1. Write a blog on Difference between HTTP1.1 v/s HTTP2.

* HTTP1.1 uses a plain text format for all it’s requests and response whereas

HTTP2 uses binary framing layer which converts the message into semantics

while maintaining the HTTP semantics like headers,verbs and method. Thus,

ensuring that the web application created before HTTP2 can interact with it properly

continuing working normally.

* HTTP 1.1 uses 6 TCP connection between client and server whereas

HTTP 2 uses single secured TCP connection.

* In HTTP 1.1 keep-alive option was introduced to reuse the same TCP connection

for multiple HTTP requests whereas previously for each HTTP requests a new

TCP connection was required.

In HTTP2 streams are created which sends all requests as a separate stream frame in single

secured TCP connection pipeline thus making it faster.

* https is mandatory in HTTP2 thus it is compulsory to have TLS setup whereas HTTP1.1

doesn’t need TLS setup.

* HTTP1.1 request header data cannot be compressed whereas HTTP 2 uses HPACK which seperats header data from request data and can compress header data.
* HTTP1.1 with each request, header data is mandatory whereas HTTP 2 enables automatic

caching of header data if it is repeated thus with every request, header data is not required,

reducing the HTTP request size using HPACK.

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

* Objects are the building blocks of javascript and hence it’s most important data-type. It is the physical representation of data.
* Unlike primitive data-types, which stores single value, objects can contain any combination of primitive data-types as well as reference data-type. Thus, can be defined as the unordered collection, of primitive or reference types, which stores data in the form of “key-value pair”. These keys can be the variables or function and in the context of javascript are called as property and methods respectively.
* For Ex: let student = { name : “John”,

age : 27,

attendance : [“Mon”, ”Tue”, ”Wed”]

displayInfo : function() {

console.log($(student.name) is of age $(student.age)}

}

output : In the above example “name”, “age”, “attendance”, “displayInfo” are the keys and “John”, “27”, [“Mon”, ”Tue”, ”Wed”] and function() are the value of their respective keys.

* Properties of Object :
* The property names of an object can be string or number. If the name is in number it is accessed using “bracket notation”.

For ex: let student = { name : “John”,

age : 27,

10 : 100

displayInfo : function( ) {

console.log(The value of key 10 is ${student [‘10’]} ) }

student.displayInfo();

* Property names can also be strings with one or more space separated and must be enclosed in quotes. It is also accessed using bracket notation.

For Ex: let student = {“student name” : “John” }

* To access property and method of an object “dot operator” is used.

For Ex: To access name of object student, syntax must be : console.log(student.name)

* Object Creation : There are several syntax’s to create an object.
* Object literal syntax : The above example where object is created using “curly braces { }” is known as Object literal syntax where object and it’s property/method are initialized directly.

Syntax : var obj = {

property1/method1 = value1,

property2/method2 = value2

}

* Object Constructor : In this “new keyword” is used to create an object in conjunction with the “Object” constructor. It defines a set of properties and methods that would be common to all objects initialized using the constructor.
* For this method “class” must be created. This class can contain constructor, data member i.e property, methods.
* For ex : class Vehicle { }

var VehicleObj = new Vehicle();

VehicleObj.type = “car”;

VehicleObj.color = “red”;

VehicleObj.model = “Tesla”

console.log(VehicleObj.model);

This example is not an efficient method for programs that require multiple objects of same kind, as it would require repeatedly writing the above lines of code again and again.

* Thus, object creation using constructor, new keyword and function() will provide more efficient way of writing the object.

For ex : function Vehicle(type,model){

this.type = type;

this.model = model;

}

let obj1 = new Vehicle(“car”,”Tesla”);

let obj2 = new Vehicle(“motorcycle”, “Harley Davidson”);

console.log(obj1.type); //output: car

console.log(obj2.model); //output: Harley Davidson.

* This approach is used in ES5. In ES6 this constructor function Vehicle() can be converted to a class declaration.

For Ex: class Vehicle{

constructor(type, model){

this.type = type;

this.model = model;

}

}

let obj1 = new Vehicle(“car”,”Tesla”);

let obj2 = new Vehicle(“motorcycle”, “Harley Davidson”);

console.log(obj1.type); //output: car

console.log(obj2.model); //output: Harley Davidson.