DAY -1 TASK

1.Difference between HTTP1.1 vs HTTP2

|  |  |
| --- | --- |
| **HTTP1** | **HTTP2** |
| **Serial Processing:** Requests and responses are processed serially, meaning one at a time. | **Multiplexing:** Multiple requests and responses can be processed in parallel over a single connection, reducing latency. |
| **Header Inefficiency:** Headers are not compressed, leading to increased overhead due to redundant information in each request and response. | **Header Compression:** Headers are compressed, significantly reducing the amount of redundant information sent with each request and response. |
| **Multiple Connections:** To retrieve multiple resources, multiple connections are required, leading to higher latency. | **Single Connection:** A single connection can be used for multiple concurrent requests, reducing the need for multiple connections. |
| **Head-of-line Blocking:** If one resource is slow to load, it can block the entire page from rendering. | **Stream Prioritization:** Requests can be assigned priority, allowing more critical resources to be delivered first. |
| **No Server Push:** The server cannot push content to the client without a specific request. | **Server Push:** Servers can push content to the client proactively, improving page load times. |
| **Resource Concatenation:** To improve performance, developers often concatenate resources (e.g., CSS and JavaScript files) into larger files. | **Binary Protocol:** Data is sent in binary format, reducing overhead and making it more efficient. |
| **No Binary Protocol:** Data is sent as plain text, making it less efficient. | **Header Compression:** Compression of headers reduces overhead and speeds up data transfer. |
| **No Priority Mechanism:** There is no native support for prioritizing requests. | **No Concatenation Needed:** Resources can be loaded individually, eliminating the need for resource concatenation. |
| **Limited Security:** While HTTPS is possible, it's not mandatory. | **Improved Security:** The use of HTTPS is mandatory for HTTP/2. |
| **Cookie Handling:** Cookies are sent with every request, even for static assets, increasing overhead. | **Backward Compatible:** HTTP/2 is designed to be fully backward compatible with HTTP/1.1, allowing gradual adoption and implementation. |

2.Write a blog about objects and its internal representation in Javascript

### ****Objects: Your Handy Containers****

JavaScript objects are like treasure chests. You can put things inside them, and each thing has a name. These names and things are what we call key-value pairs.

### ****Properties: Your Object's Features****

Think of properties as the special features of your treasure chest. They can be numbers, words, or even other treasure chests! When you create a treasure chest (object), you decide what features it should have.

### ****Methods: Tools for Your Objects****

Some treasure chests come with tools. These tools are like methods in JavaScript objects. They help your objects do things, like calculate numbers or perform special tasks.

### ****Object Creation: Setting Up Shop****

When you make a new treasure chest (object), JavaScript sets up a special place in its memory to store it. This memory not only holds your things but also keeps some hidden details to make everything run smoothly.

### ****Hidden Classes: Organizing for Speed****

JavaScript uses hidden classes to be smart about finding your things quickly. It's like having a map that shows where everything is, making it faster to grab what you need.

### ****Property Map: Neat Storage Space****

Each treasure chest has a space (property map) to store its things. This space is organized so that adding, removing, or finding things is easy and quick.

### ****Prototypes: Objects in Families****

Imagine if your treasure chest could be related to other chests. That's what prototypes do. They create a family tree where chests share some things with their relatives.

### ****Serialization: Turning Treasures into Letters****

Sometimes, you want to send your treasure chest to a friend or save it for later. Serialization is like turning your chest into a letter (string) so that it can be easily sent or stored.

### ****Memory Cleanup: Tidying Up Automatically****

JavaScript has a friend called the garbage collector. When a treasure chest is no longer needed, this friend comes and cleans up the memory, so there's space for new things.

### ****Smart Use of Treasures: Tips for You****

To be a smart treasure hunter (coder), organize your treasure chests well. Don't make too many unnecessary chests, and know how to find things without making a mess.