**Question-1;**

Difference between HTTP1.1 vs HTTP2.

HTTP stands for hypertext transfer protocol.it is used to client server communication.by using http users sends the request to the server.

|  |  |
| --- | --- |
| HTTP 1.1 | HTTP2 |
| * HTTP 1.1  is creater from 1997 * It used work on the textual format * There is head of line  blocking that blocks all the request behind it until it doesn’t get all resources. * It used request resources inlining for use getting multiple pages. * It compresses data by itself. | * HTTP is creater from 2015 * Its work on the binary protocol * Its allow multiplexing so one TCP connection is request for multiple request. * It uses PUSh frame by server that collects all multiple pages. * It uses TPACK for data compression |

**Question-2;**

objects and its internal representation in Javascript.

**objects and its internal representation in Javascript**

Objects in java script,is it’s most important data type and forms the building blocks for modern java script.these objects are quite difference from java script’s primitive data type (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 my contain any combination of these primitive data types as reference data type.

An object ,is a reference data type. variable that are assigned a reference value are given  a reference or a pointed to that value.that reference or pointed points to the location in memory where the objects is stored.the variables don’t actually store value.

For E.g if you object is a student,it will have properties like name,age,address,id,etc and methods like updateAddress,updateNam,etc.

**Objects and properties;**

A JavaScript object has properties associated with it. A property of an object can be explained as a variable that is attached to the object. Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects. The properties of an object define the characteristics of the object. You access the properties of an object with a simple dot-notation:

**objectName.propertyName**

Properties of JavaScript objects can also be accessed or set using a bracket notation (for more details see [property accessors](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_Accessors)). Objects are sometimes called associative arrays, since each property is associated with a string value that can be used to access it. So, for example, you could access the properties of the myCar object as follows:

**myCar['make'] = 'Ford';**

**myCar['model'] = 'Mustang';**

**myCar['year'] = 1969;**

An object property name can be any valid JavaScript string, or anything that can be converted to a string, including the empty string. However, any property name that is not a valid JavaScript identifier (for example, a property name that has a space or a hyphen, or that starts with a number) can only be accessed using the square bracket notation. This notation is also very useful when property names are to be dynamically determined (when the property name is not determined until runtime). Examples are as follows:

**// four variables are created and assigned in a single go,**

**// separated by commas**

**var myObj = new Object(),**

**str = 'myString',**

**rand = Math.random(),**

**obj = new Object();**

**myObj.type              = 'Dot syntax';**

**myObj['date created']   = 'String with space';**

**myObj[str]              = 'String value';**

**myObj[rand]             = 'Random Number';**

**myObj[obj]              = 'Object';**

**myObj['']               = 'Even an empty string';console.log(myObj);**

You can also access properties by using a string value that is stored in a variable:

var propertyName = 'make';

**myCar[propertyName] = 'Ford';propertyName = 'model';**

**myCar[propertyName] = 'Mustang';**

You can use the bracket notation with [for...in](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in) to iterate over all the enumerable properties of an object. To illustrate how this works, the following function displays the properties of the object when you pass the object and the object's name as arguments to the function:

**function showProps(obj, objName) {**

**var result = ``;**

**for (var i in obj) {**

**// obj.hasOwnProperty() is used to filter out properties from the object's prototype chain**

**if (obj.hasOwnProperty(i)) {**

**result += `${objName}.${i} = ${obj[i]}\n`;**

**}**

**}**

**return result;**

**}**

So, the function call showProps(myCar, "myCar") would return the following:

**myCar.make = Ford**

**myCar.model = Mustang**

**myCar.year = 1969**

**Creating Objects In JavaScript :**

**Create 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**