1.什么是i/o？

io input、output 输入输出，电脑的输入输出，例如音频录音表示声音输入、听音乐是声音的输出网络上的传输全部是在传字符串，i/o在服务器上可以理解为读写操作。

## 2.什么是并发？

一个时间段中有几个程序都处于已启动运行到运行完毕之间。

## 3.异步i/o与事件驱动

### 3.1什么是进程？

进程是为运行当中的应用程序提供运行环境的，一个运行当中的应用程序就会有一个进程与之相对应。

### 3.2什么是线程？

线程是用来运行应用程序中代码的，一个线程在一个时间只能做一件事件。

多线程，调度起来很麻烦。

node是单线程执行，用异步替代了多线程。

### 3.3同步、异步有什么不同？

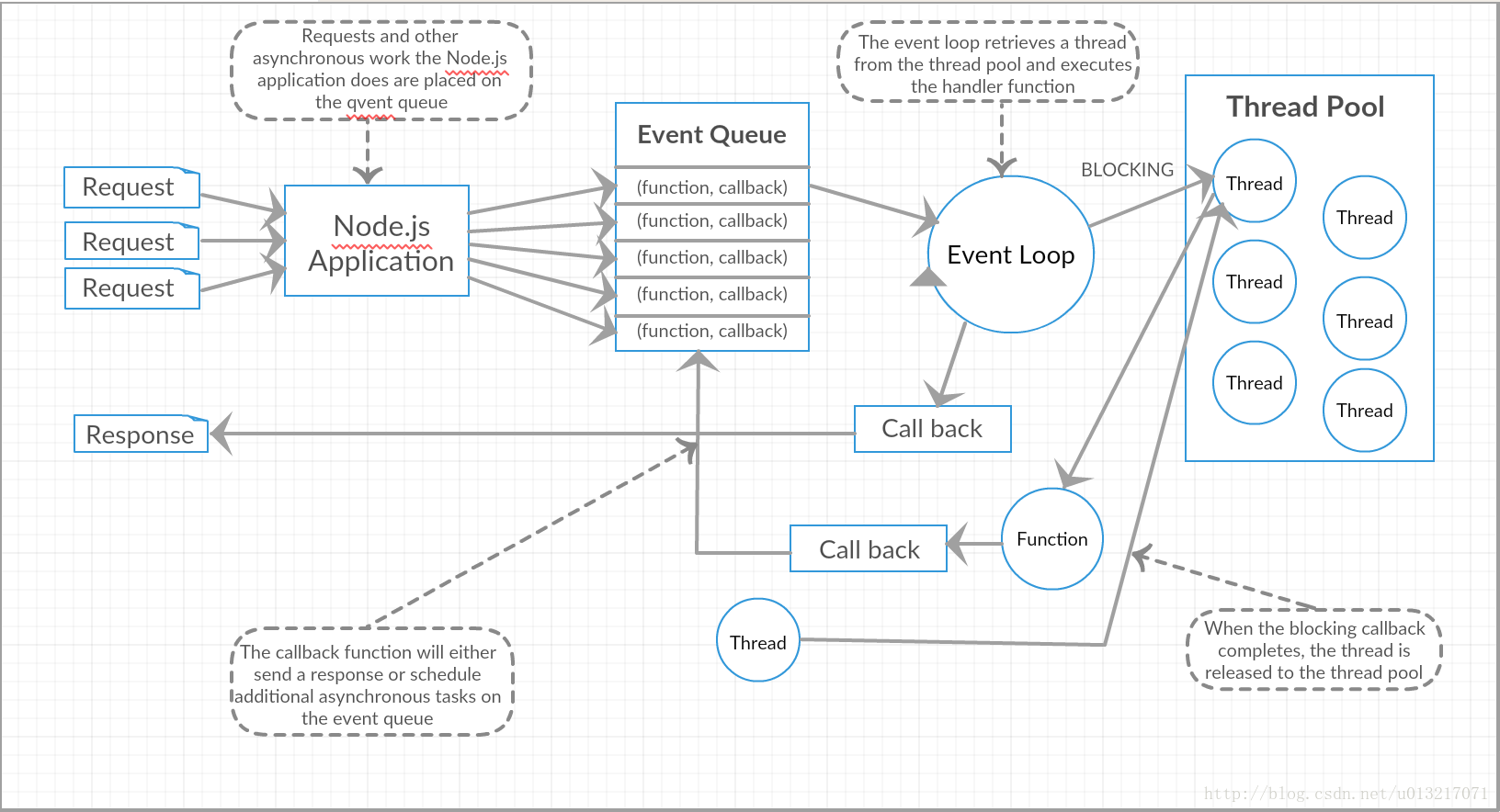
异步不会阻塞后面的代码，同步会阻塞后面的代码；一条线程先执行同步的代码后执行异步的代码。

### 3.4异步非i/o操作和异步i/o操作

异步非io setTimeout setInterval；

异步IO操作 操作文件 网络操作 fs。

### 3.5node的事件驱动模型？



**主线程：**

* 1.执行node的代码，把代码放入队列
* 2.事件循环程序（主线程）把队列里面的同步代码都先执行了，
* 3.同步代码执行完成，执行异步代码
* 4.异步代码分2种状况，
  + 1异步非io setTimeout() setInterval()   
    判断是否可执行，如果可以执行就执行，不可以跳过。
  + 2异步io 文件操作   
    会从线程池当中去取一条线程，帮助主线程去执行。

主线程会一直轮询，队列中没有代码了，主线程就会退出。

**子线程：**被放在线程池里面的线程，用来执行异步io操作

* 1.在线程池里休息
* 2.异步io的操作来了，执行异步io操作。
* 3.子线程会把异步io操作的callback函数，扔回给队列
* 4.子线程会回到线程池去休息。   
  callback 在异步io代码执行完成时被扔回主线程。

### 3.6异步和多线程的比较？

node的异步是帮助我们去做了多线程的操作，简化了代码。

nodejs的最大的优点是高并发，适合适合I/O密集型应用

痛点：比如现在要执行4个步骤，依次执行，于是就有下面的代码

step1(function (value1) {

step2(value1, function(value2) {

step3(value2, function(value3) {

step4(value3, function(value4) {

// Do something with value4

});

});

});

});

这里只有4个，那如果更多呢？解决：promise/a+规范