1. **Write a blog on Difference between HTTP 1.1 vs HTTP 2**

**DIFFERENCE BETWEEN HTTP 1.1 vs HTTP 2**

|  |  |
| --- | --- |
| **HTTP 1.1** | **HTTP 2** |
| It 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 inlining 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. |
| It was released in 1997. | It was released in Feb, 2015 by IETF. |

**HTTP 1.1 has many features enhancements and performance optimizations**

* Persistent and pipelined connections.
* Virtual hosting
* Content negotiation, chunked transfer, compression and decompression, transfer encoding and catching directives.
* Character set and language tags
* Client identification and cookies.
* Basic authentication, digest authentication and secure HTTP.

**HTTP 2 Evolution of HTTP 2**

* Target a 50% reduction in page load time.
* Minimize deployment complexity.
* Avoid the need for any changes to content by website authors.
* Bring together like-minded parties interested in exploring protocols as a way of solving the latency problem.

1. **Write a blog about objects and its internal representation of JavaScript**

**OBJECT**

Object, in JavaScript, is it’s most important data-type and forms the building blocks for modern JavaScript.

These Objects are quite different from JavaScript’s primitive datatypes (Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data-type all store a single value each.

Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.

An object, is a reference data-type. Variables that are assigned a reference value are given a reference or a pointer to that value.

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.

These keys can be variables or functions and are called properties and methods, respectively in the context of an object.

**Eg:**

**Var car = { color:’red’, name:’benz’, model:1, add: ‘one’ }**

**INTERNAL REPRESENTAION IN JAVASCRIPT**

**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.

Objects properties are basically the same as ordinary JavaScript variables, except for the attachment to objects.

**ObjectName.propertyName**

It properties named make, model and year as follows:

Var myCar = new object();

myCar.make = ‘Ford’;

myCar.model = ‘Mustang’;

MyCar.year = 1969;

Unassigned properties of an object are undefined.

An object property name can be any valid JavaScript string, or anything that can be converted to a string, including the empty string.

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 using two different kinds of expressions:

* Obj.prop
* Obj[“prop”]

According to the spec, property names are always strings. If you use a name which is not a string, it is implicitly converted to a string.

This may be a little surprising: If you use a number as a property name, it gets converted to a string as well.

Because of this, you can store values at negative or fractional array indices.

So a JavaScript object is basically a map from strings to values.