**1.Difference between HTTP1.1 VS HTTP2**

Key Features of HTTP/1.1:

1. Persistent Connections:

In HTTP/1.1, the introduction of persistent connections allows multiple requests and responses to be sent over a single TCP connection. This helps in reducing the overhead of establishing and closing connections for each request.

2. Chunked Transfer Encoding:

HTTP/1.1 supports chunked transfer encoding, which allows the server to send data in chunks rather than sending the entire content at once. This is particularly useful for streaming large files.

3. Pipelining:

Pipelining is a feature in HTTP/1.1 that enables multiple requests to be sent without waiting for the corresponding responses. However, due to head-of-line blocking, browsers often do not fully utilize this feature.

4. No Multiplexing:

Unlike HTTP/2, HTTP/1.1 does not support multiplexing. Each request requires a separate connection, leading to potential performance bottlenecks.

5. Text-Based Protocol:

HTTP/1.1 uses a text-based protocol, which makes it easy for developers to debug using tools like Wireshark. However, the human-readable nature comes at the cost of larger header sizes.

Key Features of HTTP/2:

1. Multiplexing:

Multiplexing in HTTP/2 allows multiple streams (requests and responses) to be sent concurrently over a single connection. This eliminates the need for multiple connections, reducing latency and improving performance.

2. Header Compression:

HTTP/2 uses the HPACK compression algorithm to compress header fields, reducing redundancy and saving bandwidth. This is a significant improvement over the plaintext headers in HTTP/1.1.

3. Binary Protocol:

HTTP/2 employs a binary protocol for data framing. This binary format is more efficient for parsing and serialization, contributing to faster data transmission.

4. Server Push:

Server push is a feature unique to HTTP/2 that enables the server to push additional resources to the client before they are explicitly requested. This helps in proactively delivering assets and improving page load times.

5. Stream Prioritization:

HTTP/2 introduces the concept of stream prioritization, allowing developers to assign priority to different resources. This ensures that critical resources are loaded first, enhancing the user experience.

6. Backward Compatibility:

HTTP/2 is designed to be backward compatible with HTTP/1.1. This means that even if a client or server does not support HTTP/2, the communication can still occur using HTTP/1.1.

Performance Comparison:

Latency Reduction:

HTTP/2 reduces latency by allowing multiple requests to be processed concurrently, eliminating head-of-line blocking.

Header Overhead:

The binary framing and header compression in HTTP/2 significantly reduce the overhead associated with headers, leading to faster data transfer.

Resource Loading:

With server push and multiplexing, HTTP/2 optimizes the loading of resources, resulting in faster and more efficient web page rendering.

Real-world Impact:

Websites using HTTP/2 have reported substantial improvements in performance, especially for complex web applications with many assets.

**2.Objects and its internal representation in javascript**

The Foundation: Objects in JavaScript

In JavaScript, an object is a fundamental data type that allows developers to store and organize data in a structured way. Objects are instances of classes, prototypes, or literals, and they play a pivotal role in the language's object-oriented paradigm.  
  
Creating Objects:

Object Literals:

Example

const person = {

name: 'John Doe',

age: 30,

occupation: 'Developer'

};

Object Constructor:

Example

const car = new Object();

car.make = 'Toyota';

car.model = 'Camry';

Object.create:

Example

const animal = Object.create(null);

animal.type = 'Mammal';

Internal Representation of Objects:

1. Properties:

In JavaScript, objects are essentially collections of key-value pairs, where keys are known as properties. Each property has a name (a string or a symbol) and a corresponding value. The properties of an object can be accessed using dot notation or bracket notation.

2. Prototypes:

Objects in JavaScript can inherit properties and methods from other objects through prototypes. Each object has a prototype, which is a reference to another object. When a property is accessed on an object, JavaScript searches the object's properties and then follows the prototype chain until it finds the desired property.

3. Hidden Classes:

JavaScript engines use the concept of hidden classes to optimize object property access and assignment. Hidden classes help reduce the cost of property access by organizing objects with similar shapes into the same hidden class. When an object's shape changes, a new hidden class is created.

4. Object Descriptors:

Each property in an object is associated with an object descriptor that defines attributes like writable, enumerable, configurable, and value. These descriptors influence the behavior of the property.