Difference Between HTTP/1.1 vs HTTP/2

HTTP/1.1

Key Features:

1. **Persistent Connections**: By default, HTTP/1.1 uses persistent connections, meaning that the TCP connection remains open for multiple requests/responses, reducing the latency associated with opening new connections.
2. **Pipelining**: Allows multiple requests to be sent out without waiting for the corresponding responses. However, this feature was not widely adopted due to problems like head-of-line blocking.
3. **Chunked Transfer Encoding**: Supports streaming of data, allowing a server to start sending a response before knowing its total length.
4. **Additional Headers**: Introduced several new headers and features like `Host`, which allows multiple domains to be served from a single IP address.

HTTP/2

Key Features:

1. **Binary Protocol**: Unlike HTTP/1.1, which is text-based, HTTP/2 is binary. This change enhances performance and reduces errors.
2. **Multiplexing**: Allows multiple requests and responses to be sent simultaneously over a single TCP connection. This eliminates the head-of-line blocking problem seen in HTTP/1.1.
3. **Header Compression**: Uses HPACK compression to reduce the overhead of HTTP headers, making web pages load faster.
4. **Server Push**: Enables servers to send resources to the client proactively, reducing the need for additional round-trip times.
5. **Stream Prioritization**: Clients can specify the priority of requests, allowing critical resources to load faster.

Comparison

Performance:

- **HTTP/1.1**: Suffers from head-of-line blocking and higher latency due to sequential request processing.
- **HTTP/2**: Significantly improves performance with multiplexing and header compression.

Resource Management:

- **HTTP/1.1**: Limited by the number of concurrent connections browsers can open to a single domain (typically six).
- **HTTP/2**: Efficiently uses a single connection for multiple streams, reducing the need for connection management.

Security:

- **HTTP/1.1**: Can use TLS, but it's not mandatory.
- **HTTP/2**: Often deployed with HTTPS, promoting better security practices.

Conclusion

HTTP/2 addresses many limitations of HTTP/1.1, providing better performance, lower latency, and improved resource utilization. As the web continues to grow, HTTP/2 is a significant step forward, offering a more efficient and robust protocol for modern web applications.

2. Understanding Objects and Their Internal Representation in JavaScript

Introduction

In JavaScript, objects are foundational elements that enable developers to manage and manipulate data effectively. They offer a versatile way to structure and store various data types. This blog delves into the essence of JavaScript objects, their creation, manipulation, and the internal mechanisms that optimize their performance.

What is an Object?

An object in JavaScript is a collection of key-value pairs, where each key is a string (or Symbol) and each value can be any data type, including other objects. Objects are used to model real-world entities, encapsulate related data, and define configurations.

Creating Objects

There are multiple ways to create objects in JavaScript:

Object Literals:
javascript
Copy code

```
let person = {

  name: 'John',

  age: 30,

  greet: function() {

    console.log('Hello, ' + this.name);

  }

};
```

1.

Using the `new Object()` Syntax:
javascript

Copy code

```
let person = new Object();

person.name = 'John';

person.age = 30;

person.greet = function() {

  console.log('Hello, ' + this.name);

};
```

2.

Constructor Functions:
javascript
Copy code

```
function Person(name, age) {

  this.name = name;

  this.age = age;

  this.greet = function() {

    console.log('Hello, ' + this.name);

  };

}

let person = new Person('John', 30);
```

3.

ES6 Classes:
javascript
Copy code

```
class Person {

  constructor(name, age) {

    this.name = name;
```

```
    this.age = age;
  }

  greet() {
    console.log('Hello, ' + this.name);
  }
}

let person = new Person('John', 30);
```

4.

Internal Representation

JavaScript engines employ various techniques to optimize objects. Here's a simplified overview:

Property Storage

1. **Dictionary Mode**: Initially, objects are stored as dictionaries (hash maps). This allows for flexible and fast property access and addition.
2. **Shapes and Hidden Classes**: Modern JavaScript engines like V8 use "hidden classes" or "shapes" to optimize property access. When an object is created, a hidden class is created to describe its structure. As properties are added, the hidden class evolves, allowing for fast property access based on the object's shape.

Property Access

1. **Inline Caches**: To speed up property access, JavaScript engines use inline caches. These caches store information about the location of properties, reducing the time needed to look up properties on subsequent accesses.
2. **Transition Chains**: Objects of the same shape can share transition chains, allowing the engine to quickly determine the position of properties.

Garbage Collection

JavaScript engines use garbage collectors to manage memory. Objects that are no longer referenced are marked for garbage collection, freeing up memory for other uses. This process involves:

1. **Mark-and-Sweep**: The garbage collector marks live objects and sweeps away the unmarked, freeing their memory.
2. **Generational Collection**: Objects are categorized into generations, optimizing the collection process by focusing on younger objects that are more likely to be garbage.

