2. Write a blog on Difference between HTTP1.1 vs HTTP2.

|  |  |  |
| --- | --- | --- |
| Differentiator | HTTP 1.1 | HTTP 2.0 |
| Year | 1997 | 2015 |
| Key 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. | Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. |
|  | pipelining was hard to implement due to issues such as head-of-line blocking and was not a feasible solution. | 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 | TCP starts slowly and with domain sharing (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. |
| Reliability | Slower but reliable transfers | Faster but non reliable transfers. |
| Applications | Email, Web browsing | Music Streaming, VoIP |
| Casting | Unicast | Unicast, Multicast, Broadcast |
| Acknowledgement | It acknowledges for each response | It doesn’t acknowledge |

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

* In JavaScript, Objects is the most important data-type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive data-types (Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data-types all store a single value each (depending on their types).
* 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.
* 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 = “Maruti Suzuki”;

mycar.model = “Omni”;

mycar.year = “2001”;

* you could access the properties of the myCar object as follows:

mycar['make'] = “Maruti Suzuki”;  
myCar['model'] = “Omni”;  
myCar['year'] = “2001”;

* An object property name can be any valid JavaScript string, or anything that can be converted to a string, including the empty string. However, any property name that is not a valid JavaScript identifier (for example, a property name that has a space or a hyphen, or that starts with a number) can only be accessed using the square bracket notation. This notation is also very useful when property names are to be dynamically determined (when the property name is not determined until runtime).

Create JavaScript Object with Object Literal:

* One of easiest way to create a javascript object is object literal, simply define the property and values inside curly braces as shown below

let bike = {name: 'Splendor', maker:'Hero', engine:'100cc'};

# Create JavaScript Object with Constructor

* Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor as shown below

function Vehicle(name, maker) {  
this.name = name;  
this.maker = maker;  
}  
let car1 = new Vehicle(’Figo’, 'Ford’);  
let car2 = new Vehicle(’Creta’, 'Hyundai’)  
console.log(car1.name); //Output: Figo  
console.log(car2.name); //Output: Creta