**Question 1:**

[**HTTP**](https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/) stands for *hypertext transfer protocol*, and it is the basis for almost all web applications. HTTP is the method computers and servers use to request and send information. when your browser sends an HTTP request to a web server after entering an URL. The HTTP command then provides an HTTP response to the web server with the webpage’s contents.

The first usable version of HTTP was created in 1997. 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 is based on Google’s SPDY Protocol (originally designed to speed up the serving of web pages).

It is important to note that HTTP/2 is not a replacement for HTTP. It is merely an extension, with all the core concepts such as HTTP methods, Status Codes, URIs (*Universal Resource Identifier*), and Header Fields remaining the same.

**Main difference between Http/1.1 & Http/2**

**Textual vs. Binary**

* HTTP1.x uses text-based commands to complete HTTP requests. If you were to view one of these requests, they would be perfectly readable (to a system admin, at least).
* HTTP2, on the other hand, uses binary commands (1s and 0s) to complete HTTP requests. It needs to be converted back from binary to read the request. This conversion to Binary takes place in the Binary Framing Layer, so only binary commands are transmitted over the network.

**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 (*Transmission Control Protocol*) 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.

**Server push**

* In Http/1.1, a server only serves content to a client device if the client asks for it. However, this approach is not always practical for modern webpage’s, 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.

**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.

**Question 2:**

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, are its 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.

Syntax:

var <object-name> = {key1: value1, key2: value2,... keyN: valueN};

Example: var mobile={ Brand:’Samsung’, Model:’S20’, color:’blue’, rating:4.6};

**Object property:**

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

**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]

**Creating a JavaScript Object**

With JavaScript, you can define and create your own objects.

There are different ways to create new objects:

* Create a single object, using an object literal.

Example: const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

* Create a single object, with the keyword new.

Example: const person = new Object();  
 person.firstName = "John";  
 person.lastName = "Doe";  
 person.age = 50;  
 person.eyeColor = "blue";

* Define an object constructor, and then create objects of the constructed type.
* Create an object using Object.create().