**1. Write a blog on Difference between HTTP1.1 vs HTTP2.**

Over the years, the internet protocols, such as HTTP, have developed, with versions like HTTP/1.1 and HTTP/2 playing a big role in shaping how we browse. Their differences greatly affect how fast websites load and how we experience the internet. Understanding these distinctions helps us see how they affect our online activities.

**HTTP/1.1:** It was crucial but had performance limitations.

**Multiplexing and Parallel Requests:**

HTTP/1.1 faced "head-of-line blocking," limiting simultaneous connections, slowing resource delivery.

**Header Overhead:**

HTTP/1.1 requests had redundant headers, causing latency, especially with multiple small requests.

**No Server Push:**

Server inability to initiate data transfer impacted web performance, requiring clients to request all resources themselves.

**HTTP/2:** from 2015, aimed to fix its predecessor's flaws with major upgrades.

**Multiplexing and Binary Framing:**

HTTP/2 enabled multiple data streams in one connection, removing blockages and boosting resource loading speed.

**Header Compression:**  
Using HPACK, HTTP/2 cut redundant header data, making transmission more efficient by reducing overhead.

**Server Push:**  
HTTP/2's standout feature, server push, sends resources to clients before requests, enhancing performance.

**Stream Prioritization:**

HTTP/2 supports stream prioritization, allowing more critical resources to be delivered faster, enhancing user experience.

**Conclusion:**

Transitioning from HTTP/1.1 to HTTP/2 marks a big step in web protocol advancement. With its improvements in multiplexing, header compression, and server push, HTTP/2 has transformed web performance, making browsing faster and more efficient. Embracing such modern protocols is crucial for delivering top-notch digital experiences in today's evolving online world.

Top of Form

**2. Write a blog about Objects and its representation in javascript.**

Objects in JavaScript differ from primitive data types (Number, String, Boolean, null, undefined) as they can store multiple values, unlike primitives which store a single value each based on their types.

Furthermore, objects allow nesting other objects within them, accommodating numerous data sets.

Each data set is added in a key-value pair format, separated by commas, offering flexibility to include any number of such pairs.

Example:

Let bike = {Model Name: “Kawasaki”, model: 2019, sports: {Engine: “Gas”}}

Accessing an object's properties involves two methods: the straightforward dot-notation and the bracket notation.

Example:

Console.log(bike.name) = Kawasaki

Console.log (bike. sports. Engine) = Gas

Bracket notation:

Console.log(bike[“name”]) = Kawasaki

--------------------------------END----------------------------