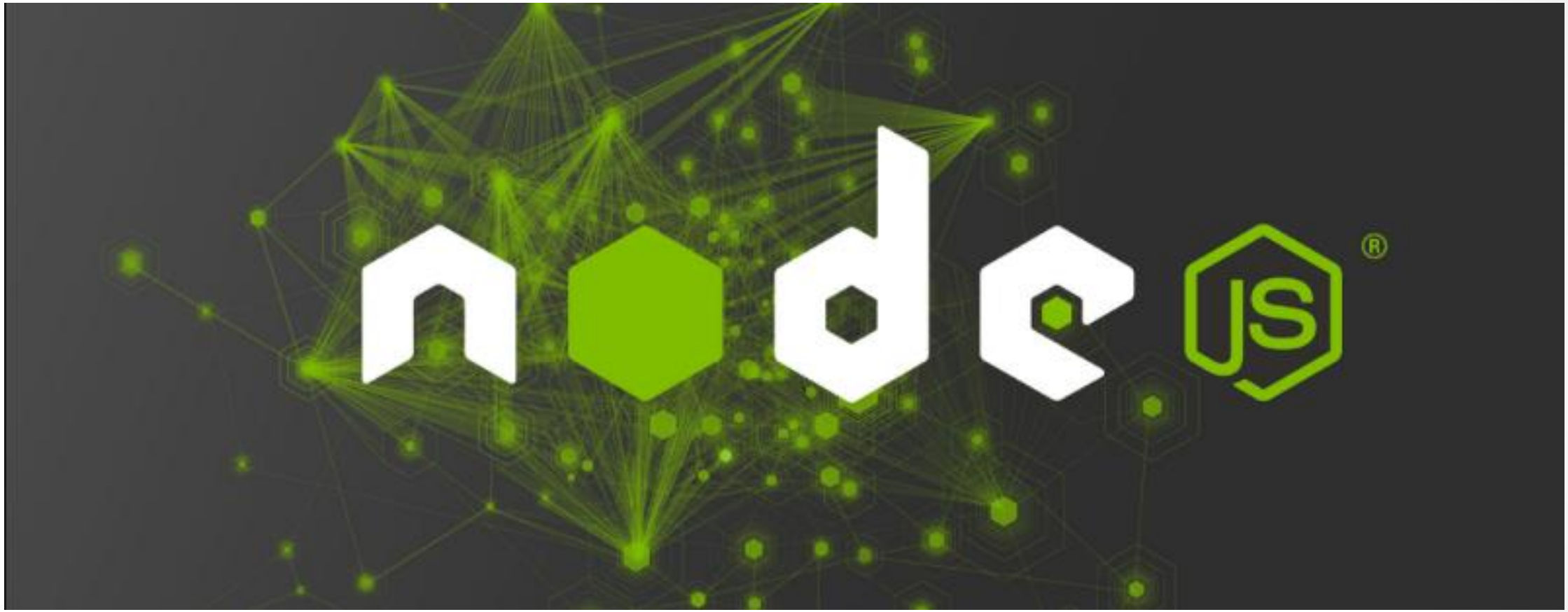

node
JS

What is Node.js

- **Node.js is a JavaScript runtime environment**. But what is that, one might ask. By run-time environment, the infrastructure to build and run software applications is meant. To build applications in JavaScript.

- Node.js as a “JavaScript runtime built on Chrome V8 engine”.
- “Node.js is an open-source and cross-platform environment to execute code”.
- “A development platform aimed at building server-side applications”.
- Node.js is “a platform with its own web server for better control”. That is certainly enough to grasp the main idea.

Summary

- Node.js is a server framework, and is free
- It runs on Windows, Linux, Mac OS, and so on
- Node.js utilizes JavaScript on the server

Reasons to use Node.js

- Good for beginner developers, JavaScript is simple to learn, rich framework (Angular, Node, Backbone, Ember)
- It is fast, due to Google innovative technologies and the event loop
- Ability to keep data in native JSON (object notation) format in your database
- Multiple modules (NPM, Grunt, etc.) and supportive community
- Good to create real-time apps, such as chats and games
- Single free codebase
- Good for data streaming, thus for audio and video files, as example
- Sponsored by Linux Foundation, as well as PayPal, Joylent, Microsoft, Walmart
- Wide range of hosting options
- JavaScript is the longest running language, 99% of developers know some of it

Event Loop

- Node.js is a single-threaded event-driven platform that is capable of running non-blocking, asynchronously programming. These functionalities of Node.js make it memory efficient. The **event loop** allows Node.js to perform non-blocking I/O operations.

Features of Event Loop:

- Event loop is an endless loop, which waits for tasks, executes them and then sleeps until it receives more tasks.
- The event loop executes tasks from the event queue only when the call stack is empty i.e. there is no ongoing task.
- The event loop allows us to use callbacks and promises.
- The event loop executes the tasks starting from the oldest first.

