1. **Difference between HTTP 1.1 and HTTP 2**

| **HTTP 1.1** | **HTTP 2** |
| --- | --- |
| Uses multiple connections, limited by browser constraints. | Supports simultaneous data transfer over a single connection, improving efficiency. |
| Headers are sent in plain text, leading to redundancy. | Headers are compressed, reducing overhead and improving speed. |
| Text-based protocol. | Binary protocol, making it more efficient to parse and faster. |
| Lacks built-in mechanisms for prioritizing requests. | Allows prioritization of requests, ensuring more important resources are fetched first. |
| Requires multiple requests to fetch related resources. | Supports server push, allowing the server to send resources proactively to the client. |
| Requires multiple connections for parallelism. | Uses a single connection per origin, reducing latency. |
| Relies on the TCP layer for flow control. | Has its own flow control mechanisms, allowing better control over data transmission. |
| Errors can delay the loading of subsequent resources. | Multiplexing allows independent processing of resources, minimizing the impact of errors. |
| Backward compatible with HTTP/1.0. | Not backward compatible with HTTP/1.1, but designed to degrade gracefully. |
| Simpler to implement. | More complex due to features like multiplexing and header compression. |

2. **Objects and internal internal representation in JavaScript.**

Objects are like containers that can hold various types of information in a structured way. They are used to represent real-world entities and their properties.

Example: let bike= {

name: 'Ducati panigale v4r',

CC: 999

topSpeed: 330

cost:” expensive”;

};  
  
**Object Properties:**

Each property in an object has two main parts:

Property Name (Key): The identifier for a property.

Property Value: The data associated with the property.

In the above example, the ‘name’ is the key and the value is ‘Ducati Panigale v4r’  
  
**Memory Allocation:**

JavaScript engines use memory to store objects and their properties. When you create an object, memory is allocated to hold its data.

JavaScript objects are dynamic, meaning you can add, modify, or remove properties at runtime.

**Conclusion:**

Objects in JavaScript are versatile data structures used to represent and organize information. Their internal representation involves memory allocation and dynamic handling of properties. Understanding these concepts is crucial for effectively working with JavaScript objects.