Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行时。

特性：1).单线程 2).异步的非阻塞I/O 3).事件驱动

exports与module.exports的区别: \* exports是对module.exports的引用 \* 外界require导入模块时,真正导入的是module.exports而不是exports

事件循环是尽可能的将相应的操作分担给操作系统，从而让单线程的nodejs专注于非阻塞式的I/O操作。



阶段总览 （Phases Overview）

计时器（timers）：本阶段执行setTimeout() 和 setInterval() 计划的回调；

I/O 回调： 执行几乎全部发生异常的 close 回调， 由定时器和setImmediate()计划的回调；

空闲，预备（idle，prepare）：只内部使用；

轮询（poll）： 获取新的 I/O 事件；nodejs这时会适当进行阻塞；

检查（check）： 调用 setImmediate() 的回调；

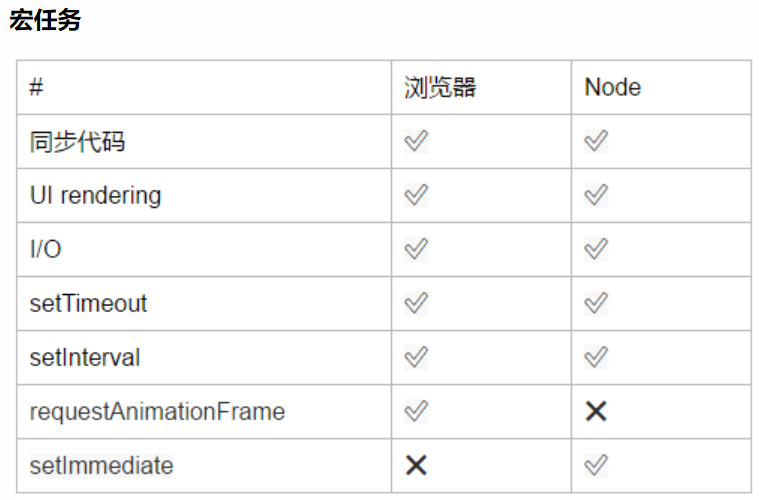
close callbacks： 例如 socket.on('close', ... );

在事件循环运行之间，Node.js 检查是否有正在等待的异步I/O 或者定时器，如果没有就清除并结束。

**2、CommonJS中require/exports和ES6中import/export区别**

　　CommonJS模块的重要特性是加载时执行，及脚本代码在require的时候，就会全部执行。一旦出现某个模块被“循环加载”就只输出已经执行的部分，还没有执行的部分是不输出的

　　ES6模块是动态引用，如果使用import从一个模块加载变量，那些变量不会缓存，而是成为一个指向被加载模块的引用,impor/export最终都是编译为require/exports来执行的



**浏览器**

先执行<script>中的同步任务，然后所有微任务，一个宏任务，所有微任务，一个宏任务......

执行完主执行线程中的任务；

取出 Microtask Queue 中任务执行直到清空；

取出 Macrotask Queue 中一个任务执行；

重复 2 和 3

**Node**

开始循环

循环中进行的操作：

清空当前循环内的 Timers Queue，清空 NextTick Queue，清空 Microtask Queue；

清空当前循环内的 I/O Queue，清空 NextTick Queue，清空 Microtask Queue；

清空当前循环内的 Check Queue，清空 NextTick Queue，清空 Microtask Queue；

清空当前循环内的 Close Queue，清空 NextTick Queue，清空 Microtask Queue；

进入下轮循环。

while (true) {

宏任务队列.shift();

微任务队列全部任务();}

while (true) {

loop.forEach((阶段) => {

阶段全部任务();

nextTick全部任务();

microTask全部任务();

});

loop = loop.next;}