**Day-1**

**Task-1**

**Differences Between HTTP/1.1 and HTTP/2**

The Hypertext Transfer Protocol (HTTP) is the foundation of data communication on the World Wide Web. Since its inception, HTTP has undergone several revisions to improve performance and efficiency. The two most widely used versions today are HTTP/1.1 and HTTP/2. Let’s explore the key differences between these two protocols.

**1.Protocol Format**

**HTTP/1.1**: Uses a textual format for communication. Each request and response is sent as plain text, which can be easily read and debugged by humans.

**HTTP/2**: Introduces a binary framing layer, converting all messages into binary format. This change enhances parsing efficiency and reduces errors.

**2. Multiplexing**

**HTTP/1.1**: Handles one request per TCP connection. This can lead to head-of-line blocking, where a single slow request can delay all subsequent requests.

**HTTP/2**: Supports full multiplexing, allowing multiple requests and responses to be sent simultaneously over a single TCP connection. This significantly reduces latency and improves page load times.

**3. Header Compression**

**HTTP/1.1**: Sends headers as plain text with each request, leading to redundant data transmission and increased overhead.

**HTTP/2**: Utilizes HPACK compression to reduce the size of headers. [This compression minimizes the amount of redundant data sent, improving overall efficiency](https://www.geeksforgeeks.org/difference-between-http-2-and-http-1-1/).

**4. Server Push**

**HTTP/1.1**: Does not support server push. The server can only respond to client requests.

**HTTP/2**: Introduces server push, allowing the server to send resources to the client proactively. [This feature can improve performance by preloading resources that the client is likely to need](https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference).

**5. Connection Management**

**HTTP/1.1**: Relies on multiple TCP connections to handle concurrent requests, which can lead to increased latency and resource consumption.

[**HTTP/2**: Uses a single TCP connection for multiple requests and responses, reducing the overhead associated with establishing and maintaining multiple connections](https://www.geeksforgeeks.org/difference-between-http-2-and-http-1-1/)

**Conclusion:**

HTTP/2 brings several advancements over HTTP/1.1, including binary framing, multiplexing, header compression, and server push. These improvements lead to faster, more efficient web communication, enhancing the user experience. As the web continues to evolve, understanding these differences can help developers make informed decisions about optimizing their applications.

**Task-2**

**Object and its Internal Representation**

**Introduction:**

JavaScript is a versatile language that treats objects as a fundamental part of its structure. Objects in JavaScript are used to store collections of data and more complex entities. Let’s dive into what objects are and how they are internally represented in JavaScript.

**What Are Objects?**

In JavaScript, an object is a collection of key-value pairs. These keys can be strings or symbols, and the values can be of any type, including other objects. Objects are different from primitive data types (like numbers, strings, and boolean) because they can hold multiple values and complex data structures.

**Creating Objects**

There are several ways to create objects in JavaScript:

1.**Object Literals:**

Example:

let car = {

make: 'Toyota',

model: 'Corolla',

year: 2020

};

2. **Constructor Functions**:

Example:

function Car(make, model, year) {

this.make = make;

this.model = model;

this.year = year;

}

let myCar = new Car('Honda', 'Civic', 2019);

**3.Object.create Method**:

Example:

let animal = {

type: 'Invertebrate'

};

let spider = Object.create(animal);

spider.name = 'Tarantula';

**Internal Representation**

Internally, JavaScript engines represent objects using a structure called a “hidden class” or “shape”. This structure helps optimize property access and method calls.

1. **Property Storage:**
   * Properties of an object are stored in a contiguous block of memory. Each property has a key and a value.
   * The keys are usually stored in a hash table, allowing for efficient lookups.
2. **Prototype Chain:**
   * Every object in JavaScript has a prototype, which is another object from which it inherits properties and methods.
   * This prototype chain allows for property and method inheritance, enabling objects to share behavior.
3. **Hidden Classes:**
   * When an object is created, the JavaScript engine assigns it a hidden class. This hidden class is modified as properties are added or removed.
   * Hidden classes help the engine optimize property access by predicting the layout of objects.

**Example:**

let person = {

firstName: 'John',

lastName: 'Doe'

};

person.age = 30;

Initially, the person object has a hidden class with firstName and lastName properties.

When the age property is added, the hidden class is updated to include this new property.

**Conclusion:**

Understanding the internal representation of objects in JavaScript can help developers write more efficient and optimized code. Objects are a powerful feature of JavaScript, enabling the creation of complex data structures and inheritance through prototypes. By leveraging hidden classes and the prototype chain, JavaScript engines can optimize property access and method calls, making your applications run faster.