Write a blog on Difference between HTTP1.1 vs HTTP2

HTTP 1.1:

Http 1.1 version was introduced in 1997. The is the major version of HTTP network protocol used by the World Wide Web. Http is a top level application protocol that exchanges information between a client computer and a local and remote web server.  This version introduces many performance enhancements like caching mechanism, transfer encoding, keep alive connections, byte range requests, and request pipelining. This protocol worked for 15 years almost.  Http 1.1 provided support for chunk transfer that allowed streaming of content. Cache control to specify policies in both requests and responses. The upgrade header indicates a preference from the client to switch to a more preferred protocol. It provides faster delivery of web pages and reduces web traffic if you compare it to Http 1.0. However, there is an increased risk of network congestion.Some of the optimizations used is Http 1.1 version are sprinting, inlining, domain shrading, and concatenating.This protocol introduces a warning header field to carry additional information about the status of a message. It can define 24 status codes, error reporting is quicker and more efficient.Protocol Http 1.1 is much secured than Http 1.0 because it uses digest authentication and NTLM authentication.In this version, SSL or secure sockets layer is not required but recommended. Digest authentication is an improvement over Http 1.0 which is now being used in Http 1.1. Moreover, Https uses SSL/TLS for secure encrypted communication.  
  
HTTP 2:

Http 2 is the new version of HTTP 1.1. The protocol introduced back in February 2015 by the Internet Engineering Task Force (IETF) Http working group. The newish protocol transport data to drastically speed up the web and can help boost your [Search Engine Optimization](https://www.curvearro.com/in/locations/kolkata/). It uses new technologies to provide full multiplexing connections.Http 2 version brings the concept of multiplexing to alternate the requests and responses without head-of-line blocking and does so over a single TCP connection.Http 2 server also pushes and request proactively into client caches In lieu of waiting for a new request for each resource.High-level compatibility with Http 1.1 such as methods, URLs, status codes, and header fields.

The Http 2 version utilizes multiplexing and server pushes to effectively reduce the page load time by a greater margin along with being sensitive to network delays.This protocol version removes the need for unnecessary optimization hacks.It brings the fundamental semantics of HTTP like headers, and status code remains the same.The security concern in Http 2 version is also good and almost same as Http 1.1. Rather Http 2 is better equipped to deal with security threats because of the new features it brings. For example, new TLS feature like connection error of type inadequate security.In Http 2 protocol, security is not at all recommended. It is because the security is encrypted since all almost all clients demand traffic to be encrypted. It also has minimum standards and minimum key size for encryption.

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

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.

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. We can 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:

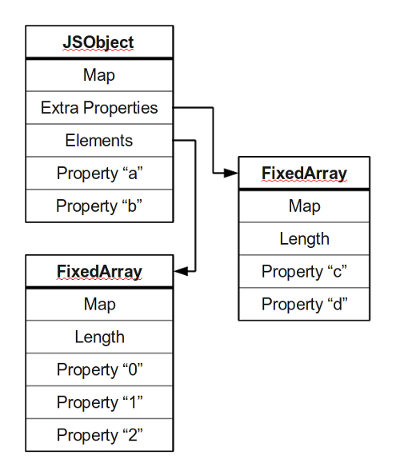
*var myCar = new Object();  
myCar.make = ‘Ford’;  
myCar.model = ‘Mustang’;  
myCar.year = 1969;*

The above example could also be written using an ****object initializer****, which is a comma-delimited list of zero or more pairs of property names and associated values of an object, enclosed in curly braces ({}):

*var myCar = {  
make: ‘Ford’,  
model: ‘Mustang’,  
year: 1969  
};*

## ****JavaScript’s internal representation of Objects:****

A simple diagram is probably the best way to give a quick overview of the object representation in Javascript.



Most objects contain all their properties in a single block of memory *(‘a’ and ‘b’)*. All blocks of memory have a pointer to a map, which describes their structure.

Named properties that don’t fit in an object are usually stored in an overflow array *(‘c’ and ‘d’)*.

Numbered properties are stored separately, usually in a contiguous array.

The JavaScript standard allows developers to define objects in a very flexible way, and it is hard to come up with an efficient representation that works for everything. An object is essentially a collection of *properties*: basically key-value pairs. We can access properties using two different kinds of expressions:

* obj.prop
* obj[“prop”]

According to the spec, property names are always strings. If we use a name that is not a string, it is implicitly converted to a string. This may be a little surprising: if we use a number as a property name, it gets converted to a string as well. So a JavaScript object is basically a map from strings to values.