Difference Between HTTP/1.1 and HTTP/2

The Hypertext Transfer Protocol (HTTP) has been the foundation of data communication on the web. Over time, the protocol has evolved to improve performance and address the growing needs of the internet. HTTP/1.1, introduced in 1999, brought significant improvements over its predecessor, HTTP/1.0. However, with the advent of more complex web applications, HTTP/2 was introduced in 2015 to further enhance the performance and efficiency of web communications.

Key Differences:

1. Multiplexing:

HTTP/1.1\*: In HTTP/1.1, each request/response pair is sent over a separate TCP connection, leading to a phenomenon known as "headofline blocking," where a single slow response can block all subsequent requests.

HTTP/2: HTTP/2 allows multiple request/response pairs to be sent simultaneously over a single TCP connection, significantly reducing latency and improving page load times.

2. Header Compression:

HTTP/1.1: Headers are sent as plain text, which can lead to large overheads, especially with repetitive information like cookies.

HTTP/2: HTTP/2 uses HPACK header compression, which reduces the size of headers, thereby saving bandwidth and reducing latency.

3. Binary Protocol:

HTTP/1.1: HTTP/1.1 uses a textual protocol, which is easy for humans to read and debug but less efficient for machines to parse.

HTTP/2: HTTP/2 is a binary protocol, which is more compact and easier for computers to parse, leading to faster processing and fewer errors.

4. Server Push:

HTTP/1.1: HTTP/1.1 does not support server push, meaning that the server cannot send resources to the client until the client explicitly requests them.

HTTP/2: HTTP/2 supports server push, allowing servers to send resources to the client proactively, further reducing load times.

5. Prioritization:

HTTP/1.1: HTTP/1.1 lacks builtin prioritization, meaning all requests are treated equally, which can lead to inefficiencies.

HTTP/2: HTTP/2 allows clients to prioritize requests, ensuring critical resources are loaded first, improving the user experience.

6. Connection Management:

HTTP/1.1: Each new request often requires a new TCP connection, leading to higher overhead and slower performance.

HTTP/2: HTTP/2 uses a single, longlived connection for multiple requests, reducing overhead and improving performance.

Objects and Their Internal Representation in JavaScript

JavaScript, a versatile and widelyused programming language, is built around objects. Understanding objects and their internal representation is crucial for writing efficient and effective code.

Objects in JavaScript:

In JavaScript, an object is a collection of keyvalue pairs where the keys are strings (or Symbols) and the values can be any data type, including other objects. Objects are dynamic, allowing for properties to be added, modified, or deleted at runtime.

Internal Representation:

1. Properties and Attributes:

Data Properties: These are the most common types of properties, consisting of a key and a value. They have attributes such as configurable, enumerable, writable, and value.

Accessor Properties: These properties are functions (getters and setters) that provide indirect access to a property’s value.

2. Prototype Chain:

Every JavaScript object has a prototype, an object from which it inherits properties and methods. This forms a prototype chain, enabling property and method sharing among objects.

3. Hidden Classes:

JavaScript engines, such as V8 (used in Chrome and Node.js), use hidden classes (or shapes) to optimize property access. When an object is created, a hidden class is also created, defining the layout of the object’s properties. Modifying the object (e.g., adding or deleting properties) can change its hidden class, potentially impacting performance.

4. Property Access:

JavaScript engines optimize property access using techniques like inline caching. Initially, accessing a property involves looking it up in the hidden class. Subsequent accesses are optimized by caching the property’s location.

5. Garbage Collection:

JavaScript manages memory automatically through garbage collection. Objects that are no longer referenced are periodically cleaned up by the garbage collector, freeing memory for other uses.

Understanding IP Address, Port, HTTP Methods, MAC Address

IP Address:

An IP (Internet Protocol) address is a unique numerical identifier assigned to each device connected to a network. It allows devices to communicate with each other over the internet. IP addresses are of two types: IPv4 (e.g., 192.168.1.1) and IPv6 (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).

Port:

A port is a logical endpoint for communication in a network. Ports are used to identify specific processes or services running on a device. Common port numbers include 80 for HTTP, 443 for HTTPS, and 21 for FTP.

HTTP Methods:

HTTP methods are used to define the actions to be performed on resources in a web application. The most common HTTP methods are:

GET: Retrieve data from the server.

POST: Submit data to the server.

PUT: Update existing data on the server.

DELETE: Remove data from the server.

PATCH: Partially update existing data on the server.

MAC Address:

A MAC (Media Access Control) address is a unique identifier assigned to a network interface card (NIC) of a device. It is used for communication within a local network. MAC addresses are typically represented as six pairs of hexadecimal digits (e.g., 00:1A:2B:3C:4D:5E).

By understanding these concepts, developers can better grasp the intricacies of network communication and build more robust and efficient web applications.