Node.js -Blocking and Non-Blocking, Asynchronous operations and Callbacks.

# Overview Of Node.js:-

Node.js is a powerful framework for building backend systems which can scale very well. It is developed on Chrome’s V8 JavaScript Engine that compiles the JavaScript directly into the native machine code.

Node.js framework is used for developing server side applications and extends JavaScript API to provide the server side functionalities. It is generally used for developed Single Page Applications, Video Streaming Sites and other web applications.

The Asynchronous coding paradigm enables us to write a non-blocking code. This helps the single threaded JavaScript run with full efficiency. A Single thread is a thread in node.js that can do only one thing at a time.Node.js uses this Single thread to get non-blocking execution.

This blog provides the basic Introduction of Node.js with some major features of Node.js and also contains detailed explanation of Blocking and Non-blocking, Asynchronous operations and callbacks in node.js.

# Major Features of Node.js:-

# Open Source:-

Node.js is an open source framework that is supported by a huge community.Node.js community is pretty much active have contributed to add new capabilities to node.js applications.

# Simple and Fast:-

Node.js is Chrome’s v8 JavaScript Engine.Node.js libraries are capable of fast code execution.

# Asynchronous:-

All Node.js libraries are asynchronous which means the node.js based servers never wait for an API to send back the response and move on to next API .

# Single Threaded:-

Node.js is able to follow the single threaded model due to which the single program handles multiple requests with the help of event looping.

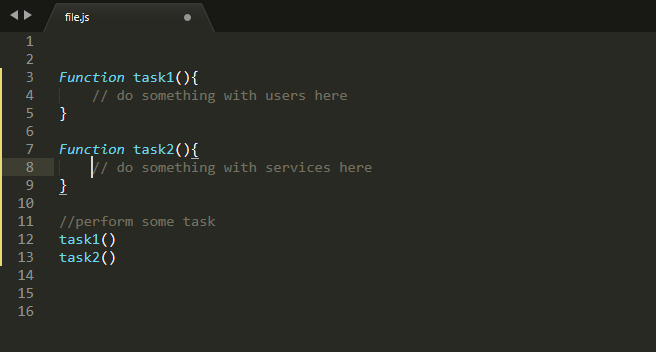
# No Buffering:-

Node.js have one of the major features that it never buffers any data.

# Cross platform:-

Node.js can easily built and deployed on various platforms like Windows, Mac and Linux.

Now, we will understand Blocking and Non-blocking in node.js-



In this example code **task1()** function executes first and after it returns then **task2()** function executes. **This is not a blocking code in javascript** even if the **task1()** takes more time before returning(like it performs some CPU intensive task like find a inverse of matrix).

# Blocking in Node.js:-

When JavaScript execution in node.js process(Each program is a process) has to wait until a non-JavaScript operation completes its execution is called blocking in node.js.

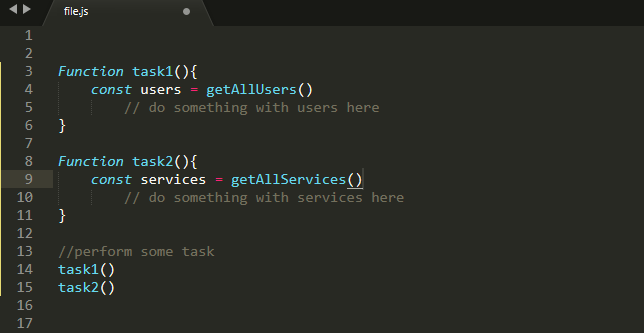
# Non-Blocking in Node.js:-

It is just opposite of blocking i.e When JavaScript execution in node.js process(Each program is a process)do not wait until a non-JavaScript operation completes is called non-blocking.

**Non JavaScript execution refers to mainly I/O operations .I/O operations are blocking.**

**I/O firstly refers to the interaction with the system’s disk and network.**

**Let’s take an another Example:-**

****

In this Example **task1** internally calls the **getAllUsers** which makes the database and send the request to database to fetch all the users.**task2** internally calls **getAllServices** which makes a HTTP request to get all the services of some third party through their API.

