Name: Silambarasan V

Task: Java script: Introduction to Browser & web

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

* HTTP, or Hypertext Transfer Protocol, serves as the foundation for communication on the World Wide Web. It facilitates the transfer of various resources, such as HTML documents, images, and scripts, between a web server and a web browser.
* Two of the most prominent protocols, HTTP/1.1 and HTTP/2, have distinct features and capabilities that impact how data is transmitted between clients and servers.

|  |  |  |
| --- | --- | --- |
| **Feature** | **HTTP/1.1** | **HTTP/2** |
| **Multiplexing** | No multiplexing, leading to head-of-line blocking | Multiplexing allows multiple requests and responses to be sent and received in parallel over a single connection, reducing latency and improving efficiency |
| **Header Compression** | Headers are not compressed, leading to redundant data transmission | Headers are compressed, reducing overhead and improving performance |
| **Binary Protocol** | Text-based protocol, requiring parsing for both client and server | Binary protocol, allowing for more efficient parsing and reduced overhead |
| **Server Push** | Not supported | Server push enables servers to send additional resources to the client before they are requested, improving page load times |
| **Prioritization** | No built-in prioritization mechanism | Requests can be prioritized, allowing critical resources to be fetched first, enhancing user experience |
| **TLS Requirement** | Optional | Required for HTTP/2, providing security by default |

**Multiplexing:**

HTTP/2 introduces multiplexing, allowing multiple requests and responses to be sent and received in parallel over a single connection. This feature significantly reduces latency and improves the efficiency of data transfer, overcoming the head-of-line blocking issue present in HTTP/1.1.

**Header Compression:**

Unlike HTTP/1.1, where headers are sent as plain text, HTTP/2 utilizes header compression, reducing redundant data transmission and improving performance by minimizing overhead.

**Binary Protocol:**

HTTP/1.1 is a text-based protocol, requiring parsing for both client and server. In contrast, HTTP/2 is a binary protocol, which allows for more efficient parsing and reduced overhead.

**Server Push:**

HTTP/2 introduces server push, a feature that enables servers to push additional resources to the client before they are requested. This capability can significantly improve page load times by reducing the number of round trips required to fetch resources.

**Prioritization:**

HTTP/2 includes a built-in prioritization mechanism, allowing clients to specify the priority of requests. This feature ensures that critical resources are fetched first, enhancing the user experience, which is not present in HTTP/1.1.

**TLS Requirement:**

While TLS (Transport Layer Security) is optional in HTTP/1.1, it is required for HTTP/2. This requirement ensures that communication between clients and servers is encrypted by default, providing security and privacy.

1. **Write a blog about objects and its internal representation in Java script?**

JavaScript, being a versatile and powerful programming language, offers a rich set of features for handling data. One of the fundamental data types in JavaScript is the object. Objects in JavaScript are key to understanding how data is organized and manipulated within the language.

**Objects:**

object in JavaScript is a collection of key-value pairs. This means that each value in an object is associated with a unique key, allowing for efficient storage and retrieval of data. Objects in JavaScript can store various types of data, including strings, numbers, arrays, and even other objects.

**Key Features of JavaScript Objects:**

**Dynamic Nature:** JavaScript objects are mutable, allowing properties to be added, modified, or removed at runtime.

**Prototype-based Inheritance:** Objects in JavaScript can inherit properties and methods from other objects through prototype chaining, enabling the creation of complex inheritance hierarchies.

**JSON Compatibility:** JavaScript Object Notation (JSON) syntax is based on JavaScript object literals, making it easy to serialize and deserialize data for communication between client and server.

**Objects in JavaScript can be represented using several syntaxes:**

**Object Literals:** The most common way to create objects is by using object literals, enclosed in curly braces {}.

**Constructor Functions:** Objects can also be created using constructor functions, which are functions used to initialize new objects with properties and methods.

