Asynchronous programming

- Asynchronous programming in JavaScript is a programming paradigm that allows you to perform tasks concurrently without blocking the main execution thread. In traditional synchronous programming, tasks are executed one after the other, and the program waits for each task to complete before moving on to the next one.
- Asynchronous programming in JavaScript relies on non-blocking operations and the use of callbacks, promises, or async/await syntax to handle asynchronous tasks.

Libuv & Event Loop

Libuv includes an event loop that manages and dispatches events and callbacks for asynchronous operations. This event loop is at the heart of Node.js and is responsible for handling non-blocking operations, making it possible to handle many concurrent connections efficiently.

Domain Name System Resolution (DNS)

Domain Name System resolution, or DNS resolution, is the procedure that transforms human-friendly domain names (like www.example.com) into the IP addresses that computers use to recognize one another on a network. Because it allows us to utilize domain names rather than difficult-to-remember IP numbers to access servers and websites.

Multiple Synchronous Task

Multiple synchronous task execution in JavaScript involves performing a number of operations one after another, making sure that each job is finished before the next one starts. We don't require additional structures for synchronous tasks because JavaScript is essentially single-threaded, which indicates that tasks are executed sequentially.

Preserving Order

Preserving order refers to maintaining the sequence or arrangement of items or elements in the same order as they were originally presented or specified.

zlib.inflate(buffer, callback)

The zlib.inflate() function in Node.js is used to decompress data that has been compressed using the zlib compression algorithm. It takes a compressed buffer as input which is Binary representation of the compressed data and uses a callback function to handle the decompressed data.