1. 多重循环跳出break mutipleAnser([‘a’,’b’,’c’],[‘c’,’a’]) //[1,0,1]
2. //格式化多选题提交数据 选上的为1 不选上的为0
3. mutipleAnser(originList,resultList) {
4. let tempList = originList.slice()
5. one:
6. for(let i=0;i<tempList.length;i++) {
7. two:
8. for(let j=0;j<resultList.length;j++) {
9. if(resultList[j] === tempList[i]) {
10. tempList[i] = 1
11. break two;
12. }
13. }
14. }
15. for(let i=0;i<tempList.length;i++) {
16. if(tempList[i] !== 1) {
17. tempList[i] = 0
18. }
19. }
20. return tempList
21. }

2.async await语法让异步代码看起来更像同步代码，体现在async异步函数内部，从外部看async依旧是一个异步操作



3. await命令后面的 Promise 对象如果变为reject状态，则reject的参数会被catch方法的回调函数接收到。

    任何一个await语句后面的 Promise 对象变为reject状态，那么整个async函数都会中断执行（相当于抛出错误throw xxx）。



3. 前面已经说过，await命令后面的Promise对象，运行结果可能是rejected，所以最好把await命令放在try...catch代码块中。



4. 多个await命令后面的异步操作，如果不存在继发关系，最好让它们同时触发。



上面代码中，getFoo和getBar是两个独立的异步操作（即互不依赖），被写成继发关系。这样比较耗时，因为只有getFoo完成以后，才会执行getBar，完全可以让它们同时触发。



上面两种写法，getFoo和getBar都是同时触发，这样就会缩短程序的执行时间。

5. await命令只能用在async函数之中，如果用在普通函数，就会报错。



上面代码会报错，因为await用在普通函数之中了。但是，如果将forEach方法的参数改成async函数，也有问题。

正确的写法是采用for循环。

