# 后端笔记

## express的核心是

|  |
| --- |
| 路由、中间件、模板引擎... |

## express特点

## express和koa的区别

## nodejs内存泄露

## node创建子进程，进程间的通信

## node异步非阻塞 I/O 底层实现原理

## koa怎样实现中间件原理

## node适用场景

## node 中的 Event loop

|  |
| --- |
| Node 中的 Event loop 和浏览器中的不相同。  Node 的 Event loop 分为6个阶段，它们会按照顺序反复运行    1.timer  timers 阶段会执行 setTimeout 和 setInterval  一个 timer 指定的时间并不是准确时间，而是在达到这个时间后尽快执行回调，可能会因为系统正在执行别的事务而延迟。  下限的时间有一个范围：[1, 2147483647]，如果设定的时间不在这个范围，将被设置为1。  2.I / O  I / O 阶段会执行除了 close 事件，定时器和 setImmediate 的回调  3. idle, prepare  idle, prepare 阶段内部实现  4. poll  poll 阶段很重要，这一阶段中，系统会做两件事情  1.执行到点的定时器  2.执行 poll 队列中的事件  并且当 poll 中没有定时器的情况下，会发现以下两件事情  1.如果 poll 队列不为空，会遍历回调队列并同步执行，直到队列为空或者系统限制  2.如果 poll 队列为空，会有两件事发生  如果有 setImmediate 需要执行，poll 阶段会停止并且进入到 check 阶段执行 setImmediate如果没有 setImmediate 需要执行，会等待回调被加入到队列中并立即执行回调  如果有别的定时器需要被执行，会回到 timer 阶段执行回调。  5.check  check 阶段执行 setImmediate  6.close callbacks  close callbacks 阶段执行 close 事件  并且在 Node 中，有些情况下的定时器执行顺序是随机的  setTimeout(() *=>* {  *console*.log('setTimeout');  }, 0);  setImmediate(() *=>* {  *console*.log('setImmediate');  })  // 这里可能会输出 setTimeout，setImmediate  // 可能也会相反的输出，这取决于性能  // 因为可能进入 event loop 用了不到 1 毫秒，这时候会执行 setImmediate  // 否则会执行 setTimeout  上面介绍的都是 macrotask 的执行情况，microtask 会在以上每个阶段完成后立即执行。  setTimeout(() *=>* {  *console*.log('timer1')  *Promise*.resolve().then(*function* () {  *console*.log('promise1')  })  }, 0)  setTimeout(() *=>* {  *console*.log('timer2')  *Promise*.resolve().then(*function* () {  *console*.log('promise2')  })  }, 0)  // 以上代码在浏览器和 node 中打印情况是不同的  // 浏览器中打印 timer1, promise1, timer2, promise2  // node 中打印 timer1, timer2, promise1, promise2  Node 中的 process.nextTick 会先于其他 microtask 执行。  setTimeout(() *=>* {  *console*.log("timer1");  *Promise*.resolve().then(*function* () {  *console*.log("promise1");  });  }, 0);  process.nextTick(() *=>* {  *console*.log("nextTick");  });  // nextTick, timer1, promise1 |

## node 的 Event Loop: 6个阶

|  |
| --- |
| timer 阶段: 执行到期的setTimeout / setInterval队列回调  I/O 阶段: 执行上轮循环残流的callback  idle, prepare  poll: 等待回调  1.执行回调  2.执行定时器  如有到期的setTimeout / setInterval， 则返回 timer 阶段  如有setImmediate，则前往 check 阶段  check  执行setImmediate  close callbacks |

## nodejs的初衷是什么，有哪些不一样的特性

## node创建子进程，进程间的通信

## 负载均衡

## node异步非阻塞 I/O 底层实现原理

## Apache和Nginx的优缺点

## koa怎样实现中间件原理