Blog on difference between **HTTP - 1.1** and **HTTP – 2**

HTTP (Hyper Text Transfer Protocol) is the means of communication between Client and Server. Client sends a request to server and the server returns back a response. Two significant version of protocol were HTTP 1.1 and HTTP 2, the earlier one was a standard protocol for a long time then the later came with significant changes to performance efficiency, security and other advantages. Few of the main factors which differ these 2 protocols are discussed below.

* Multiplexing

HTTP/1.1: Each request and response is handled sequentially over a single TCP connection. This means that if a browser wants to load multiple resources (like images, scripts, and stylesheets) from a server, it must wait for each request to complete before sending the next one.

* + - HTTP/2: One of the significant improvements in HTTP/2 is the introduction of multiplexing. Multiple requests and responses can be sent and received in parallel over a single connection. This eliminates the need for multiple TCP connections, reduces latency, and improves overall performance, especially for complex web pages with numerous resources.
* Header Compression
  + Headers in HTTP requests and responses can be verbose, containing redundant information in each request/response cycle. This can lead to increased overhead, especially for small payloads.
  + HTTP/2 features header compression, where headers are efficiently encoded using a technique called HPACK. This reduces overhead and improves network utilization, resulting in faster loading times, especially for repeated requests to the same server.
* Server Push
  + In the traditional model of HTTP/1.1, a server responds to client requests. If the server wants to send additional resources that the client hasn't requested yet (like associated CSS or JavaScript files), it needs to wait for the client to request them.
  + HTTP/2 introduces server push, a mechanism where the server can proactively send resources to the client before they are explicitly requested. This can significantly reduce the number of round trips and speed up page loading times, especially for complex web applications.
* Binary Protocol
  + HTTP/1.1 primarily uses textual representations for headers and payloads, which are human-readable but can be less efficient for machine processing.
  + HTTP/2 is a binary protocol, meaning that data is encoded in binary format for transmission. While this format is not human-readable, it is more efficient for parsing and processing by computers and networks, contributing to better performance.
* TLS – Security
  + TLS (Transport Layer Security) encryption is optional in HTTP/1.1, although it is strongly recommended for security reasons.
  + HTTP/2 requires the use of TLS encryption (HTTPS) for all connections. This enforces a higher level of security and privacy for data transmitted between clients and servers, protecting against various security threats and vulnerabilities.

In conclusion, while HTTP/1.1 served as a robust foundation for the web, HTTP/2 represents a significant leap forward in terms of performance, efficiency, and security. The introduction of features like multiplexing, header compression, server push, and a binary protocol has revolutionized how web content is delivered and experienced. As web technologies continue to evolve, understanding these underlying protocols becomes essential for developers, administrators, and users alike to ensure a fast, secure, and seamless web experience.

Blog about object and its internal representation in JS

Objects in Javascript are reference data types unlike other primitive data types which store single values. Objects are used to store keyed collections of various data. It stores data in the form of Key value pairs. Objects can be created in multiple ways,

let user = new Object();

let user = {};

let user = {

name: “John”, // key – “name”, value – “John”

age: 25,

“is eligible”: true // keys have multiword should be covered in quotes

};

The keys are always of type string and the value can be of any type. There are two ways to access the object properties(keys),

* Dot operator
* Square Notation

Dot operator can be used to access all property names which are single words, while square notation is used to access property names having multiword.

Console.log(user.name); // John

Console.log(user.is eligible); // gives syntax error

Console.log(user[“is eligible”]) // returns true

We can use “in” operator to access any property, if that property is available it returns the corresponding value. If the property is not available it return undefined instead of an error.

Console.log(user.place) // returns undefined

Console.log(“place” in user) // returns false

Console.log(“name” in user) // returns true // the key should be quoted

To loop over all the keys in an object, there is a special loop called **for..i**,

for(key in object) {

}

for(let key in user) {

console.log(key); // name, age, is eligible

console.log(user[key]); // John, 25, true

}