

**二、同步任务和异步任务**

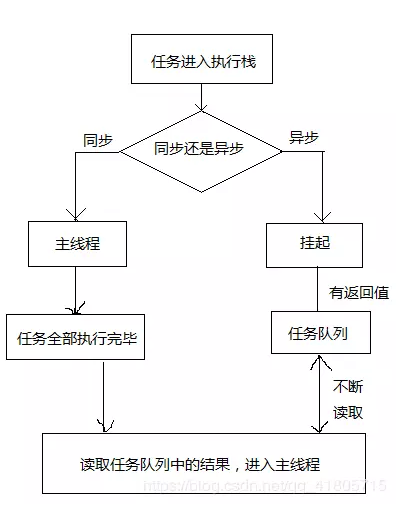
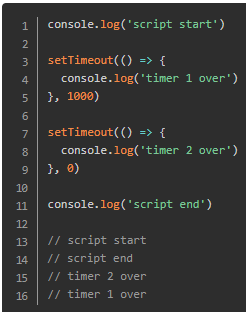
javascript是单线程。单线程就意味着，所有任务需要排队，前一个任务结束，才会执行后一个任务。如果前一个任务耗时很长，后一个任务就不得不一直等着。  
于是js所有任务分为两种：同步任务，异步任务  
**同步任务**是调用立即得到结果的任务，同步任务在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务；

**异步任务**是调用无法立即得到结果，需要额外的操作才能预期结果的任务，异步任务不进入主线程、而进入"任务队列"（task queue）的任务，只有"任务队列"通知主线程，某个异步任务可以执行了，该任务才会进入主线程执行。  
JS引擎遇到异步任务（DOM事件监听、网络请求、setTimeout计时器等），会交给相应的线程单独去维护异步任务，等待某个时机（计时器结束、网络请求成功、用户点击DOM），然后由 事件触发线程 将异步对应的 回调函数 加入到消息队列中，消息队列中的回调函数等待被执行。

**具体来说，异步运行机制如下：**

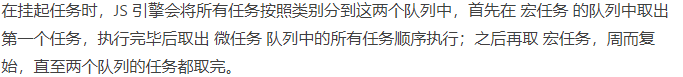
* （1）所有同步任务都在主线程上执行，形成一个[执行栈]
* （2）主线程之外，还存在一个"任务队列"（task queue）。只要异步任务有了运行结果，就在"任务队列"之中放置一个事件。
* （3）一旦"执行栈"中的所有同步任务执行完毕，系统就会读取"任务队列"，看看里面有哪些事件。那些对应的异步任务，于是结束等待状态，进入执行栈，开始执行。
* （4）主线程不断重复上面的第三步。

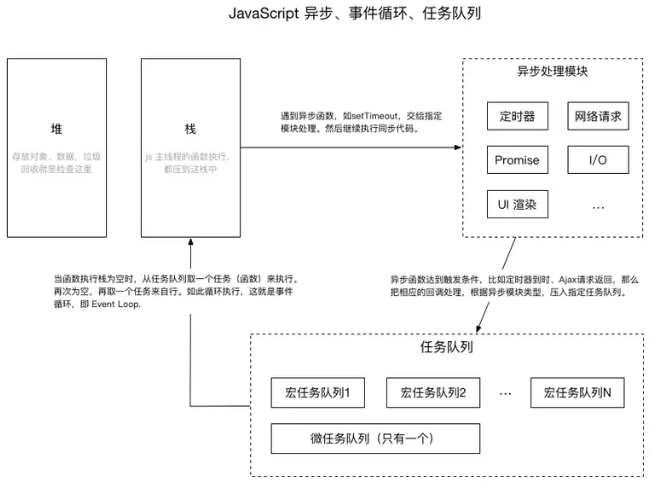
主线程从"任务队列"中读取事件，这个过程是循环不断的，所以整个的这种运行机制又称为Event Loop（事件循环）

