**Difference between HTTP1.1 and HTTP/2:**

What is HTTP?

**(HyperText Transfer Protocol)** The communications protocol used to connect to Web servers on the Internet or on a local network (intranet). The primary function of HTTP is to establish a connection with the server and send HTML pages back to the user's browser. It is also used to download data from the server either to the browser or to any requesting application that uses HTTP.

Addresses of websites begin with an http:// prefix; however, Web browsers typically default to the HTTP protocol. For example, typing www.yahoo.com is the same as typing http://www.yahoo.com. In fact, only yahoo.com has to be typed in. The browser adds the rest.1

HTTP 1.1:

HTTP/1.1, the first standardized version of HTTP, was introduced in 1997. It presented significant performance optimizations (over HTTP/0.9 and HTTP/1.0) and transformed the way requests and responses were exchanged between clients and servers.

HTTP 2:

In 2010 Google introduced an experimental protocol, SPDY, which supported multiplexing (multiple requests/responses sent and received asynchronously over a single TCP connection) but as it gained traction IETF’s HTTP Working Group came up with HTTP/2 in 2015, which is based on the SPDY protocol.

Difference between HTTP 1.1 and HTTP 2:

| **Differentiator** | **HTTP/1.1** | **HTTP/2** |
| --- | --- | --- |
| Features | 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. |
| Status Code | 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. |
| Authentication Mechanism | 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. |
| Caching | 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. |
| Web Traffic | 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 sharding (resources can be downloaded simultaneously by using multiple domains), 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. |

**Object and its Internal Representation in Javascript**

What is Object:

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.

**objectName.propertyName**

Like all JavaScript variables, both the object name (which could be a normal variable) and property name are case sensitive. We 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.**

**const 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. 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 an empty string. However, any property name that is not a valid JavaScript identifier cannot use dot notation. For example, a property name that has a space or a hyphen, that starts with a number, or that is held inside a variable can only be accessed using the square bracket notation. This notation is also very useful when property names are to be dynamically determined, i.e. not determinable until runtime.

JavaScript object property names (keys) can only be strings or Symbols — all keys in the square bracket notation are converted to strings unless they are Symbols. For example, in the above code, when the key anotherObj is added to the myObj, JavaScript will call the toString() method of anotherObj, and use the resulting string as the new key.

We can also access properties with a string value stored in a variable. The variable must be passed in bracket notation.

Properties of an Object:

There are three native ways to list/traverse object properties:

1. for...in loops.

This method traverses all of the enumerable string properties of an object as well as its prototype chain.

1. Object.keys(myObj).

This method returns an array with only the enumerable own string property names ("keys") in the object myObj, but not those in the prototype chain.

1. Object.getOwnPropertyNames(myObj).

This method returns an array containing all the own string property names in the object myObj, regardless of if they are enumerable or not.

There is no native way to list "hidden" properties (properties in the prototype chain which are not accessible through the object, because another property has the same name earlier in the prototype chain).