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

**Header compression:**

Both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. But HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes from every HTTP packet.

**Buffer Overflow:**

Server and client machine TCP connection requires both of these to have a certain buffer space for holding incoming requests.

Though these buffers can hold numerous or large requests, they may also lack space due to small or limited buffer size. It causes [**buffer overflow**](https://www.wallarm.com/what/buffer-overflow-attack-definition-types-use-by-hackers-part-1)at receiver’s end, resulting in data packet loss. For example, packets received after the buffer is full, will be lost.

To prevent it from happening, a flow control mechanism stops the sender from transmitting an overwhelming amount of data to the receiver side.

The flow control mechanism in **HTTP/1.1** relies on the basic TCP connection. In beginning itself, both the machines set their buffer sizes automatically. If the receiver’s buffer is full, it shares the receive window details, telling how much available space is left. The receiver acknowledges the same and sends an opening signal.

Note that flow control can only be implemented on either end of the connection. Moreover, since HTTP/1.1 uses a TCP connection, each connection demands an individual flow control mechanism.

**HTTP/2** multiplexes data streams utilizing the same (one) TCP connection. So, in this case, both machines can implement their flow controls instead of using the transport layer. The application layer shares the available buffer size data, after which, both machines set their receive window details on the multiplexed streams level. In addition, the flow control mechanism does not need to wait for the signal to reach its destination before modifying the receive window.

**Background:**

For better contextualization of the certain alterations that HTTP/2 made to its precursor, we’ll take a quick look at their basic functionalities and development details first.

**HTTP/1.1**

HTTP protocol was developed in 1989 as the common language that enables client and server machines’ interaction. Process steps are as enlisted:

1. The client (browser) has to send a request to the server using the method (GET/POST).
2. Server responds with the requested resource, for example – image, alongside the status of what it did to the client’s request.

Keep in mind that this is not a one-time process. Such requests and responses needs to be transferred between both these machines until the client receives all the resources, essential to load a web page on the end-user’s (your) screen.

This request-response exchange can be regarded as an IP stack being handled by transfer layer and networking layers before finally reaching to the application layer. Now, let’s see how HTTP/2 handles the same scenario.

**HTTP/2**

HTTP/2 was released at Google as the significant improvement of its predecessor. It was initially modeled after the SPDY protocol and went through significant changes to include features like multiplexing, header compression, and stream prioritization to minimize page load latency. After its release, Google announced that it would not provide support for SPDY in favor of HTTP/2.

The major feature that differentiates HTTP/2 from HTTP/1.1 is the binary framing layer. Unlike HTTP/1.1, HTTP/2 uses a binary framing layer. This layer encapsulates messages – converted to its binary equivalent – while making sure that its HTTP semantics (method details, header information, etc.) remain untamed. This feature of HTTP/2 enables gRPC to use lesser resources.

**Objects and its internal representation in Javascript:**

An object is a collection of properties, and a property is an association between a name (or key) and a value pairs. A property's value can be a function, in which case the property is known as a method.

The **syntax** of creating object using object literal is given below:

Property and value is separated by colon(:).

**Example:**

Let person={

firstname : ”Dhanasekaran”,

lastname : “M”,

age : 35

};