基于异步事件模型的Node.js大致是这样运行的：

向Node.js提交异步操作，比如建立网络连接，读取网络流数据，向文件写入数据，请求数据库服务等，同时针对这些异步操作注册回调函数。这些异步操作会提交给IO线程池或者工作线程池。

在线程池中，操作是并发的执行，也就是读网络流和向文件写数据，或者请求数据库服务都是并发的（可能是这样子的，具体的操作怎么完成，是node的事） ，执行完毕后会将就绪事件放入完成队列中。

Node.js 在提交完操作请求之后，进入循环(或叫事件循环吧)。循环的过程如下：

a. 检查有没有计时器超时(setTimeout/setInterval)

b. 检查当前是否为空闲状态，执行空闲任务(process.nextTick)

c. 检查IO完成队列（各种网络流读写、文件读写、标准输入输出上的事件都会进入这个队列）是否有就绪事件，

若完成队列中有就绪事件，就把队列里的所有事件（可能有多个操作已经完成）信息都取出来，对这些事件信息，挨个地调用与其相关的回调函数。这个过程是同步的，执行“写数据完成”事件的回调函数完成之后，才会去调用“读到网络数据”事件的回调函数；

若是队列中没有就绪事件，而且没有空闲(idle)任务，就会做一段时间的等待（线程被阻塞在此处），等待的超时时间由计时器周期决定。（不能因为等待而耽误timer和idle的事件处理）。

d. 进入下一轮循环。

从上面这个过程可以看出，你脚本中注册的所有回调函数都是在这个循环过程中被依次调用的。若有一个回调函数执行大的计算任务，很长时间不返回的话，就会让整个循环停顿下来，其它回调函数就不能在事件到来时即时被回调。因此，建议长任务处理过程中，即时将剩下的处理通过process.nextTick丢入下一轮循环中有idle事件中，或者process.spawn一个进程来执行。

总之，除了你的代码是同步执行的以外，其它所有的事情都是并发的。

原文：https://blog.csdn.net/yusiguyuan/article/details/20405011