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

Ans:

\*Request and Response Multiplexing:

HTTP/1.1: This version relies on a single request-response mechanism. Only one request can be outstanding on a connection at a time, and responses must be received in the order they were requested.

HTTP/2: It introduces multiplexing, allowing multiple requests and responses to be processed in parallel over a single connection. This significantly improves the loading speed of web pages, especially for resources such as images, stylesheets, and scripts.

\*Header Compression:

HTTP/1.1: Headers are sent with each request and response, leading to redundancy and increased overhead.

HTTP/2: It uses header compression to reduce redundancy and lower the amount of data transferred over the network. This results in faster page loads and improved efficiency.

\*Binary Protocol:

HTTP/1.1: It uses plain text for communication, which can be verbose and less efficient.

HTTP/2: It employs a binary protocol, which is more compact and easier for machines to parse. This reduces latency and improves performance.

\*Server Push:

HTTP/1.1: The server cannot push resources to the client without a specific request.

HTTP/2: Server push allows the server to send additional resources to the client preemptively, improving page load times by eliminating the need for multiple round-trip requests.

\*Connection Handling:

HTTP/1.1: Requires multiple connections to load multiple resources concurrently, leading to increased latency.

HTTP/2: Supports multiplexing over a single connection, reducing the number of connections and improving efficiency.

2. Write a blog about objects and its internal representation in Javascript?

Ans:

In JavaScript, objects are composite data types that encapsulate data and behavior. They are essentially containers for key-value pairs, where keys are strings (or symbols) and values can be of any data type, including other objects. This key-value pair structure makes objects an ideal choice for representing complex entities.

Internal Representation of Objects

Internally, JavaScript objects use various mechanisms for their representation, with the most common being the hash table. Let's explore some key aspects of the internal representation:

1. Properties and Methods:

Properties: Represent data stored within an object. They can be accessed using dot notation (object.property) or bracket notation (object['property']).

Methods: Represent functions or behavior associated with an object. They are invoked using the object's name followed by parentheses (object.method()).

2. Prototype Chain:

JavaScript follows a prototype-based inheritance model. Each object has a prototype, which serves as a template for properties and methods. If an object doesn't have a specific property or method, JavaScript looks up the prototype chain until it finds the desired property or until it reaches the end of the chain (usually the base Object).

3. Hash Tables:

Objects use hash tables for efficient property access. The keys are hashed to determine the memory location where the corresponding value is stored. This allows for constant-time average complexity for property access.

4. Dynamic Nature:

Objects in JavaScript are dynamic, meaning properties can be added or removed at runtime. This flexibility enables developers to adapt objects based on the evolving requirements of their applications.

Example:

// Creating an object

const car = {

make: 'Toyota',

model: 'Camry',

year: 2022,

start: function() {

console.log('Engine started!');

}

};

// Accessing properties and invoking a method

console.log(car.make); // Output: Toyota

car.start(); // Output: Engine started!

// Adding a new property dynamically

car.color = 'Blue';

// Internal representation involves updating the hash table with the new property.