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

HTTP, which stands for Hypertext Transfer Protocol, is the protocol used by the World Wide Web to exchange information between web servers and clients. Since its inception, HTTP has undergone several versions, with HTTP/1.1 being the most widely used until the release of HTTP/2 in 2015. Below are the key differences between HTTP/1.1 and HTTP/2.

**Multiplexing**

One of the most significant differences between HTTP/1.1 and HTTP/2 is the way they handle requests and responses.

In HTTP/1.1, each request/response transaction requires a separate connection to be established, which can be time-consuming and resource intensive.

HTTP/2 uses multiplexing, which means multiple requests can be sent and received over a single connection simultaneously. This significantly reduces latency and improves performance, especially for websites with many resources to load.

**Header Compression**

HTTP/1.1 uses plain text for headers, which can be verbose and result in increased data transfer.

HTTP/2 compresses headers, which reduces the amount of data that needs to be transmitted. This improves performance, especially for websites with a large number of resources or mobile devices with limited bandwidth.

**Server Push**

HTTP/2 introduced server push, which allows the server to push resources to the client's cache before the client requests them. This can improve performance by reducing the number of roundtrips required to load a page and improving cache hit rates.

In HTTP/1.1, the server must wait for the client to request each resource before it can send it.

**Compatibility**

HTTP1.1 is supported by all web browsers and servers, while HTTP2 requires a more modern browser and server. However, most modern browsers and servers now support HTTP2, and its adoption is increasing rapidly.

**Binary Protocol**

HTTP/1.1 uses plain text for its protocol, which is human-readable but can be resource-intensive. In contrast, HTTP/2 uses a binary protocol, which is more efficient and easier for computers to process. This improves performance, especially for mobile devices and low-bandwidth connections.

**Conclusion**

HTTP/2 offers significant performance improvements over HTTP/1.1 due to its use of multiplexing, header compression, server push, compatibility, and binary protocol. However, the adoption of HTTP/2 requires both the client and server to support it. As a result, it may take some time before HTTP/2 becomes the dominant protocol used by the web.

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

In JavaScript, objects are one of the most fundamental data types, and they play a crucial role in programming. An object is a collection of key-value pairs, where the keys are strings and the values can be any data type, including another object.

The internal representation of an object in JavaScript consists of a hidden object header and a property map. The object header contains metadata about the object, such as its type, size, and class. The property map is a data structure that maps property keys to property descriptors.

An object in JavaScript is a collection of key-value pairs, where the keys are strings or symbols and the values can be any data type, including other objects. Here is an example of creating an object in JavaScript:

const person = {

name: 'John',

age: 30,

address: {

street: '123 Main St',

city: 'Anytown',

state: 'CA'

}

};

In the above example, we created an object named person with three properties: name, age, and address. The address property is another object that contains three properties of its own.

When an object is created in JavaScript, it is represented internally as a collection of properties. Each property has a key and a value, and the key is always a string or a symbol. The value can be any data type, including another object.

JavaScript objects also have a prototype chain, which allows objects to inherit properties from their parent objects. For example, we can create a new object that inherits properties from the person object by using the Object.create() method :

const employee = Object.create(person);

employee.jobTitle = 'Developer';

When a property is accessed or modified on an object in JavaScript, the runtime looks up the property in the object's internal property map. If the property is found, the associated value is returned. If the property is not found in the object, JavaScript looks up the property in the object's prototype chain until it reaches the top-level Object.prototype object.

const city = person.address.city;

In conclusion, objects are a fundamental data type in JavaScript, and their internal representation is based on a collection of properties and property descriptors. JavaScript uses a prototype chain to allow objects to inherit properties from their parent objects. Understanding how objects are represented internally can help you write more efficient and effective JavaScript code.