**DAY 1 TASK**

**Difference between HTTP/2 and HTTP/1.1**

HTTP/1.1 and HTTP/2 are different versions of the Hypertext Transfer Protocol, which is used for communication on the World Wide Web. Here are some key differences between HTTP/1.1 and HTTP/2:

1. **Multiplexing:**

* **HTTP/1.1:** Uses a single connection per request, which means that multiple requests and responses cannot be processed simultaneously on the same connection.
* **HTTP/2:** Supports multiplexing, allowing multiple requests and responses to be sent and received in parallel over a single connection. This can lead to improved performance and reduced latency.

1. **Header Compression:**

* **HTTP/1.1:** Headers are not compressed, leading to a higher amount of overhead and larger data transfer.
* **HTTP/2:** Implements header compression, which significantly reduces the amount of redundant data in the headers, resulting in lower overhead and faster page loading times.

1. **Binary Protocol:**

* **HTTP/1.1:** Uses a text-based protocol, which is human-readable but can be inefficient in terms of parsing and processing.
* **HTTP/2:** Employs a binary protocol, which is more efficient for both machines to parse and for data transmission. This can lead to improved performance.

1. **Server Push:**

* **HTTP/1.1:** Clients have to explicitly request each resource, leading to a less efficient data exchange process.
* **HTTP/2:** Introduces server push, allowing the server to proactively send resources (like images, stylesheets) to the client without waiting for explicit requests. This can lead to faster page loading times.

1. **Prioritization:**

* **HTTP/1.1:** Does not provide mechanisms for prioritizing requests, leading to potential performance issues.
* **HTTP/2:** Supports request prioritization, allowing clients to specify the priority of different resources. This helps optimize the loading of critical resources first, improving overall page loading times.

1. **Connection Management:**

* **HTTP/1.1:** Requires multiple connections to fetch resources in parallel, which can be less efficient.
* **HTTP/2:** Uses a single connection per origin, reducing the overhead associated with multiple connections and improving resource loading times.

|  |  |
| --- | --- |
| **HTTP/1.1** | **HTTP/2** |
| It works on the textual format. | It works on the binary protocol. |
| There is head of line blocking that blocks all the requests behind it until it doesn’t get its all resources. | It allows multiplexing so one TCP connection is required for multiple requests. |
| It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |

In summary, HTTP/2 introduces several features and optimizations over HTTP/1.1, such as multiplexing, header compression, binary protocol, server push, prioritization, and improved connection management, to provide a more efficient and faster web browsing experience.

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

**Introduction:**

* JavaScript, a versatile and dynamic programming language, is widely used for web development due to its ability to create interactive and dynamic content. One of the fundamental features of JavaScript is its support for objects, which play a crucial role in representing and organizing data. In this blog post, we'll delve into the concept of objects in JavaScript and explore their internal representation.

**Understanding Objects in JavaScript:**

* In JavaScript, an object is a composite data type that allows you to store and organize data in key-value pairs. Objects can represent real-world entities and are a cornerstone of the language

**Internal Representation of Objects:**

* Internally, JavaScript engines use various mechanisms to represent objects efficiently. One common approach is using a combination of hash tables and hidden classes.

**Hash Tables:**

* Objects in JavaScript are often implemented using hash tables, which provide fast access to values based on their keys.
* Each property of an object is stored as a key-value pair in the hash table.

**Hidden Classes:**

* JavaScript engines use hidden classes to optimize property access and method invocation.
* When an object is created, the engine assigns it a hidden class that defines the layout of properties.
* Subsequent objects created with the same structure share the same hidden class, improving performance.

**Optimizations and Property Access:**

* JavaScript engines apply optimizations to improve the speed of property access and method invocation. These optimizations include inline caching, which stores property lookup results for faster subsequent accesses.

**Conclusion:**

* Objects are a fundamental part of JavaScript, providing a powerful way to structure and organize data. Understanding the internal representation of objects, including hash tables and hidden classes, can help developers write more efficient code. JavaScript engines employ various optimizations to ensure fast property access and method invocation, contributing to the language's performance.

**-------------------------------------------------------------------------------------------------------------------------------**