**Full stack development (MERN)**

**Day 01 Task**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Description** | **HTTP1.1** | **HTTP2** |
| 1 | ****Multiplexing**** | HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. | HTTP/2 is able to use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource. |
| 2 | ****Server push**** | Typically, a server only serves content to a client device if the client asks for it. | However, this approach is not always practical for modern webpages, which often involve several dozen separate resources that the client must request. HTTP/2 solves this problem by allowing a server to "push" content to a client before the client asks for it. The server also sends a message letting the client know what pushed content to expect – like if Bob had sent Alice a Table of Contents of his novel before sending the whole thing. |
| 3 | ****Header compression**** | Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. Small files load more quickly than large ones. To speed up web performance, both HTTP/1.1 and HTTP/2 compress HTTP messages to make them smaller. | However, HTTP/2 uses a more advanced compression method called HPACK that eliminates redundant information in HTTP header packets. This eliminates a few bytes from every HTTP packet. Given the volume of HTTP packets involved in loading even a single webpage, those bytes add up quickly, resulting in faster loading. |
| 4 | **Working method** | It works on the textual format. | It works on the binary protocol. |

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

In javascript. Objects , data types such as numbers, strings, and booleans are distinct from primitive datatypes . Unlike Primitive data types which can only contain one value, objects can contain multiple values in the form of key-value pairs. These keys can be either variables or functions and are referred to as properties and methods, respectively, within the object's context.

ObjectName.ObjectProperty = propertyValue;

The format for removing an object's property is as follows:

delete ObjectName.ObjectProperty;

One can use the following syntax to retrieve a property from an object:

objectName.property

//or

objectName["property”]

//or

objectName[expression]

In summary, we can define Java Script properties as the values that are linked to a JavaScript object .

**Object methods**

An object method is a function definition that is stored within an object property.

i.e.,

It can be assumed that there will be a mechanical operation involved in initiating the car.

function(){return ignition.on}

The actions of stopping, braking, and turning headlights on and off are similar. Therefore, a straightforward definition of Java Script Object methods is that they are actions that can be carried out on objects.

**Create JavaScript Object with Object Literal**

To generate a JavaScript Object , a quick and effortless method is through Object Literal . All that is required is to enclose the attributes and corresponding values within curly brackets, as illustrated below.

let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};

**Create JavaScript Object with Constructor**

A constructor, which is essentially a function, can be used with the new keyword to create instances of the same type, as demonstrated below.

function Vehicle(name, maker) {

this.name = name;

this.maker = maker;

}

let car1 = new Vehicle(’Fiesta’, 'Ford’);

let car2 = new Vehicle(’Santa Fe’, 'Hyundai’)

console.log(car1.name); //Output: Fiesta

console.log(car2.name); //Output: Santa Fe

**Using the JavaScript Keyword new**

This instance generates a fresh javascript object that consists of four attributes.

var person = new Object();

person.firstName = “John”;

person.lastName = “Doe”;

person.age = 50;

person.eyeColor = “blue”;

**Using the Object.create method**

The Object.create() method enables the creation of objects, providing the flexibility to select the prototype object for the new object, without the need to specify a constructor function.

// Animal properties and method encapsulation

var Animal = {

type: 'Invertebrates', // Default value of properties

displayType: function() { // Method which will display type of Animal

console.log(this.type);

}

};

// Create new animal type called animal1

var animal1 = Object.create(Animal);

animal1.displayType(); // Output:Invertebrates

// Create new animal type called Fishes

var fish = Object.create(Animal);

fish.type = 'Fishes';

fish.displayType();

// Output:Fishes