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

**HTTP/1.1:** The first usable version of HTTP was created in 1997. This first version of HTTP was called HTTP/1.1. This version is still in use on the web.

**HTTP/2:** In 2015, a new version of HTTP called HTTP/2 was created. HTTP/2 solves several problems that the creators of HTTP/1.1 did not do. HTTP/2 is much faster and more efficient than HTTP/1.1. HTTP/2 is faster by how it prioritizes content during the loading process.

**1. Multiplexing and Concurrency:**

HTTP/1.1 uses separate connections for each file, which can slow down the loading of a web page. But, HTTP/2 introduces multiplexing, which means multiple requests and responses can be sent over a single connection at the same time. This allows for better use of available network resources and speeds up the loading process.

**2. Header Compression:**

Every time when browser requests a file from a web server, it sends along a set of headers containing important information. In HTTP/1.1, these headers are sent in plain text, which adds unnecessary overhead and increases the amount of data transferred. HTTP/2, compresses these headers, reducing their size and improving overall efficiency.

**3. Server Push:**

HTTP/2 introduces a feature called server push, which allows a web server to proactively send additional resources to your browser before they are even requested. This reduces the need for multiple round trips between your browser and the server, resulting in faster page load times and a smoother browsing experience.

**4. Prioritization:**

In HTTP/1.1, all requests are processed in the order they are received. This means that if a critical resource is requested after less important ones, it may cause delays in rendering the page. HTTP/2 addresses this issue by allowing the client to prioritize resources. This ensures that important resources are fetched and displayed first, optimizing the user experience.

**5. Stream and Binary Protocol:**

HTTP/1.1 uses plain text for communication, which is easy for humans to read but less efficient for machines. HTTP/2, on the other hand, employs a binary protocol, which is more compact and faster to process. Additionally, HTTP/2 introduces the concept of streams, allowing multiple requests and responses to be processed simultaneously, further improving performance.

**6. Compatibility and Adoption:**

One of the great things about HTTP/2 is its compatibility with existing web infrastructure. You don't need to change your browser or website completely to benefit from HTTP/2. Most modern web browsers and servers already support HTTP/2, making it easy to adopt and enjoy its advantages.

**2. Objects and Their Internal Representation in JavaScript**

**Objects:**

An object is a collection of key-value pairs, where each value can be of any data type. Objects are containers that hold related information and they are a way to access and manipulate that information. Objects represent real-world entities, such as a person, a car, or an event, and their properties and behaviors.

**Internal Representation of Objects:**

JavaScript objects are implemented using a variety of techniques. One common approach is using a hash table or a dictionary-like data structure.

1. **Key-Value Pairs**:

Objects in JavaScript are essentially collections of key-value pairs. Each key acts as an identifier, and the associated value can be of any data type, including other objects. These key-value pairs are stored within the object's internal structure.

2**. Hashing and Memory Allocation:**

When we create an object in JavaScript, memory is allocated to store its properties and their corresponding values. To efficiently access these properties, a process called hashing is used. Hashing involves applying a specific algorithm to the key to generate a unique hash value. This hash value is then used as an index to locate the corresponding value in memory.

3. **Dynamic Nature:**

JavaScript objects are dynamic in nature. This means that you can add, modify, or delete properties from an object at any time, even after the object has been created. This flexibility is possible because JavaScript objects use a flexible data structure that can accommodate changes dynamically.

4. **Object Prototypes:**

JavaScript objects have prototypes, which serve as a template or blueprint for creating new objects. Prototypes allow objects to inherit properties and methods from other objects, helps in code reuse and inheritance in JavaScript.