Node 中的事件循环模型

clearImmediate:清空立即执行函数
clearInterval:清除循环定时器
clearTimeout: 清除延迟定时器

setImmediate:设置立即执行函数
setInterval:设置循环定时器
setTimeout: 设置延迟定时器

第一个阶段：timers(定时器阶段--setTimeout,setInterval)
1.开始计时
2.执行定时器的回调(有定时器到时间才会执行，否则进入下一阶段)

第二个阶段：pending callbacks (系统阶段)

第三个阶段：idle, prepare (准备阶段)

第四个阶段：poll(轮询阶段，核心!!!)
---如果回调队列里有待执行的回调函数
从回调队列中取出回调函数(定时器在回调队列中但不取出定时器的回调执行)，同步执行(一个一个执行)，直到回调队列为空了，或者达到系统最大限度(少见)。
---如果回调队列为空
如果有设置过 setImmediate
进入下一个 check 阶段，目的：为了执行 setImmediate 所设置的回调。
如果未设置过 setImmediate
在此阶段停留，等待回调函数被插入回调队列。
若定时器到点了，进入下一个 check 阶段，原因：为了走第五阶段，随后走第六阶段，随后第一阶段(因为只有第一阶段才能执行定时器的回调)

第五个阶段：check (专门用于执行 setImmediate 所设置的回调)

第六个阶段：close callbacks (关闭回调阶段)

process.nextTick() ---- 用于设置立即执行函数(“VIP”-----能在任意阶段优先执行)

延迟定时器

```
setTimeout(()=>{
  console.log('setTimeout 所指定的回调函数执行了')
})
```

立即执行函数（回调）

```
setImmediate(()=>{
   console.log('我是 setImmediate 指定的回调')
})
```

立即执行函数（回调）

```
 process.nextTick(()=>{
    console.log('process.nextTick 所指定的回调执行了')
 })

 console.log('我是主线程上的代码')
```