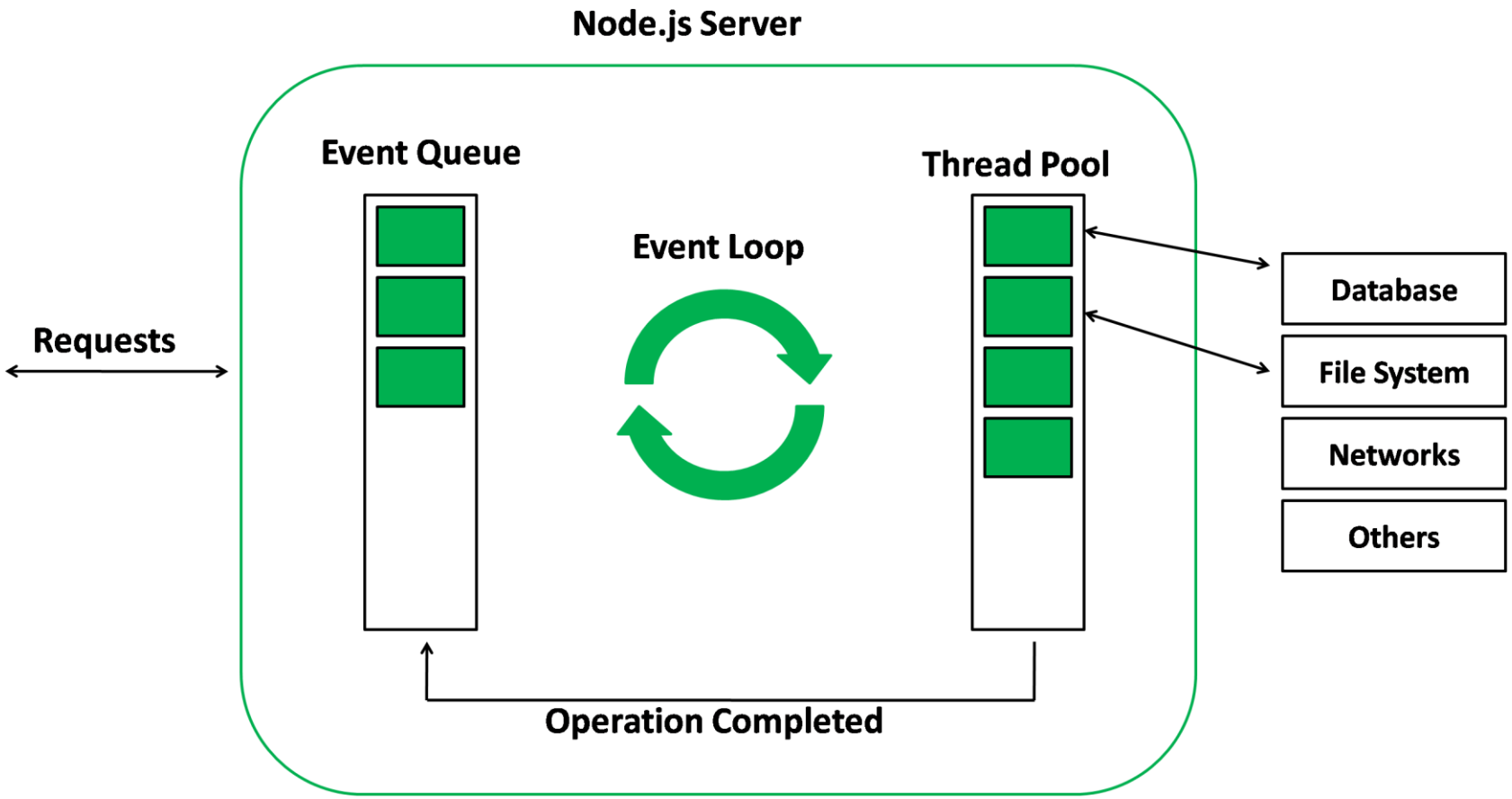
```
console.log("This is the first statement");

setTimeout(function(){
        console.log("This is the second statement");
}, 1000);

console.log("This is the third statement");
```

Event loop Working (Working of Example)

- In the above example, the first console log statement is pushed to the call stack and “This is the first statement” is logged on the console and the task is popped from the stack.
- Next, the setTimeout is pushed to the queue and the task is sent to the Operating system and the timer is set for the task. This task is then popped from the stack.
- Next, the third console log statement is pushed to the call stack and “This is the third statement” is logged on the console and the task is popped from the stack.
- When the timer set by setTimeout function (in this case 1000 ms) runs out, the callback is sent to the event queue. The event loop on finding the call stack empty takes the task at the top of the event queue and sends it to the call stack. The callback function for setTimeout function runs the instruction and “This is the second statement” is logged on the console and the task is popped from the stack.

Working of the Event loop

- When Node.js starts, it initializes the event loop, processes the provided input script which may make async API calls, schedule timers, then begins processing the event loop. In the previous example, the initial input script consisted of console.log() statements and a setTimeout() function which schedules a timer.
- When using Node.js, a special library module called libuv is used to perform async operations. This library is also used, together with the back logic of Node, to manage a special thread pool called the libuv thread pool.This thread pool is composed of four threads used to delegate operations that are too heavy for the event loop. I/O operations, Opening and closing connections, setTimeouts are the example of such operations.
- When the thread pool completes a task, a callback function is called which handles the error(if any) or does some other operation. This callback function is sent to the event queue. When the call stack is empty, the event goes through the event queue and sends the callback to the call stack.


Node.js Server
Event Queue
Thread Pool
Event Loop
Requests
Database
File System
Networks
Others
Operation Completed