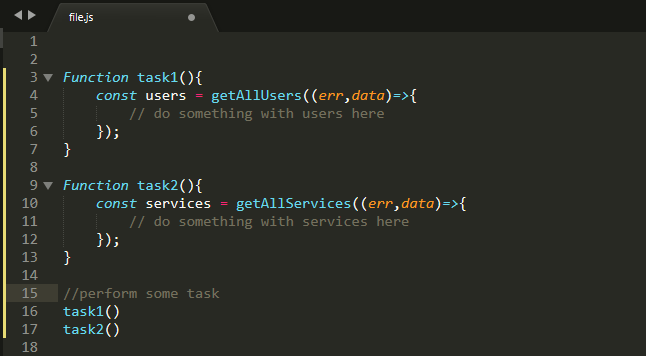
**Here task2() will be blocked till task1() returns because a single thread can execute only one thing at a time.**

Now, you must be know that how node.js can convert the blocking calls into non-blocking execution.

It uses the Event Loop and callbacks mechanism to move the I/O operations from the JavaScript thread to the system’s kernel.

Most of the System kernels are multi-threaded, they can handle multiple requests and executes operations in the background concurrently. When one of these operations executes completely, the kernel informs the node.js and then the callback may be added to the queue to be executed.

So, Here we convert the blocking code into the non-blocking code using callbacks.

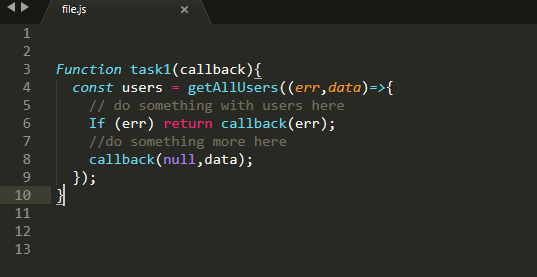


Here in the Above code **getAllUsers** and **getAllServices** take callbacks. This callback mechanism is used by the node.js and called callbacks when the kernel is finished with the I/O operations.

# Structure of Callback in Node.js:-

A Callback is a JavaScript function which is called when the given task is completed. It has some conventions for this callback functions:-

* The Callback function is passed as the last parameter to any function.
* Callback function is called after the function is done with all of its operations.
* All Callback functions have error value as a first parameter. If there is no error then first parameter is set to be null and rest being return value.



Now I think you must understood that node.js is able to make asynchronous calls for non-blocking operations using callbacks mechanism.

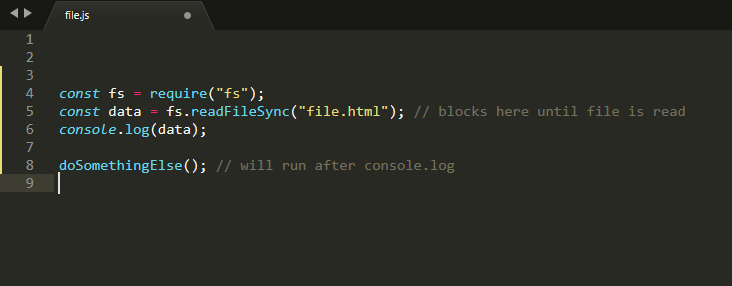
Node.js makes the framework very fast and efficient one of the reason in node.js that no function directly performs I/O. So, the process never blocks. we used callbacks in these asynchronous calls allows you to have as many I/O operations as your OS can handle , happening simultaneously.

Both Blocking and non-blocking variants are provided by the many functions in node.js.

* The non-blocking code invariably takes a callback function as a parameter.
* Some of the blocking equivalent names usually ends with **Sync**. These functions execute synchronously and blocked the code.

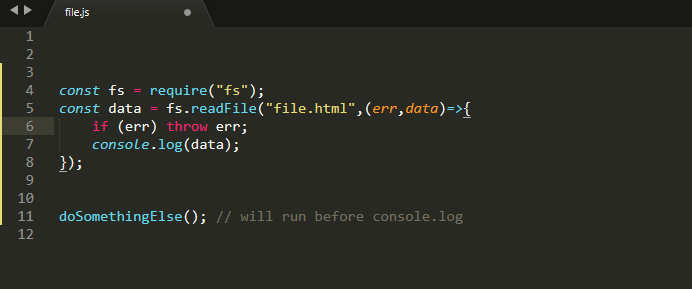
Lets take an Example of Synchronous & Asynchronous modes using file system:-

# Synchronous file read:-



**In Synchronous version if an error is thrown then it will have to be caught else the process will be crashed.**

# Asynchronous file read:-



**In Asynchronous version developer is responsible for choosing the way of error handling. We should always use non-blocking variant over the locking variant of the function.**

# Conclusion:-

Node.js is an outstanding flexible and customizable JavaScript framework. Due to its high popularity in the IT industry more node.js developers will be required in the future. The node.js web applications load faster as compared to the other frameworks .It provides several benefits to the developers which is basically the actual reason why some of the best companies around the world use node.js for their projects.