1. **Difference between HTTP1.1 vs HTTP2**

|  |  |
| --- | --- |
| **HTTP1.1** | **HTTP 2** |
| * **HTTP/1.1 relies on multiple connections to download different resources simultaneously. Each resource request establishes a new connection, leading to potential inefficiencies due to the overhead of creating and maintaining these connections.** | * **HTTP/2 introduces multiplexing, allowing multiple requests and responses to be sent and received over a single TCP connection. This significantly reduces latency and improves performance by eliminating the need to establish multiple connections for parallel downloads.** |
| * **In HTTP/1.1, headers are sent with each request and response, which can result in significant overhead, especially for small payloads.** | * **HTTP/2 features header compression, where redundant header fields are removed and the remaining headers are compressed using a more efficient encoding scheme like HPACK. This reduces the amount of data transmitted, improving overall efficiency.** |
| * **HTTP/1.1 requires the browser to request each individual resource needed to render a web-page. This can lead to sub-optimal performance, especially for complex pages with many resources.** | * **HTTP/2 introduces server push, enabling servers to proactively send additional resources to the client that it anticipates will be needed, based on the initial request. This can lead to faster page loads by eliminating the need for additional round trips to request essential resources** |
| * **HTTP/1.1 treats all resources as equally important, which can lead to situations where critical resources are delayed by non-essential ones.** | * **HTTP/2 allows for stream prioritization, enabling clients to assign priority levels to different resources. This ensures that critical resources are downloaded and rendered first, leading to improved user experience, particularly on pages with complex layouts or multimedia content.** |
| * **HTTP/1.1 communicates using plain text, which is human-readable but can be less efficient in terms of parsing and transmission.** | * **HTTP/2 uses a binary protocol, which is more compact and efficient for both parsing and transmission by machines. While not human-readable, this format is optimized for performance and reduces overhead, contributing to faster load times and better overall efficiency.** |

1. **Objects and it’s internal representation in JavaScript**

**In JavaScript, objects are fundamental data types that store collections of key-value pairs, often referred to as properties. Here are five key points about objects and their internal representation in JavaScript:**

* **Dynamic Properties:**

**JavaScript objects are dynamic, meaning properties can be added or removed at run time. This flexibility allows for the creation of complex data structures and enables dynamic manipulation of object properties throughout the execution of a program.**

* **Property Descriptor:**

**Each property in a JavaScript object has an associated property descriptor, which defines various attributes such as value, writability, enumerability, and configurability. These descriptors control how the property behaves and can be accessed or modified using methods like Object.getOwnPropertyDescriptor() and Object.defineProperty().**

* **Object Prototype Chain:**

**In JavaScript, objects inherit properties and methods from their prototype, forming a prototype chain. When a property or method is accessed on an object, JavaScript searches up the prototype chain until it finds a matching property or reaches the end of the chain (i.e., the prototype of Object). This mechanism enables property inheritance and allows for efficient memory usage by sharing common properties and methods among objects.**

* **Memory Representation:**

**Internally, JavaScript engines typically use a combination of techniques such as hash tables, dictionaries, and hidden classes to represent objects and optimize property access. Objects are stored in memory as a collection of key-value pairs, with each property occupying memory space proportional to its size and type.**

* **Object Serialization and Deserialization:**

**JavaScript objects can be serialized into JSON (JavaScript Object Notation) format using JSON.stringify(), which converts objects into a string representation suitable for transmission over the network or storage in a file. Conversely, JSON strings can be deserialized into JavaScript objects using JSON.parse(), reconstructing the original object with its properties intact. This serialization process facilitates data exchange between different systems and platforms in a standardized format**