| 1. **DIFFERENCE BTW**   **HTTP/1** | **HTTP/2** |
| --- | --- |
| It works on the textual format. | It works on the binary protocol. |
| There is head of line blocking that blocks all the requests behind it until it doesn’t get its all resources. | It allows multiplexing so one TCP connection is required for multiple requests. |
| It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |

**2)Objects and its internal representation in Javascript**

Object:

In JavaScript, an object is a standalone entity, with properties and type. Compare it with a cup, for example. A cup is an object, with properties. A cup has a color, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.

**Creating Objects in JavaScript:**

1. By object literal
2. By creating instance of Object directly (using new keyword)

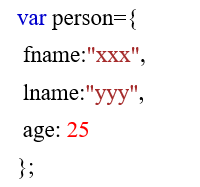
**By object literal:**

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



Property and value is separated by colon(:).

**Example:**



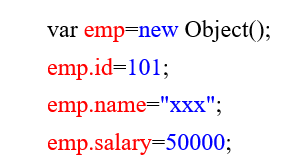
**By creating instance of Object directly (using new keyword):**

The syntax of creating object directly is given below:



Here, **new keyword** is used to create object.

**Example:**



**Accessing JavaScript Objects:**

The syntax for accessing the property of an object is

*objectName.property*

or

*objectName*[“*property*”]

Accessing ‘fname’ from example 1 using dot operator,



Accessing ‘name’ form example 2 using [],



**BLOG DIFFERENCE BTW HTTP1 AND HTTP2**

**HTTP1**

**.It was no longer required for each connection to be terminated immediately after y request was served with a response; instead, with the keep-alive header, it**

**BLOG DIFFERENCE BTW HTTP1 AND HTTP2**

**HTTP1**

**.It was no longer required for each connection to be terminated immediately after every request was served with a response; instead, with the keep-alive header, it**

**was possible to have persistent connections. It allowed multiple requests/responses per TCP connect**

.**HTTP/1.1 provided support for chunk transfers that allowed streaming of content dynamically as chunks and for additional headers to be sent after the message body. This enhancement was particularly useful in cases where values of a field remained unknown until the content had been produced. For example, when the content had to be digitally signed, it was not possible to do so before the entire content gets generated.**

HTTP2

. It introduces the concept of a server push where the server anticipates the resources that will be required by the client and pushes them prior to the client making requests. The client retains the authority to deny the server push; however, in most cases, this feature adds a lot of efficiency to the process.

. It is a binary protocol i.e. only binary commands in the form of 0s and 1s are transmitted over the wire. The binary framing layer divides the message into frames that are segregated based on their type – Data or Header. This feature greatly increases efficiency in terms of security, compression and multiplexing.