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

* **Answer:**

HTTP (**Hypertext Transfer Protocol**) is the underlying protocol of the World Wide Web, enabling communication between web servers and clients. The latest versions of HTTP are HTTP1.1 and HTTP2, which have distinct differences in terms of their features, capabilities, and performance. In this blog, we will explore the key differences between HTTP1.1 and HTTP2.

* **HTTP1.1:**

HTTP1.1 was first introduced in 1999 and has been the primary protocol used on the web for many years. It is a stateless protocol, meaning that each request and response cycle is independent of the previous ones. This means that for each request, a new connection needs to be established between the server and the client.

1. Connection handling: In HTTP1.1, a new connection is required for each request, which can lead to slower loading times for websites with many requests.
2. Header compression: HTTP1.1 uses textual headers, which can be verbose and add to the data transfer size. It also lacks proper header compression techniques, which can increase the data transfer size further.
3. Multiplexing: HTTP1.1 does not support multiplexing, meaning that only one request can be processed at a time, which can lead to slow page loading times.
4. Server push: HTTP1.1 does not support server push, meaning that the server cannot push resources to the client before they are requested.

* **HTTP2:**

**HTTP2** is the latest version of the HTTP protocol, which was released in 2015. It aims to improve upon the limitations of HTTP1.1 and provide better performance, speed, and security. Here are some of the key features of HTTP2:

1. Multiplexing: HTTP2 supports multiplexing, which means that multiple requests can be sent and received over a single connection, reducing the number of connections required to load a page.
2. Header compression: HTTP2 uses binary headers, which are more efficient than textual headers, and proper header compression techniques to reduce the data transfer size.
3. Server push: HTTP2 supports server push, which means that the server can push resources to the client before they are requested, reducing the page loading time.
4. Prioritization: HTTP2 allows for requests to be prioritized, meaning that the most important requests can be processed first, leading to faster page loading times.
5. Security: HTTP2 requires the use of HTTPS, which provides better security and privacy than HTTP1.1.

* **Conclusion**:

HTTP2 is a significant improvement over HTTP1.1, with features such as multiplexing, header compression, server push, prioritization, and improved security. These features help to improve website performance, reduce page loading times, and enhance the user experience. As a result, HTTP2 is becoming increasingly popular, and many web servers and browsers now support it. However, HTTP1.1 is still widely used, and websites may continue to support it for backward compatibility with older browsers and devices.

**The below table explains the key features difference between HTTP1.1 and HTTP2;**

|  |  |  |
| --- | --- | --- |
| **Feature** | **HTTP1.1** | **HTTP2** |
| Connection Handling | Each request requires a new connection | |  |  | | --- | --- | |  | Multiple requests can be sent over a single connection | |
| Header Compression | Textual headers with no proper compression techniques | Binary headers with proper compression techniques |
| Multiplexing | Does not support multiplexing, only one request can be processed at a time | Supports multiplexing, allowing for multiple requests to be processed simultaneously |
| Server Push | Does not support server push | Supports server push, allowing the server to push resources to the client before they are requested |
| Prioritization | No prioritization available | Allows for requests to be prioritized |
| Security | No requirements for HTTPS | Requires the use of HTTPS |

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

* **Answer:**

Javascript is an **object-oriented programming language**, which means that objects are a fundamental part of the language. In fact, almost everything in Javascript is an object, including functions, arrays, and even primitives like strings and numbers. In this blog, we'll explore what objects are in Javascript, and how they are represented internally.

* **What are objects in Javascript**?

In Javascript, an object is a collection of key-value pairs, where the key is a string, and the value can be of any type. Objects can be created using object literals, constructor functions, or the ES6 class syntax.

**Example of an object literal:**

const person = {

name: "John",

age: 30,

address: {

street: "123 Main St",

city: "Anytown",

state: "CA"

}

};

In this example, we have an object called **person** that has three properties: **name, age**, and **address.** The **address** property is itself an object that contains three properties.

* Internal representation of objects in Javascript Internally, objects in Javascript are represented as collections of properties, where each property is represented by a string key and a value. Objects can have properties that are functions, also known as methods. When an object is created, it is assigned a unique identity that distinguishes it from other objects. This identity is known as the object's memory address.
* Javascript engines use a variety of techniques to optimize the performance of object property access and manipulation. One common technique is to use a hash table to store the object's properties. A hash table is a data structure that provides constant-time lookup of values based on a key.
* When a property is accessed or modified, the engine first looks up the property's key in the hash table. If the key is found, the corresponding value is returned or updated. If the key is not found, a new property is created and added to the object's hash table.
* Another technique used by some engines is called inline caching. When a property is accessed or modified multiple times, the engine may optimize the code by caching the property's location in memory, so that subsequent access or modification can be performed more quickly.
* **Conclusion:**

In Javascript, objects are a fundamental part of the language, and are used to represent data and behavior. Internally, objects are represented as collections of properties, stored in a hash table. The efficient manipulation of these properties is critical to the performance of Javascript programs, and modern engines use a variety of techniques to optimize this process. Understanding the internal representation of objects in Javascript can help you write more efficient and performant code.