1. **Difference between HTTP1.1 vs 2**

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.

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.
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. 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. **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.

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

"A JavaScript object is a collection of named values having state and behaviour (properties and method)". For example: Person, car, pen, bike, Personal Computer, Washing Machine etc. Take the case of cars. All cars have the same properties, but the property values differ from car to car. All cars have the same methods, but the methods are performed at different times.

Let's have an example of mercedes car and list out its properties(Features):

1. Make: Mercedes
2. Model: C-Class
3. Color: White
4. Fuel: Diesel
5. Weight: 850kg
6. Mileage: 8Kmpl
7. Rating: 4.5

Taking the above as reference, list out the objects, Object properties and Methods.

1. **Objects:** 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".

2)Object Properties: The name: values pairs (in JavaScript objects) are called **properties**.

var car = {Make: "Mercedes", Model: "C-Class", Color: "White", Fuel: Diesel, Weight: "850kg",Mileage: "8Kmpl", Rating: 4.5};

PROPERTY	PROPERTY VALUE
Make	Mercedes
Model	C-Class
Colour	White
Fuel	Diesel
Weight	850kg
Mileage	8kmpl
Rating	4.5

The object properties can be different primitive values, other objects and functions. 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, "Properties are the values associated with a JavaScript object".

3)Object Methods

An object method is an object property containing a function definition.i.e., Let's assume to start the car there will be a mechanical functionality.

```
function()
{
    return ignition.on
    }
```

and so similar is to stop/brake/headlights on & off, etc.

Finally, "Methods are actions that can be performed on objects.".