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

HTTP/1.1 and HTTP/2 are both versions of the Hypertext Transfer Protocol (HTTP), which is used for communication between web servers and clients. Here are some of the main differences between the two:

1. Multiplexing: One of the biggest differences between HTTP/1.1 and HTTP/2 is in the way that they handle multiple requests. In HTTP/1.1, only one request can be sent over a single TCP connection at a time. This means that if a web page has multiple resources (such as images or CSS files), the client must make multiple requests, and each request incurs the overhead of setting up a new TCP connection. In contrast, HTTP/2 supports multiplexing, which allows multiple requests to be sent over a single TCP connection simultaneously, reducing the overhead and improving performance.

2. Binary protocol: HTTP/2 uses a binary protocol, which is more efficient than the text-based protocol used by HTTP/1.1. The binary protocol uses a compressed header format, which reduces the size of each request and response, leading to faster transfer speeds.

3. Server push: HTTP/2 introduces the concept of server push, which allows the server to send resources to the client before they are requested. This can improve performance by reducing the number of round-trips required to load a web page.

4. Stream prioritization: In HTTP/2, requests and responses are split into smaller units called streams. HTTP/2 introduces the ability to prioritize these streams, allowing more important resources to be delivered more quickly.

5. Connection management: HTTP/2 uses a more efficient way of managing connections between clients and servers. In HTTP/1.1, each new request requires a new connection to be established, which can lead to increased latency. HTTP/2 uses a single connection that can be reused for multiple requests, reducing the overall number of connections and improving performance.

Overall, HTTP/2 is designed to improve performance and reduce latency by addressing some of the limitations of HTTP/1.1. However, not all web servers and clients support HTTP/2, so some applications may still use HTTP/1.1.

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

In JavaScript, objects are a fundamental data type and are used to represent collections of related data and functionality. Objects in JavaScript are created using the object literal syntax, which consists of a pair of curly braces enclosing one or more key-value pairs, separated by commas.

For example, here's an object that represents a person:

const person = {

name: "Alice",

age: 30,

address: {

street: "123 Main St",

city: "Anytown",

state: "CA",

zip: "12345"

},

sayHello: function() {

console.log(`Hello, my name is ${this.name}.`);

}

};

In this example, person is an object with several properties, including name, age, address, and sayHello. The address property is itself an object with its own properties, street, city, state, and zip. The sayHello property is a function that can be called on the object.

Internally, JavaScript objects are represented as a collection of key-value pairs, where the keys are strings or symbols, and the values can be any data type, including other objects, arrays, functions, and primitive values such as numbers and strings. JavaScript objects are implemented using a hash table data structure, which allows for efficient lookup and insertion of properties.

When a property is accessed or modified on an object, JavaScript performs a hash table lookup to find the corresponding value. If the property does not exist, it is added to the object's internal representation. If the property does exist, its value is updated.

In summary, objects in JavaScript are a versatile data type that can represent complex data structures and encapsulate behavior. They are implemented using a hash table data structure, which provides efficient property lookup and insertion.

1. Codekata practice

