

# 时间循环

1. 同步和异步任务分别进入不同‘场所’，同步的进入主线程，异步的进入Event Table 并注册函数。
2. 当指定的事情完成时（如 promise 被resovlve）, Event Table 会将对应函数移入Event Queue。
3. 主线程的任务执行完毕（也就是执行栈为空），回去Event Queue读取对应的函数，进入主线程执行。
4. 上述过程不断重复，也就是 Event Loop(时间循环)

# 监听主线程执行栈

js引擎用 monitoring process进程，会持续不断的检查主线程执行栈是否为空。一旦为空，就回去Event Queue那里检查是否有等待被调用的函数。

# 定时器

setTimeOut

setTimeOut是异步的

延迟相应时间后将 绑定 函数被推入Event Queue

setTimeout(fn,0)

其真实含义是：

fn 在主线程最早可得的空闲时间执行。也就是说只有执行栈内的同步任务全部执行完成，栈为空就马上执行。

实际上根据HTML标准，最低是4ms的Event Loop

setInterval

每隔相应的延迟，就会把fn 推入Event Queue。

promise.nextTick(cb)

在时间循环的下一次循环中调用cb。

处理广义的同步任务和异步任务，我们对于任务有更精细的定义：

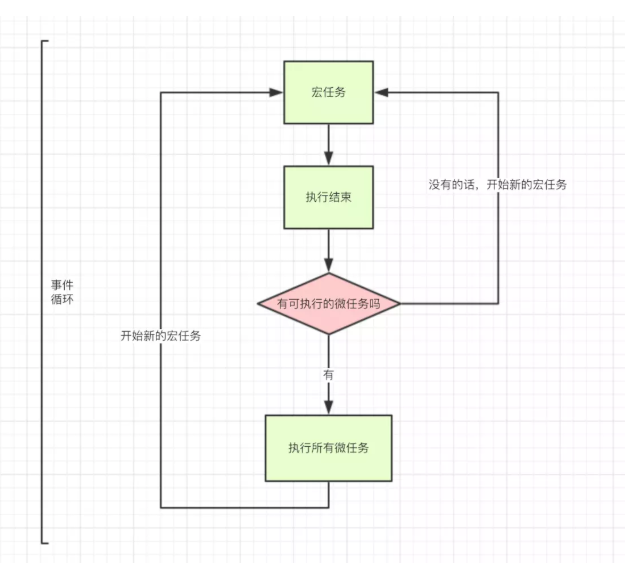
macro-task( 宏任务 )

包括整体代码script , setTimerout , setInterval

micro-task(微任务)

Promise , process.nextTick

不同类型的任务会进入相应的Event Queue， 例如setTimeout , setInterval 会进入相同的Event Queue



执行顺序

宏 --》 微