**Day 1 Task**

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

Answer:

**HTTP:**

* The Hypertext Transfer Protocol, or HTTP, is an application protocol that has been the de facto standard for communication on the World Wide Web since its invention in 1989.
* HTTP stands for hypertext transfer protocol, and it is the basis for almost all web applications. More specifically, HTTP is the method computers and servers use to request and send information.
* The first usable version of HTTP was created in 1997. Because it went through several stages of development, this first version of HTTP was called HTTP/1.1. This version is still in use on the web. In 2015, a new version of HTTP called HTTP/2 was created.
* HTTP/2 solves several problems that the creators of HTTP/1.1 did not anticipate. In particular, HTTP/2 is much faster and more efficient than HTTP/1.1.

**Differences between HTTP1.1 vs 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.
* In contrast, 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. HTTP/2 does this by splitting data into binary-code messages and numbering these messages so that the client knows which stream each binary message belongs to.

1. **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.

1. **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.
* 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.

1. **Prioritization:**

* Prioritization refers to the order in which pieces of content are loaded. Prioritization affects a webpage's load time.
* For example, certain resources, like large JavaScript files, may block the rest of the page from loading if they have to load first.
* More of the page can load at once if these render-blocking resources load last.
* In HTTP/2, developers have hands-on, detailed control over prioritization. This allows them to maximize perceived and actual page load speed to a degree that was not possible in HTTP/1.1. HTTP/2 offers a feature called weighted prioritization. This allows developers to decide which page resources will load first, every time.

|  |  |
| --- | --- |
| **HTTP/1.1** | **HTTP/2** |
| It works on the textual format. | It works on the binary protocol. |
| There is head of line blocking that blocks all the requests behind it until it doesn’t get its all resources. | It allows multiplexing so one TCP connection is required for multiple requests. |
| It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |

Write a blog about objects and its internal representation in Javascript

Answer:

Object in JavaScript:

Write a blog about objects and its internal representation in Javascript

Answer:

Object in JavaScript:

Write a blog about objects and its internal representation in Javascript

Answer:

Object in JavaScript:

Write a blog about objects and its internal representation in Javascript

Answer:

Object in JavaScript:

Write a blog about objects and its internal representation in Javascript

Answer:

Object in JavaScript:

Write a blog about objects and its internal representation in Javascript

Answer:

Object in JavaScript:

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

Answer:

**Objects in JavaScript:**

* Objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs.
* These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.
* Objects, in JavaScript, is most important data-type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive data-types(Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data-types all store a single value each (depending on their types).
* Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.
* The following code assigns a **simple value** (Mercedes) to a **variable** named car:
  + var car = "Mercedes";
* Objects are variables too. But objects can contain many values.
* The following code assigns **many values** (Mercedes, C-class, White and soo on) to a **variable** named Car:
  + var car = {Make: “Mercedes”, Model: “C-Class”, Color: “White”, Fuel: Diesel, Weight: “850kg”, Mileage: “8Kmpl”, Rating: 4.5};
* The values are written as **name:value** pairs (name and value separated by a colon).
* Syntax:
* var <object-name> = {key1: value1, key2: value2,... keyN: valueN};
* So, conclusion and definition for JS objects is “JavaScript objects are containers for named values”.

**Object Properties**

* The name:values pairs (in JavaScript objects) are called **properties**.
* var car = {Make: “Mercedes”, Model: “C-Class”, Color: “White”,
* Properties can usually be changed, added, and deleted, but some are read only.

**The syntax for adding a property to an object is :**

* ObjectName.ObjectProperty = propertyValue;

**The syntax for deleting a property from an object is:**

* delete ObjectName.ObjectProperty;

**The syntax to access a property from an object is:**

objectName.property        // Car.Make or objectName["property”]    // Car["Make"] or

objectName[expression]   // x = "Make"; Car[x]

* So, Conclusion and simple definition for Java Script properties is “Properties are the values associated with a JavaScript object”.

# **Creating Objects In JavaScript :**

# **Creating JavaScript Object with Object Literal**

* One of easiest way to create a JavaScript object is object literal, simply define the property and values inside curly braces as shown below
* let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};

# **Create JavaScript Object with Constructor**

* Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor as shown 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**

The following example also creates a new JavaScript object with four properties:

**Example:**

var person = new Object();  
person.firstName = “John”;  
person.lastName = “Doe”;  
person.age = 50;  
person.eyeColor = “blue”;

# **Using the Object.create method**

* Objects can also be created using the [Object.create()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/create) method. This method can be very useful, because it allows you to choose the prototype object for the object you want to create, without having to define 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