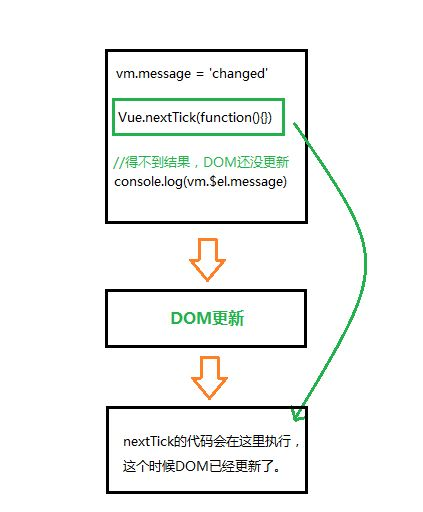
8.谈谈你对Vue.nextTick的认识

官方文档解释：在**下次dom更新**循环结束之后执行**延迟回调**。在**修改数据**之后立即使用这个方法，获取更新后的dom。简单来说，Vue 在修改数据后，视图不会立刻更新，而是等同一事件循环中的所有数据变化完成之后，再统一进行视图更新。

因为Vue的dom是**异步更新**的，具体步骤如下：

1. 所有同步任务都在主线程上执行，形成一个执行栈
2. **model层修改数据**之后，watcher监听到数据变化，会将**dom操作**推入事件队列(异步)中；
3. 等到**主线程任务**执行完毕之后,Vue开始执行异步队列中的回调；
4. 当**dom更新完毕**之后执行Vue.nextTick的回调函数，此时视图层已经是最新的dom。  
   该方法一般用在**created，mounted钩子函数中**，因为**mounted不能保证所有子组件已经挂载**，所以如果想在上面两个**钩子函数中操作dom**最好使用Vue.nextTick。另外，可以在函数中修改数据后使用该方法对更新后的dom进行操作。

vm.message = 'changed' //改变数据

console.log(vm.$el.textContent) //想要立即使用更新后的DOM。这样不行，因为设置message后DOM还没有更新 .并不会得到'changed'

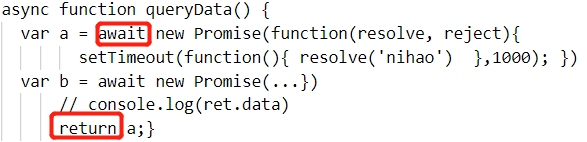
//nextTick里面的代码会在DOM更新后执行 Vue.nextTick(function(){ console.log(vm.$el.textContent) //可以得到'changed' })

9.  JS中常见的异步调用  - 定时任务  - ajax  - 事件函数

10.JWT是一个轻量级的跨域认证规范,允许我们使用JWT在用户服务器间传递信息(后端.sign()生成)

11.async关键字

1)普通函数定义前加async关键字 普通函数变成异步函数 异步函数默认返回promise对象

2)在异步函数内部使用**return**关键字进行结果返回,结果会被包裹的promise对象中 return关键字代替了resolve方法

3)在**异步函数内部**使用throw关键字抛出程序异常

resolve->return->then

5)调用**异步函数**再**链式调用**then方法获取**异步函数执行**结果

6)调用异步函数再链式调用catch方法获取异步函数**执行的错误**信息

async/await 让异步代码看起来、表现起来更像同步代码,await用于等待一个异步函数执行完成

异步函数:通过时间循环异步执行的函数

await关键字:强制其他代码等待，直到Promise完成并返回结果

1) await关键字只能出现在异步函数中

2)await promise await后面只能写promise对象 写其他类型的API是不不可以的

3)await关键字可是暂停异步函数

async/await相⽐于Promise的优势

代码读起来**更加同步**，Promise虽然摆脱了回调地狱，但是**then的链式调⽤**也会带来额外的阅读负担 Promise传递中间值⾮常麻烦，⽽async/await⼏乎是同步的写法，⾮常优雅

11.HTTP的keep-alive是⼲什么的？

加⼊Connection: keep-alive来告诉对⽅这个请求响应完成后不要关闭

1. HTTPS是如何保证安全的 对称加密：即通信的双⽅都使⽤同⼀个秘钥进⾏加解密
2. 事件委托 利⽤事件冒泡,,将事件委托给父节点 **节省内存**占⽤，减少事件注册

新增⼦对象时⽆需再次对其绑定事件，适合动态添加元素

14.浏览器重绘与重排的区别？

重排:渲染树要**按照样式**(节点,尺寸)重新布局 重绘:位置,背景颜色,需要更新

『重绘』不⼀定会出现『重排』，『重排』必然会出现『重绘』