Prototypes and Inheritance

JavaScript uses prototypal inheritance, where each object has a prototype (another object) from which it can inherit properties and methods.

1. **Prototype Chain**: When accessing a property, JavaScript first looks at the object itself. If the property is not found, it looks at the object's prototype, and so on up the prototype chain.

`__proto__` Property: This property points to an object's prototype.
javascript
Copy code

```
let person = { name: 'John' };

let employee = { __proto__: person, job: 'Developer' };

console.log(employee.name); // John
```

2.

Dynamic Nature of Objects

JavaScript objects are dynamic, meaning properties can be added, modified, or removed at runtime. This flexibility allows for powerful and adaptable code but can also introduce performance considerations. For instance, adding properties dynamically can trigger the creation of new hidden classes, which might affect performance.

Understanding IP Addresses, Ports, HTTP Methods, and MAC Addresses

IP Address

What is an IP Address?

An IP (Internet Protocol) address is a unique identifier assigned to each device connected to a network. It serves two primary functions: identifying the host or network interface and providing the location of the host in the network. IP addresses are essential for the routing of information across networks.

Types of IP Addresses

1. **IPv4**:
    - **Format**: Consists of four decimal numbers separated by periods (e.g., 192.168.0.1).
    - **Range**: 32-bit address space, allowing for about 4.3 billion unique addresses.
    - **Example**: 192.168.1.1
2. **IPv6**:
    - **Format**: Consists of eight groups of four hexadecimal digits separated by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
    - **Range**: 128-bit address space, allowing for a virtually unlimited number of unique addresses.
    - **Example**: 2001:0db8:85a3:0000:0000:8a2e:0370:7334

Public vs. Private IP Addresses

- **Public IP Address**: Used to identify devices on the wider internet. These addresses are assigned by ISPs (Internet Service Providers).
- **Private IP Address**: Used within a private network. These addresses are not routable on the internet and are used for local device communication. Common ranges include:
    - 192.168.0.0 - 192.168.255.255
    - 10.0.0.0 - 10.255.255.255
    - 172.16.0.0 - 172.31.255.255

Port

What is a Port?

A port is a logical endpoint in a network connection. It is used to differentiate between multiple services or applications running on a single device. Ports allow a single IP address to be used for multiple connections simultaneously.

Port Numbers

- **Range**: Port numbers range from 0 to 65535.
- **Categories**:
    - **Well-Known Ports**: 0-1023, assigned to common protocols and services (e.g., HTTP on port 80, HTTPS on port 443).
    - **Registered Ports**: 1024-49151, assigned to specific services by the IANA (Internet Assigned Numbers Authority).
    - **Dynamic/Private Ports**: 49152-65535, used for temporary or private connections.

HTTP Methods

What are HTTP Methods?

HTTP methods are used to perform different actions on resources identified by URLs (Uniform Resource Locators) in a web application. Each method specifies the type of request being made to the server.

Common HTTP Methods

1. **GET**:
    - **Purpose**: Retrieve data from the server.
    - **Idempotent**: Yes (multiple identical requests yield the same result).
    - **Safe**: Yes (no side effects on the server).
    - **Example**: Fetching a webpage.
2. **POST**:
    - **Purpose**: Submit data to the server, often resulting in a change in state or side effects on the server.
    - **Idempotent**: No (multiple identical requests may have different effects).
    - **Safe**: No.
    - **Example**: Submitting a form.
3. **PUT**:
    - **Purpose**: Replace a resource or create a new one if it does not exist.
    - **Idempotent**: Yes.
    - **Safe**: No.
    - **Example**: Updating a user's profile.
4. **DELETE**:
    - **Purpose**: Remove a resource from the server.
    - **Idempotent**: Yes.
    - **Safe**: No.
    - **Example**: Deleting a user account.
5. **PATCH**:
    - **Purpose**: Apply partial modifications to a resource.
    - **Idempotent**: Yes.

    - **Safe**: No.
    - **Example**: Updating a field in a user profile.

MAC Address

What is a MAC Address?

A MAC (Media Access Control) address is a unique identifier assigned to the network interface of a device. It operates at the data link layer (Layer 2) of the OSI (Open Systems Interconnection) model and is used for network communication within a local network segment.

Format and Structure

- **Format**: Typically represented as six groups of two hexadecimal digits separated by colons or hyphens (e.g., 00:1A:2B:3C:4D:5E).
- **Structure**: The first three groups (24 bits) represent the Organizationally Unique Identifier (OUI), which identifies the manufacturer. The remaining three groups (24 bits) are assigned by the manufacturer and are unique to each device.

Use Cases

- **Local Network Communication**: Ensures that data packets are delivered to the correct hardware within the same local network.
- **ARP (Address Resolution Protocol)**: Translates IP addresses into MAC addresses to facilitate communication on a local network