**JavaScript provides several methods for manipulating objects:**

**Accessing Properties:** Properties of an object can be accessed using dot notation or bracket notation.

console.log(person.name);

console.log(person["age"]);

**Adding Properties:** New properties can be added to an object using assignment.

**Modifying Properties:** Existing properties can be modified by simply assigning a new value.

**Removing Properties:** Properties can be removed from an object using the delete keyword.

1. **Read about IP address, port, HTTP methods, MAC address**

**IP address:**

An IP address, or Internet Protocol address, is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. It serves two main purposes: identifying the host or network interface and providing the location of the device in the network.

There are two primary types of IP addresses: IPv4 and IPv6. IPv4 addresses consist of a series of four numbers separated by periods, such as 192.168.1.1, while IPv6 addresses are longer and use hexadecimal notation, like 2001:0db8:85a3:0000:0000:8a2e:0370:7334.

IP addresses are crucial for routing data packets across the Internet, allowing devices to communicate with each other. They can be either static (unchanging) or dynamic (assigned by a DHCP server and subject to change).

Beyond basic identification, IP addresses can also reveal information about a device's location and network provider, which is why techniques like IP geolocation are used for various purposes, including targeted advertising, content localization, and cybersecurity.

**Port:**

In JavaScript, a "port" typically refers to a communication channel between different processes or environments. This can be relevant in various contexts:

1. **Networking:** Ports are used to identify specific endpoints on a host device in networking. For example, when creating a server with Node.js, you might specify a port number (e.g., 3000) for incoming connections.

2. **Web Development:** In web development, ports are often used to specify different services running on the same server. For instance, you might have a Node.js server running on port 3000 for your backend API, and a separate port for your frontend development server.

3. **Browser Environment:** In web browsers, ports are used in the context of cross-origin requests. For security reasons, browsers restrict JavaScript from making requests to different domains unless those domains explicitly allow it through mechanisms like CORS (Cross-Origin Resource Sharing). Ports play a role in determining whether a request is considered same-origin or cross-origin.

4. **Message Passing:** In environments like Electron or web workers, ports can be used for inter-process communication. This allows different parts of an application to communicate with each other asynchronously.

Overall, in JavaScript, ports are a fundamental concept for facilitating communication between different processes, services, or environments.

**HTTP methods:**

HTTP methods, also known as HTTP verbs, are essential components of the Hypertext Transfer Protocol (HTTP), defining the actions that can be performed on a resource. The primary HTTP methods include:

1. **GET:** Used to retrieve data from the server. GET requests should only retrieve data and should not modify it. They are idempotent, meaning making the same request multiple times produces the same result.

2. **POST:** Used to submit data to be processed to the server. POST requests can create new resources, update existing ones, or trigger some server-side action. They are not idempotent since submitting the same data multiple times may result in different outcomes.

3. **PUT:** Typically used to update or replace an existing resource on the server. PUT requests are idempotent, meaning that making the same request multiple times has the same effect as making it once. PUT requests should contain the entire resource representation, not just the parts that changed.

4. **DELETE:** Used to request the removal of a specific resource from the server. DELETE requests are idempotent, meaning that making the same request multiple times has the same effect as making it once.

5. **PATCH:** Used to apply partial modifications to a resource. Unlike PUT, which typically replaces the entire resource, PATCH is used for modifying specific parts of the resource. PATCH requests are not guaranteed to be idempotent.

6. **HEAD:** Similar to GET, but the server only returns the HTTP header without the actual content of the resource. HEAD requests are useful for obtaining meta-information about a resource without transferring the entire content.

7. **OPTIONS:** Used to request information about the communication options available for a given resource. The server responds with a list of supported methods, headers, and other relevant information.

These HTTP methods allow for various interactions with web servers, enabling the retrieval, creation, modification, and deletion of resources on the web. Choosing the appropriate method for a given task is crucial for building efficient and secure web applications.

**MAC address:**

A MAC address, short for Media Access Control address, is a unique identifier assigned to network interfaces for communications on a network segment. It's like a fingerprint for network devices, allowing them to be uniquely identified. MAC addresses are typically assigned by the manufacturer and are used in technologies like Ethernet and Wi-Fi to ensure data gets to the right destination on a network. They consist of 6 pairs of hexadecimal digits, separated by colons or hyphens, such as "00:1A:2B:3C:4D:5E."