HTTP (Hypertext Transfer Protocol) serves as the backbone of communication between web servers and clients, facilitating the exchange of information across the World Wide Web. As technology advances and demands for faster, more efficient web experiences increase, protocols evolve to meet these needs. Two such iterations of HTTP, namely HTTP/1.1 and HTTP/2, have played pivotal roles in shaping the modern web.

**HTTP/1.1:**

HTTP/1.1, standardized in 1997, has been the workhorse of the web for over two decades. Despite its reliability, HTTP/1.1 exhibits certain limitations that hinder optimal performance, particularly in the context of modern web applications:

Multiplexing : In HTTP/1.1, each request-response cycle typically requires its own separate connection. This can lead to a phenomenon known as the "head-of-line blocking," where the transmission of one resource is delayed by the presence of other resources in the queue

Resource Prioritization: :Lacks built-in support for request prioritization, meaning that all resources are treated with equal importance. Consequently, critical resources such as stylesheets or JavaScript files may be delayed behind less crucial assets.

Server Push: Does not support server push

.**HTTP/2 :**

Recognizing the need for a more efficient communication protocol, HTTP/2 was developed to address the shortcomings of its predecessor while leveraging advancements in web technologies. Here's how HTTP/2 differs from HTTP/1.1:

Multiplexing: HTTP/2 supports multiplexing, allowing multiple requests and responses to be sent and received over a single connection concurrently. This feature eliminates head-of-line blocking and enhances the overall efficiency of resource delivery.

Resource Prioritization:Introduces stream prioritization, enabling developers to specify the importance of different resources and ensure that critical assets are delivered promptly.

Server Push: Introduces server push, a mechanism that allows servers to proactively send resources to clients before they are requested

2.objects and its internal representation in Javascript

Objects are important data types in javascript. Objects are different than primitive datatypes (i.e. number, string, boolean, etc.). Primitive data types contain one value but Objects can hold many values in form of Key: value pair. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

**There are four ways To create a object :**

## **1.Create JavaScript Object with Object Literal :**

One of easiest way to create a javascript object is object literal, simply define the property and values inside curly braces

Example :

let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};

## **2.Create JavaScript Object with Constructor:**

Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor

## **3.Using the JavaScript Keyword new :**

Create new javascript object with “new” keyword

var person = new Object();

person.firstName = “John”;

## **4.Using the Object.create method :**

Objects can also be created using the Object.create() method. This method can be very useful, because it allows you to choose the prototype object for the object you want to create, without having to define a constructor function.

Prototype is nothing but we can access another object properties