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

The evolution of the World Wide Web has been marked by constant innovation and improvement in communication protocols. One significant leap in this journey is the transition from HTTP/1.1 to HTTP/2. While both versions serve as fundamental protocols for transferring data over the web, they differ in various aspects, aiming to enhance performance, efficiency, and user experience.

****HTTP/1.1:****

* In the traditional HTTP/1.1, only one request can be processed at a time per connection. If a browser wants to download multiple resources simultaneously, it has to establish multiple connections, leading to potential delays and increased latency.
* Headers, which contain information about the request and response, are sent as plain text in each interaction. This redundancy results in increased data transfer and longer loading times.
* HTTP/1.1 relies on a text-based protocol, making it human-readable but less efficient for machines to process.
* In HTTP/1.1, resources are loaded in the order they are requested. There is no inherent mechanism to prioritize critical resources over non-critical ones.
* In HTTP/1.1, the server cannot push content to the client unless requested, leading to potential delays in resource loading.

****HTTP/2:****

* HTTP/2 introduces multiplexing, allowing multiple streams of data to be sent and received simultaneously over a single connection. This minimizes latency and enhances the overall efficiency of data transfer.
* HTTP/2 utilizes header compression, significantly reducing the overhead associated with headers. This optimization leads to faster loading times and more efficient use of network resources.
* HTTP/2 is a binary protocol, which means that data is represented in a format that is more compact and can be processed faster by machines. This results in improved performance and reduced latency.
* HTTP/2 introduces stream prioritization, allowing developers to specify the importance of each resource. This ensures that critical resources are loaded first, optimizing the user experience.
* HTTP/2 supports server push, enabling the server to push resources to the client before they are explicitly requested. This can further reduce latency and improve page load times.

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

JavaScript, a versatile and widely-used programming language, owes much of its flexibility and power to its unique handling of objects. Objects play a pivotal role in JavaScript, serving as the building blocks for complex data structures and enabling the creation of dynamic and interactive web applications. In this blog, we'll delve into the fascinating world of objects and explore their internal representation in JavaScript.

**Objects in JavaScript:**

In JavaScript, an object is a composite data type that allows you to store and organize data in key-value pairs. Objects can represent real-world entities and their properties, making them a crucial component of the language.

**Example:**

let car = {

brand: 'Toyota',

model: 'Camry',

year: 2022,

color: 'Blue'

};

Objects in JavaScript can contain various data types as values, including strings, numbers, arrays, and even other objects. This flexibility makes them incredibly versatile for modeling diverse scenarios.

**Internal Representation of Objects :**

JavaScript engines, like V8 in Chrome or Spider Monkey in Firefox, handle the internal representation of objects. Let's explore the key aspects of how objects are represented internally:

1. **Properties and Methods :**

* Each object has properties, which are essentially key-value pairs that store data.
* Methods, on the other hand, are functions associated with an object. They can perform actions or provide functionality related to the object.

1. **Memory Allocation :**

* Objects are allocated memory dynamically. Memory is assigned based on the properties and methods present in the object.
* The size of the object can change during its lifecycle as properties are added or removed.

1. **Prototypes :**

* JavaScript objects are often linked to a prototype object. If a property or method is not found in the object itself, the JavaScript engine looks for it in the object's prototype chain.
* This prototype chain allows for inheritance, enabling objects to inherit properties and methods from their prototypes.

1. **Hidden Classes :**

* JavaScript engines use a concept called hidden classes to optimize object property access and manipulation.
* When an object is created, the JavaScript engine assigns it a hidden class, which helps in optimizing property access and improves performance.