1. **Difference between HTTP1.1 vs HTTP2**

The first usable version of http was created in 1997. The first version of http was called HTTP 1.1. This version is still on the web.

In 2025 a new version was created. It was created to solve many of the problems that were not anticipated when http 1.1 was created. In Particular HTTP2 is faster and more efficient than http 1.1 The Differences are as follows

The Table below displays the major differences between http1.1 and http2

| **Feature** | **HTTP/1.1** | **HTTP/2** |
| --- | --- | --- |
| Multiplexing | No multiplexing. Supports one request per connection. | Supports multiplexing. Multiple requests and responses can be sent and received on the same connection simultaneously. |
| Binary Protocol | Data is transmitted as plaintext. | Employs a binary framing layer, which enhances efficiency and reduces parsing complexity. |
| Server Push | Not supported. | Supports server push, allowing servers to push responses to the client before the client requests them. |
| Stream Prioritization | No support for prioritisation. Requests are processed in the order they are received. | Supports stream prioritisation, enabling clients to specify the importance of individual resources for better performance optimization. |
| Connection Usage | Requires multiple connections for parallelism, leading to higher resource consumption. | Requires only a single connection per origin, reducing resource consumption and improving efficiency. |
| Server Load | One request per connection can lead to higher server load due to the need for multiple connections. | Multiplexing reduces the number of connections needed, lowering server load and resource consumption. |
| Latency Reduction | Limited ability to reduce latency due to the sequential nature of requests and responses. | Multiplexing and other optimizations in HTTP/2 help reduce latency, improving overall performance. |

1. **objects and its internal representation in Javascript**

In JavaScript, objects are a fundamental data type used to store collections of key-value pairs. They are versatile and can represent various entities, from simple data structures to complex entities in an application. Internally, objects in JavaScript are typically implemented using hash tables or similar data structures for efficient key-based access.

Here's a brief overview of how objects and their internal representation work in JavaScript:

**Object Declaration:**

Objects are declared using curly braces {} syntax, and properties are defined inside them using key-value pairs.

**Key-Value Pairs:**

* Each property of an object has a key (also called a property name) and a corresponding value. Keys are strings (or symbols), and values can be of any data type.

**Internal Representation:**

* Internally, JavaScript engines typically use hash tables to implement objects. Hash tables allow for fast lookup of values based on their keys. When you access a property of an object, the JavaScript engine hashes the property name to determine its storage location in memory, making property access very efficient.

**Dynamic Nature:**

* One of the key features of JavaScript objects is their dynamic nature. Properties can be added, modified, or deleted at runtime.