|  |  |
| --- | --- |
| **HTTP1.1** | **HTTP2** |
| 1. No multiplexing support. Each request/response requires a separate connection. | 1. Supports multiplexing, allowing multiple streams over a single connection, improving efficiency |
| 1. Headers are uncompressed, leading to larger data size and slower performance. | 1. Utilizes header compression (HPACK) to reduce overhead, resulting in faster data faster |
| 1. Requests are processed in the order they are received, leading to potential delays. | 1. Allows for request prioritization, enabling more important resources to be loaded first. |
| 1. Limited parallel connections per domain, leading to a slower loading experience. | 1. Single connection for multiple requests, reducing latency and improving speed. |
| 1. Text-based protocol, which can be inefficient for certain data types. | 1. Binary framing layer enhances efficiency by reducing parsing complexity. |
| 1. No support for server push. | 1. Supports server push, allowing the server to send multiple responses for a single request. |
| 1. It works on the textual format. | 1. It works on the binary protocol. |
| 1. There is head of line blocking that blocks all the requests behind it until it doesn’t get its all resources. | 1. It allows multiplexing so one TCP connection is required for multiple requests. |
| 1. It uses requests resource Inlining for use getting multiple pages | 1. It uses PUSH frame by server that collects all multiple pages |
| 1. It compresses data by itself. | 1. It uses HPACK for data compression. |

1. A blog on Difference between HTTP1.1 vs HTTP2

Ans:

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

Ans:

**Objects in JavaScript**

In JavaScript, an object is a complex data type that allows you to store and organize data using key-value pairs. Objects can represent real-world entities and their properties, making them a powerful tool for modelling various scenarios in your code. Objects are used for everything from organizing data in a structured manner to creating instances of classes in object-oriented programming.

JavaScript follows a prototype-based model, which means that objects can inherit properties and methods from other objects through their prototype chain. This chain is created through the prototype property, allowing objects to share and override behaviour.

In JavaScript, objects are the building blocks of data representation and manipulation. Whether you are dealing with simple key-value pairs or creating complex prototypes, a solid understanding of how objects are structured and stored internally is essential. This knowledge not only helps you write more efficient code but also enables you to leverage the full power of JavaScript's object-oriented capabilities. So, the next time you work with objects in JavaScript, remember to peek beneath the surface and appreciate the intricate internal representation that makes them so versatile and powerful.

**Example:**

var car =

{

wheels: 4,

colour: "red",

mileage: 15,

brand: "BMW"

}

console.log(car.brand);