1. Objects in JavaScript

- JavaScript is a computer language that is very flexible and popular, and it uses objects a lot. Learning objects and how they are internally represented is essential to become proficient in JavaScript development.
- In JavaScript, objects are one of the most fundamental data structures. They are collections of key-value pairs, where keys are strings (or symbols) and values can be of any data type, including other objects, functions, arrays, and primitive values.
- Objects in JavaScript are dynamic, meaning you can add, modify, or delete properties and methods at runtime.
- Creating Objects in JavaScript

  There are several ways to create objects in JavaScript:

  Object Literals: Using curly braces {} to define key-value pairs.

  Constructor Functions: Using functions as templates to create multiple instances of objects using the new keyword.

  Object.create() method: Creating objects with a specified prototype.

  Class syntax (ES6): Defining objects using the class keyword and constructor method.

- The internal representation includes hidden classes for object structures, prototype chains for property lookup, and garbage collection for memory management, optimizing JavaScript execution.

```
const person = {
  firstName: 'John',
  lastName: 'Doe',
};
```

2. Difference between HTTP/1.1 vs HTTP/2

HTTP 1.1	HTTP2
In HTTP/1.1, browsers typically establish multiple TCP connections to fetch resources in parallel. However, each connection can only handle one request at a time, leading to inefficient resource utilization.	HTTP/2 introduces multiplexing, allowing multiple requests and responses to be sent and received on the same TCP connection concurrently. This feature significantly improves the efficiency of resource loading, especially for complex web pages with many resources.
HTTP/1.1 uses plain text for communication between clients and servers, which can be human-readable but inefficient in terms of parsing and transmission.	HTTP/2 is a binary protocol, meaning that data is encoded in binary format rather than plain text. This allows for more efficient parsing and transmission of data, reducing latency and improving performance.
HTTP/1.x uses formats like gzip to compress the data transferred in the messages. However, the header component of the message is always sent as plain text.	HTTP/2 uses HPACK compression to decrease the average size of the header. This compression program encodes the header metadata using Huffman coding, which significantly reduces its size as a result. In addition, HPACK keeps track of previously transferred header values and further compresses them as per a dynamically modified index shared between client and server.
HTTP/1.1 is widely supported by web servers and browsers and is compatible with most existing web infrastructure.	HTTP/2 offers significant performance improvements, adoption has been slower due to the need for server and browser support. However, most modern web servers and browsers now support HTTP/2