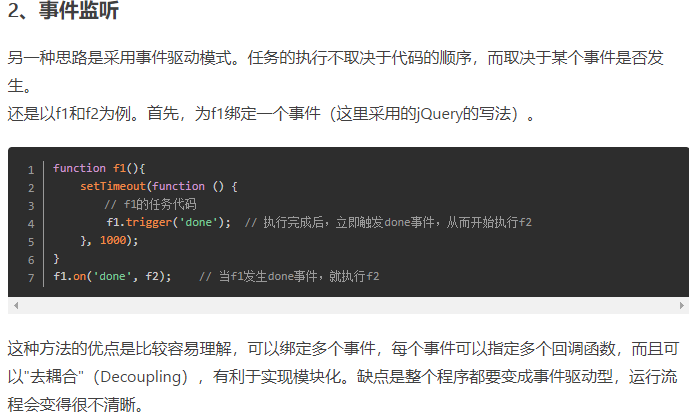
timer 2 over0毫秒后添加到任务队列队尾，timer 1 over1秒添加到任务队列队尾，等待主线程任务执行完，从队头依次执行任务队列中的任务





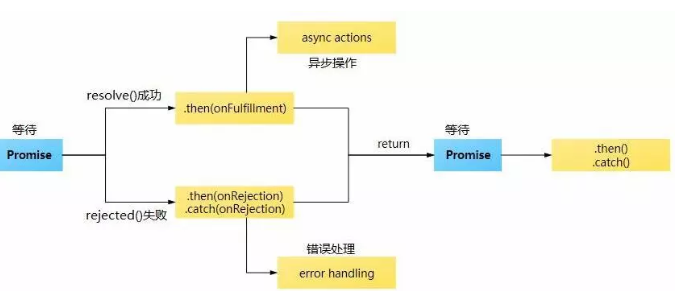


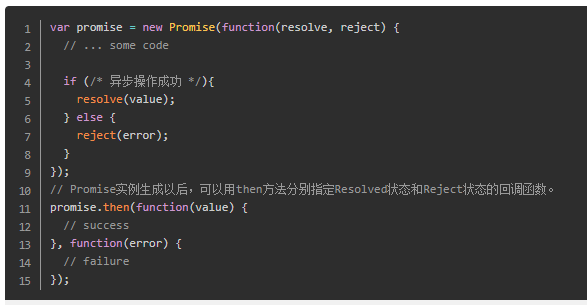


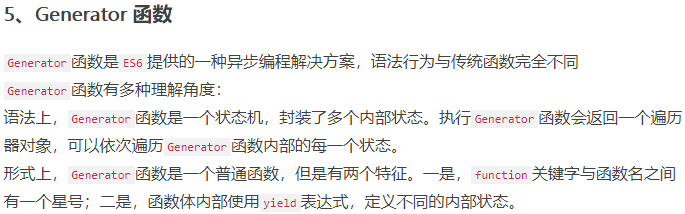




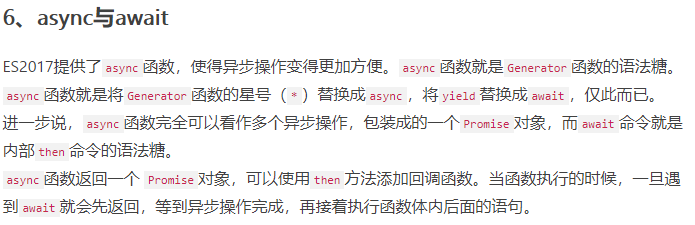


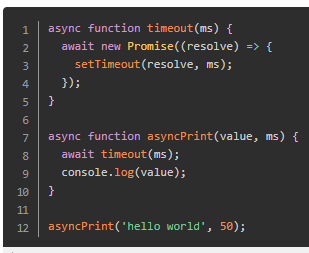












上面代码指定 50 毫秒以后，输出hello world。