1) Difference between HTTP1.1 and HTTP2  
  
**HTTP1.1**  
Developed by Timothy Berners-Lee in 1989 as a communication standard for the World Wide Web, HTTP is a top-level application protocol that exchanges information between a client computer and a local or remote web server. In this process, a client sends a text-based request to a server by calling a method like GET or POST.   
**HTTP2**  
HTTP/2 began as the SPDY protocol, developed primarily at Google with the intention of reducing web page load latency by using techniques such as compression, multiplexing, and prioritization. Hypertext Transfer Protocol working group of the IETF (Internet Engineering Task Force) publication of HTTP/2 in May 2015. 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, and Header Fields remaining the same.

|  |  |
| --- | --- |
| **HTTP1.1** | **HTTP2** |
| It was no longer required for each connection to be terminated immediately after every request was served with a response; instead, with the keep-alive header, it was possible to have persistent connections. It allowed multiple requests/responses per TCP connection. | It introduces the concept of a server push where the server anticipates the resources that will be required by the client and pushes them prior to the client making requests. The client retains the authority to deny the server push; however, in most cases, this feature adds a lot of efficiency to the process. |
| The Upgrade header was used to indicate a preference from the client that made it possible to switch to a more preferred protocol if found appropriate by the server. | Introduces the concept of multiplexing that interleaves the requests and responses without head-of-line blocking and does so over a single TCP connection. |
| It supports connection reuse i.e. for every TCP connection there could be multiple requests and responses, and pipelining where the client can request several resources from the server at once. However, pipelining was hard to implement due to issues such as head-of-line blocking and was not a feasible solution. | Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. It is done using streams which can be prioritized, can have dependencies and individual flow control. It also provides a feature called server push that allows the server to send data that the client will need but has not yet requested. |
| Introduces a warning header field to carry additional information about the status of a message. Can define 24 status codes, error reporting is quicker and more efficient. | Underlying semantics of HTTP such as headers, status codes remains the same. |
| It is relatively secure since it uses digest authentication, NTLM authentication. | Security concerns from previous versions will continue to be seen in HTTP/2. However, it is better equipped to deal with them due to new TLS features like connection error of type Inadequate Security. |
| Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags. | HTTP/2 does not change much in terms of caching. With the server push feature if the client finds the resources are already present in the cache, it can cancel the pushed stream |
| HTTP/1.1 provides faster delivery of web pages and reduces web traffic as compared to HTTP/1.0. However, TCP starts slowly and with domain sharing connection reuse and pipelining, there is an increased risk of network congestion. | HTTP/2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays. |

  
  
  
  
  
  
  
  
  
2) Write a blog about objects and its internal representation in JavaScript  
  
  
1 Objects in JavaScript, it is most important data-type and forms the building blocks for modern JavaScript.   
2 These objects are quite different from JavaScript’s primitive data-types(Number, String, Boolean, null, undefined and symbol)   
3 Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.  
4 An object is a reference data type. Variables that are assigned a reference value are given a reference to that value.   
5 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.   
6 These keys can be variables and are called properties and methods, respectively, in the context of an object.  
  
EX1: objects create in single line  
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};  
  
EX2: An object definition can span multiple lines  
const person = {

firstName: "John",

lastName: "Doe",

age: 50,

eyeColor: "blue"

};  
  
**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  
  
EX:   
var myCar = new Object();

myCar.make = 'Ford';

myCar.model = 'Mustang';

myCar.year = 1969;  
  
  
**Object with Constructor**

Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same.  
  
EX:  
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  
  
  
  
EX: Inside classes, there can be special methods named constructor().

class people {

constructor()

{

this.name = "Adam";

}

}

let person1 = new people();

console.log(person1.name);  
  
// Output : Adam   
  
--------------------------------------------------------------------------------------------------------------------------------------