1. Difference between HTTP 1.1 vs HTTP2

HTTP 1.1:

* 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. In response, the server sends a resource like an HTML page back to the client.
* GET method is a request used to ask for data from the host server listed after Host:
* By responding to the request, the web server returns an HTML page to the requesting client, in addition to any images, stylesheets, or other resources called for in the HTML.
* The requests and responses will go back and forth between the server and client until the web browser has received all the resources necessary to render the contents of the HTML page on the screen.

Pipelining and Head-of-Line Blocking:

* The first response that a client receives on an HTTP GET request is often not the fully rendered page. Instead, it contains links to additional resources needed by the requested page. The client discovers that the full rendering of the page requires these additional resources from the server only after it downloads the page.
* HTTP/1.1 takes care of this problem by introducing persistent connections and pipelining. With persistent connections, HTTP/1.1 assumes that a TCP connection should be kept open unless directly told to close.
* This allows the client to send multiple requests along the same connection without waiting for a response to each, greatly improving the performance of HTTP/1.1 over HTTP/1.0.

HTTP/2:

* 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.
* From a technical point of view, one of the most significant features that distinguishes HTTP/1.1 and HTTP/2 is the binary framing layer, which can be thought of as a part of the application layer in the internet protocol stack.
* As opposed to HTTP/1.1, which keeps all requests and responses in plain text format, HTTP/2 uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers.
* An application-level API would still create messages in the conventional HTTP formats, but the underlying layer would then convert these messages into binary. This ensures that web applications created before HTTP/2 can continue functioning as normal when interacting with the new protocol.
* The conversion of messages into binary allows HTTP/2 to try new approaches to data delivery not available in HTTP/1.1, a contrast that is at the root of the practical differences between the two protocols.

Advantages of the Binary Framing Layer:

* In HTTP/2, the binary framing layer encodes requests/responses and cuts them up into smaller packets of information, greatly increasing the flexibility of data transfer.
* As opposed to HTTP/1.1, which must make use of multiple TCP connections to lessen the effect of HOL blocking, HTTP/2 establishes a single connection object between the two machines.
* A single TCP connection also improves the performance of the HTTPS protocol, since the client and server can reuse the same secured session for multiple requests/responses. Maintaining a single connection can greatly reduce the resources required for HTTPS performance.

| **HTTP/1.1** | **HTTP/2** |
| --- | --- |
| It uses works on the textual format. | It works on the binary protocol. |
| There is head of line blocking that blocks all the requests behind it until it doesn’t get its all resources. | It allows multiplexing so one TCP connection is required for multiple requests. |
| It uses requests resource In lining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |

* Although the multiplexing inherent in the binary framing layer solves certain issues of HTTP/1.1, multiple streams awaiting the same resource can still cause performance issues.

1. Write about objects and its internal representation in JavaScript

Objects:

* JavaScript is designed on a simple object-based paradigm. An object is a collection of properties, and a property is an association between a name (or key) and a value. A property's value can be a function, in which case the property is known as a method.
* In JavaScript, an object is a standalone entity, with properties and type. Compare it with a cup, for example. A cup is an object, with properties. A cup has a colour, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.

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***

* Like all JavaScript variables, both the object name (which could be a normal variable) and property name are case sensitive. You can define a property by assigning it a value.
* For example, let’s create an object named *myCar* and give it properties named *make, model*, and *year* as follows:

var myCar = new Object();

myCar.make = 'Ford';

myCar.model = 'Mustang';

myCar.year = 1969;

* Properties of JavaScript objects can also be accessed or set using a bracket notation. Objects are sometimes called associative arrays, since each property is associated with a string value that can be used to access it.
* For example.,

myCar['make'] = 'Ford';

myCar['model'] = 'Mustang';

myCar['year'] = 1969;