/1.1. HTTP/2 brought significant improvements over its predecessor in terms of performance, security, and efficiency. In this blog post, we will discuss the differences between HTTP/1.1 and HTTP/2.

1. Multiplexing

One of the most significant changes in HTTP/2 is the introduction of multiplexing. In HTTP/1.1, each request/response pair must be completed before the next one can be sent. This means that if a page has many resources, such as images, scripts, and stylesheets, the browser must open a new connection for each resource, resulting in a slower loading time.

HTTP/2 uses a single connection between the client and server, allowing for multiple requests and responses to be sent simultaneously, improving the speed of the page load. Multiplexing reduces latency and decreases the number of round trips required to load a page, resulting in a faster user experience.

2. Server Push

Another significant improvement in HTTP/2 is the ability for the server to push resources to the client without the client requesting them explicitly. In HTTP/1.1, the client must request each resource individually, which can be slow and inefficient.

With server push, the server can anticipate the resources the client will need based on the initial request and push them proactively. This reduces the number of round trips required to load a page, further improving the speed of the page load.

3. Header Compression

HTTP/2 introduces a new header compression algorithm called HPACK. In HTTP/1.1, headers are sent in plain text, which can be verbose and redundant, resulting in larger requests and responses.

HPACK compresses the headers before they are sent, reducing the size of the requests and responses. This improves the efficiency of the protocol, reduces bandwidth consumption, and improves performance.

4. Binary Protocol

HTTP/1.1 uses a text-based protocol, which can be slow and inefficient. HTTP/2, on the other hand, uses a binary protocol, which is more efficient and requires less parsing. The binary protocol reduces the overhead of the protocol, resulting in faster performance and lower latency.

5. Security

HTTP/2 requires the use of Transport Layer Security (TLS) encryption, which provides an additional layer of security for data in transit. In contrast, TLS is optional in HTTP/1.1, making it more vulnerable to attacks such as man-in-the-middle.

Conclusion:

HTTP/2 is a significant improvement over HTTP/1.1 in terms of performance, security, and efficiency. Multiplexing, server push, header compression, binary protocol, and mandatory TLS encryption are some of the key features that make HTTP/2 a better protocol for the modern web. While HTTP/1.1 is still widely used, it is slowly being phased out in favour of HTTP/2, which offers a faster and more secure browsing experience.

{2}: A blog about objects and its internal representation in JavaScript.

In JavaScript, everything is an object. Objects are the fundamental building blocks of JavaScript, and they play a central role in the language's syntax, semantics, and functionality.

In simpler words, objects in JavaScript are like containers that hold related pieces of data or functionality, represented as key-value pairs. For example, an object representing a person might have keys for the person's name, age, and address, with values for each of these properties.

Internally, JavaScript objects are implemented as dynamic hash tables, which allow for efficient lookup and insertion of key-value pairs. This means that when you access a property of an object, JavaScript uses a hashing function to quickly find the corresponding value.

Additionally, objects in JavaScript can inherit properties and methods from other objects, which makes them very flexible and extensible. When you create a new object using a constructor function or class, it's actually based on a prototype object that defines its properties and methods.

Finally, it's important to note that objects in JavaScript are reference types, which means that when you assign an object to a variable or pass it as a function argument, you're actually passing a reference to the object's location in memory, rather than a copy of the object itself.

There are various methods of representing objects in JavaScript like object literals , constructor function and classes. While object literals are the most common and mostly used way to represent object. Constructor function and classes are more complex methods but they are more feature rich.