**Title: Exploring the Differences Between HTTP/1.1 and HTTP/2**

**1. Request/Response Multiplexing:**

One of the key enhancements in HTTP/2 is the introduction of request/response multiplexing. In HTTP/1.1, only one request-response transaction could take place at a time on a given connection, which often led to performance bottlenecks. HTTP/2, on the other hand, allows multiple requests and responses to be exchanged concurrently over a single connection. This multiplexing feature significantly improves efficiency by reducing latency and optimizing resource utilization.

**2. Binary Protocol:**

While HTTP/1.1 used a plain-text format for its messages, HTTP/2 employs a binary protocol. By utilizing a binary format, HTTP/2 reduces the overhead of parsing and decreases the size of each individual message. The binary framing layer provides efficient and optimized data transmission, resulting in faster and more streamlined communication between clients and servers.

**3. Header Compression:**

HTTP/2 incorporates a new header compression mechanism called HPACK. In contrast, HTTP/1.1 sends headers with each request and response, resulting in redundant data being transmitted over the network. With HPACK, header fields are efficiently compressed and sent as a series of indexed references, significantly reducing the amount of data transmitted and improving overall performance.

**4. Server Push:**

Another notable feature introduced in HTTP/2 is server push. With server push, the server can proactively send resources to the client before they are explicitly requested. This capability allows the server to anticipate the client's needs and deliver essential resources preemptively, reducing the number of round trips required and improving page load times. In HTTP/1.1, such resources would have to be requested individually, incurring additional latency.

**5. Stream Prioritization:**

HTTP/2 introduces the concept of stream prioritization, enabling more efficient resource allocation. Streams can be assigned a priority level, allowing the server to determine which resources should be transmitted first. This prioritization ensures that critical resources are delivered promptly, enhancing the user experience and optimizing network utilization.

**6. Backward Compatibility:**

One of the key advantages of HTTP/2 is its backward compatibility with HTTP/1.1. HTTP/2 can be deployed on the same ports as HTTP/1.1, and if a client or server doesn't support HTTP/2, the communication falls back to the HTTP/1.1 protocol. This compatibility enables a smooth transition and widespread adoption of HTTP/2 without disrupting existing systems

**Title: Exploring Objects and Their Internal Representation in JavaScript**

**The Basics of Objects in JavaScript:**

In JavaScript, objects are dynamic, mutable collections of key-value pairs, where the keys are strings or symbols, and the values can be of any data type, including other objects. Objects are created using either the object literal syntax or the `new Object()` constructor.

**1. Properties:**

Properties are the key-value pairs that make up an object. Each property has a name (also called a key) and a corresponding value. Properties in JavaScript objects can be accessed and manipulated using dot notation (`object.property`) or bracket notation (`object['property']`).

Properties can be categorized into two types: data properties and accessor properties.

- Data Properties: Data properties contain a value, which can be of any JavaScript type, such as a number, string, or even another object. Data properties also have additional attributes like `writable`, `enumerable`, and `configurable`, which control their behavior.

- Accessor Properties: Accessor properties do not contain a value directly but instead define getter and setter functions to retrieve or modify a value. These functions are invoked automatically when accessing or assigning a value to the property.

**2. Prototype Chain**

The prototype chain is a mechanism that allows objects to inherit properties and methods from other objects. Every object in JavaScript has an internal link to another object called its prototype. When accessing a property or method on an object, JavaScript first checks if the object itself has that property. If not, it follows the prototype chain to look for the property in its prototype object. This process continues until the property is found or until the end of the chain is reached (i.e., the prototype is null).

By utilizing the prototype chain, JavaScript implements inheritance and promotes code reuse. Objects can inherit properties and methods from their prototypes, allowing for hierarchical relationships and shared behavior among objects.

Object Representation: Objects as Hash Tables

Under the hood, JavaScript objects are often implemented as hash tables or dictionaries. Hash tables provide efficient lookup and retrieval of properties based on their keys. When a property is accessed or modified, JavaScript internally performs a hash function on the property name to determine its memory location within the object's hash table. This enables quick access and manipulation of properties, even for large objects.

The hash table implementation ensures that property access and modification have an average time complexity of O(1), making objects in JavaScript highly performant for most use cases.

**Certainly! Here's a brief overview of IP address, port, HTTP methods, and MAC address:**

**1. IP Address:**

An IP (Internet Protocol) address is a numerical label assigned to each device connected to a computer network. It serves as an identifier for both the source and destination of data packets transmitted over an IP-based network, such as the internet. IP addresses can be either IPv4 (32-bit) or IPv6 (128-bit) format. IPv4 addresses are typically written as four sets of numbers separated by periods (e.g., 192.168.0.1), while IPv6 addresses are represented in a more complex hexadecimal format.

**2. Port:**

In networking, a port is a communication endpoint in an operating system that allows multiple applications or services to utilize network resources simultaneously. Ports are identified by numerical values ranging from 0 to 65535. They help in distinguishing between different network services running on a single IP address. Ports are categorized into three ranges: well-known ports (0-1023), registered ports (1024-49151), and dynamic or private ports (49152-65535).

**3. HTTP Methods:**

HTTP (Hypertext Transfer Protocol) is a protocol used for communication between web browsers (clients) and web servers. HTTP defines several methods (also known as verbs) that specify the type of action to be performed on a resource. The common HTTP methods include:

- GET: Retrieves a representation of a resource from the server.

- POST: Sends data to the server to create a new resource.

- PUT: Updates an existing resource on the server.

- DELETE: Deletes a resource on the server.

- PATCH: Partially updates an existing resource.

- HEAD: Retrieves only the headers of a resource, without the body.

- OPTIONS: Retrieves the available communication options for a resource.

HTTP methods enable clients to interact with web servers in a standardized manner, facilitating various operations on web resources.

**4. MAC Address:**

A MAC (Media Access Control) address is a unique identifier assigned to network interfaces at the hardware level. It is assigned by the manufacturer and embedded in the network interface card (NIC) of a device. MAC addresses are typically 48 bits long and represented as a series of six groups of two hexadecimal digits separated by colons or hyphens. MAC addresses are used for local network communication, allowing devices to identify and communicate with each other within a Local Area Network (LAN).

MAC addresses are primarily associated with the Data Link Layer of the OSI (Open Systems Interconnection) model and provide a means to deliver data packets within a network based on the unique identifier assigned to each device.