**Node.JS event loop**

The event loop has six phases. This allows Node.JS to perform Non-blocking I/O operations.

**Timer phase:**

* The setTimeout() and setInterval() place call backs in the timer queue.
* Event loop enters the timer phase, the first phase, and checks for any event in the timer queue and executes it.
* The event loop pushes the call back functions from the timer queue to the call stack queue where the processing takes place.

**Pending I/O phase:**

* The event loop checks for pending I/O call backs and executes it.
* All pending I/O call backs are placed in the pending I/O queue.
* During this phase the event loop pushes it to the call stack and executes it

**Idle, Prepare phase:**

* Performs libuv internal stuff and prepares for the polling phase

**Polling phase:**

* Retrieves new I/O events by performing polling.
* So this phase accepts new incoming connections.
* Adds the events to their corresponding queue for the event loop to execute it.

**Check handlers:**

* The call backs from setImmediate() functions are executed here by the event loop.
* setImmediate() calls are added to the immediate queue.
* They are executed once the polling phase completes for an iteration.

**Close handlers:**

* Executes close or destroy callbacks like close connections in sockets.
* After executing this phase the event loop checks whether the loop is alive or not. If so then starts executing the states again for pending call backs from the corresponding queues.