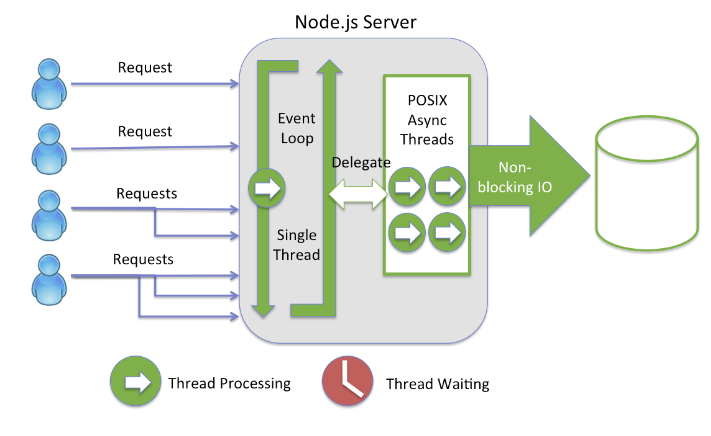
**Working of nodeJS single threading to handle multiple requests**

NodeJS application uses the “Single Threaded” Event Loop Model to handle multiple concurrent clients. There are many web application technologies like JSP, Spring MVC, ASP.net, HTML, AJAX, JQuery etc. All these technologies have “Multi threaded Request-Response” architecture to handle multiple concurrent clients.

Node JS Platform does not follow the Request/Response Multi-Threaded Stateless Model. It follows Single Threaded with Event Loop Model. Node JS Processing model mainly based on Javascript Event based model with Javascript callback mechanism.

****

Multiple clients make multiple requests to the NodeJS server. NodeJS receives these requests and places them into the **EventQueue .**

NodeJS server has an internal component referred to as the EventLoop which is an infinite loop that receives requests and processes them. This EventLoop is single threaded. In other words, EventLoop is the listener for the EventQueue. As Node JS follows this architecture, it can handle more and more concurrent client requests very easily.

The main heart of Node JS Processing model is “Event Loop”.

Here are Single Threaded Event Loop Model Processing Steps:

* Clients Send requests to Web Server.
* Node JS Web Server internally maintains a Limited Thread pool to provide services to the Client Requests.
* Node JS Web Server receives those requests and places them into a Queue. It is known as “Event Queue”.
* Node JS Web Server internally has a Component, known as “Event Loop”. Why it got this name is that it uses an indefinite loop to receive requests and process them. (See some Java Pseudo code to understand this below).
* Event Loop uses Single Thread only. It is the main heart of Node JS Platform Processing Model.
* Even Loop checks if any Client Request is placed in the Event Queue. If not, then wait for incoming requests for indefinitely.
* If yes, then pick up one Client Request from Event Queue
* Starts process that Client Request
* If that Client Request Does Not require any Blocking IO Operations, then process everything, prepare response and send it back to client.
* If that Client Request requires some Blocking IO Operations like interacting with Database, File System, External Services then it will follow different approach
* Checks Threads availability from Internal Thread Pool
* Picks up one Thread and assigns this Client Request to that thread.
* That Thread is responsible for taking that request, process it, perform Blocking IO operations, prepare response and send it back to the Event Loop
* Event Loop in turn, sends that Response to the respective Client.

**References:-**

[**https://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop**](https://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop)