**DAY 1 -TASK**

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

|  |  |
| --- | --- |
| HTTP1.1 | HTTP2 |
| It introduced in 1997. | It introduced in 2015. |
| It is first version of http(1.0,1.1). | It is an updated new version of http2. |
| It process one request at a time. | It process multiple requests at a time. |
| It is not possible | It has higher prioritization over control compared to HTTP 1.1 |
| It performs each connection to the TCP which leads to delay. | It allows to multiple requests or response to the each TCP connection. |
| It required for parallel requests leads to high latency at multiple connection. | It required for single requests leads to low latency at multipe connection. |
| It consumes more bandwidth. | It consumes low bandwidth. |
| It performs the action which leads to slow the page loading. | It eliminates head-of-line blocking problem leads to faster page loading. |
| It performs the protocol of plain text,so it took low efficient for parse and overhead. | It uses binary protocol so it took more efficient for parse and reducing overhead. |
| It sends header data with each request leads to redundant for multiple request to same server. | It easily compressed the header data which reducing the redundant for each request to the same server. |
| It does not provide the secure connection to the client. | It provide the highly secure connection to the client (HTTPS),S-SECURE. |
| It does not support the HTTP2. | It can support the client side compatible ,so they can communicate the HTTP1.1 through using HTTP2. |
| It may compress the HTTP message but it took slower process. | It also compress the HTTP message using the advanced compression pack like “HPACK”to speed the process. |
| Its an outdated version. | Its an current updated version. |

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

**Introduction**

JavaScript, as a versatile and widely-used programming language, relies heavily on objects for data storage and manipulation. Understanding how objects are internally represented and how they function is crucial for any JavaScript developer. In this blog post, we will delve into the internal representation of objects in JavaScript and explore the fundamentals of objects, properties, and prototypes.

**Objects: The Cornerstone of JavaScript**

In JavaScript, almost everything is an object, or it can behave like one. Objects are composite data types that allow you to group related data and functions into a single entity. They are constructed using two main components:

1. **Properties**: These are key-value pairs, where the key (also known as a property name) is a string (or Symbol in modern JavaScript), and the value can be any data type, including other objects or functions.
2. **Methods**: These are functions that are attached to objects as properties. Methods allow objects to perform actions or provide behavior related to the data they store.

**Internal Representation of Objects**

Under the hood, JavaScript engines use various mechanisms to represent objects efficiently. The primary data structure for this purpose is known as a **dictionary** or **hash map**. Here's a simplified overview of how objects are represented internally:

**Hash Map for Properties**

* When you create an object in JavaScript, the engine allocates memory for it and initializes an empty hash map (or similar data structure) to store its properties and methods.
* Each property (key-value pair) is stored in this hash map, and the key is used to look up and access the associated value quickly.
* This hash map is optimized for fast property access, making object property retrieval efficient.

**Hidden Classes and Inline Caching**

* JavaScript engines use the concept of "hidden classes" (also known as "shapes" or "shapes and inline caches") to optimize property access.
* When you add or remove properties from an object, it can change the object's hidden class. This might seem like a minor detail, but it has a significant impact on performance.
* Engines can use inline caching to speed up property access. If an object's hidden class is known, property access is much faster. However, if the hidden class changes, the engine must update the cache.

**Prototypes and Inheritance**

One of the essential features of JavaScript objects is their ability to inherit properties and methods from other objects. This is achieved through **prototypes**. Each object in JavaScript has a prototype, which is a reference to another object. When a property or method is accessed on an object, and it doesn't exist on the object itself, JavaScript looks up the prototype chain to find it.

Prototypes allow for efficient code reuse and a powerful mechanism for implementing inheritance in JavaScript. Understanding prototypes is vital for building complex applications and working with object-oriented patterns.

**Conclusion**

Objects are the heart and soul of JavaScript. Their internal representation as hash maps, along with concepts like hidden classes and prototypes, makes them a fundamental aspect of the language. When you create and manipulate objects in JavaScript, you're not just working with data structures; you're harnessing the power of a rich and flexible object model that underlies the entire language. By mastering these concepts, you can write more efficient and maintainable JavaScript code.