**TASK**

1.) Blog on difference between HTTP1.1 vs HTTP2

The **Hypertext Transfer Protocol (HTTP)** is the foundation of the World Wide Web, and is used to load webpages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack. A typical flow over HTTP involves a client machine making a request to a server, which then sends a response message.

An HTTP method, sometimes referred to as an HTTP verb, indicates the action that the HTTP request expects from the queried server. For example, two of the most common HTTP methods are ‘GET’ and ‘POST’; a ‘GET’ request expects information back in return (usually in the form of a website), while a ‘POST’ request typically indicates that the client is submitting information to the web server (such as form information, e.g. a submitted username and password).

* The evolution of the web is marked by continual improvements in protocols and technologies that drive seamless communication between clients and servers. In this journey, the transition from HTTP/1.1 to HTTP/2 has been a pivotal moment, ushering in a new era of enhanced performance and efficiency.

**HTTP/1.1: The Workhorse of the Web**

HTTP/1.1, the stalwart of web communication for over a decade, has been the backbone of data transfer on the internet. While it has served its purpose admirably, certain limitations have prompted the need for a more sophisticated protocol.

**Key Limitations of HTTP/1.1:**

1. **Sequential Resource Loading:**

- In HTTP/1.1, each resource (e.g., images, scripts, stylesheets) requires a separate connection. This sequential loading can result in latency and slower page rendering.

2. **Verbose Headers:**

- Headers in HTTP/1.1 are text-based and not compressed, leading to larger file sizes and increased bandwidth usage.

3. **No Built-in Prioritization:**

- Without native support for prioritization, critical resources may be delayed by less important ones, affecting overall page load times.

**HTTP/2: Overcoming the Limitations**

HTTP/2, developed to address the shortcomings of its predecessor, introduces several features aimed at improving performance and user experience.

**Key Features of HTTP/2:**

1. **Multiplexing:**

- One of the most significant improvements is the introduction of multiplexing in HTTP/2. Unlike HTTP/1.1, which opens separate connections for each resource, HTTP/2 allows multiple resources to be transmitted concurrently over a single connection. This significantly reduces latency and speeds up page loading times.

2. **Header Compression:**

- HTTP/2 employs header compression, reducing the size of headers and minimizing the overhead associated with each request. This leads to more efficient use of bandwidth and faster data transfer.

3. **Server Push:**

- A groundbreaking feature of HTTP/2 is server push. Servers can proactively send resources to the client before they are explicitly requested. This can preemptively provide necessary assets, further accelerating page load times.

4. **Prioritization:**

- HTTP/2 introduces stream prioritization, enabling the server to assign importance to different resources. This ensures that critical elements are delivered and processed first, optimizing the user experience, especially on complex web pages.

5. **Binary Protocol:**

- Unlike the plain text nature of HTTP/1.1, HTTP/2 uses a binary protocol. This binary framing simplifies parsing and processing, contributing to improved performance.

* The shift from HTTP/1.1 to HTTP/2 marks a significant milestone in the evolution of web protocols. HTTP/2's multiplexing, header compression, server push, prioritization, and binary protocol collectively contribute to a more efficient, responsive, and faster web experience. While HTTP/1.1 has been foundational, the demands of modern web applications and the need for optimized performance have made HTTP/2 the new standard. Embracing HTTP/2 not only future-proofs websites but also ensures that users can enjoy a smoother and more seamless online experience. As we bid farewell to HTTP/1.1, the adoption of HTTP/2 paves the way for a more streamlined and dynamic internet landscape.

2.) Blog about Objects and its Internal representation in JavaScript

* JavaScript, as a versatile and widely-used programming language, employs a variety of data types to represent and manipulate information. One of the fundamental and powerful data structures in JavaScript is the object. Objects play a crucial role in organizing and managing data within the language. In this blog, we'll delve into the concept of objects in JavaScript, exploring their internal representation and the mechanisms that make them so essential in modern web development.

**Object:** In JavaScript, an object is a composite data type that allows you to store and organize data in key-value pairs. Unlike primitive data types such as numbers or strings, objects can hold different types of values and even other objects, making them highly flexible and dynamic. Objects are used to model real-world entities and represent complex structures in the code. In JavaScript, almost "everything" is an object. Booleans, Numbers, Strings, Dates, Maths, Regular expressions, Arrays, Functions , Objects are always objects.

**Internal Representation of Objects:**

Understanding how objects are internally represented in JavaScript is crucial for effective programming and optimization. The key components of an object's internal structure are:

1. **Properties:**

- Objects are defined by their properties, which are essentially key-value pairs.

- Each property consists of a name (or key) and a corresponding value. The value can be of any data type, including other objects or functions.

2. **Methods:**

- Objects can contain functions as values, referred to as methods.

- Methods represent actions that can be performed on the object or computations related to the object's data.

3. **Prototype Chain:**

- JavaScript uses a prototype-based inheritance model, where objects can inherit properties and methods from other objects.

- The prototype chain allows objects to share and reuse functionality, enhancing code reusability and maintainability.

4. **Hidden Classes:**

- JavaScript engines use optimization techniques like hidden classes to improve the performance of object property access.

- Hidden classes help organize and optimize memory storage for objects with similar structures, reducing the overhead of property access.

**Creating Objects in JavaScript:**

JavaScript variables can also contain many values. Objects are variables too. But objects can contain many values.

-Object values are written as **name : value** pairs (name and value separated by a colon).

**let person = {firstName:"Rakesh", lastName:"E", age:25, eyeColor:"brown"};**

Objects in JavaScript can be created using various syntaxes. The most common is the object literal notation:

// Object literal notation

-Using an object literal, you both define and create an object in one statement. The following example creates a new JavaScript object with four properties:

**const person = {firstName:"Rakesh ", lastName:"Elumalai", age:25, eyeColor:"brown"};**

-Spaces and line breaks are not important. An object definition can be in multiple lines:

**const person = {};  
person.firstName = "Rakesh";  
person.lastName = "Elumalai";  
person.age = 50;  
person.eyeColor = "Brown";**

-Another way to create objects is through the `Object` constructor:

// Object constructor

**const car = new Object();**

**car.make = 'Honda';**

**car.model = 'Brio';**

**car.year = 2022;**

Objects are dynamic, allowing you to add, modify, or delete properties and methods at runtime, providing great flexibility in handling data.

**Use Cases and Best Practices:**

1. **Organizing Data:**

- Objects are ideal for modeling entities with multiple properties. For example, a `Person` object might have properties like `name`, `age`, and `address`.

2. **Encapsulation:**

- Encapsulating related data and behavior within an object promotes code organization and reduces global namespace pollution.

3. **JSON (JavaScript Object Notation):**

- Objects are closely related to JSON, a widely used data interchange format. JSON objects resemble JavaScript objects, facilitating seamless data exchange between server and client.

* Objects in JavaScript serve as a fundamental building block for structuring and managing data. Their flexibility and dynamic nature make them a powerful tool for developers. Understanding the internal representation of objects, including properties, methods, prototype chains, and hidden classes, is crucial for writing efficient and maintainable code. Embrace the versatility of objects in JavaScript to enhance your programming skills and create robust, organized, and scalable applications.